PONTIFICIA UNIVERSIDAD CATÓLICA DEL ECUADOR FACULTAD DE INGENIERÍA MAESTRÍA EN REDES Y COMUNICACIONES

ESTUDIO Y EVALUACIÓN DEL DESEMPEÑO DE TCP SOBRE REDES DE CONMUTACIÓN ÓPTICA DE RÁFAGAS (OBS, *OPTICAL BURST SWITCHING)* MEDIANTE UN PROTOTIPO BASADO EN SIMULACIÓN

DISERTACIÓN PREVIA A LA OBTENCIÓN DEL TITULO DE MAGISTER EN REDES DE COMUNICACIONES

Ing. ANTONIO JAVIER VENEGAS LÓPEZ

DIRECTORA:

MSc. MARÍA SOLEDAD JIMÉNEZ

Quito, abril de 2017

CERTIFICACIÓN

Certificamos que el presente proyecto de Disertación **"Estudio y evaluación del desempeño de TCP sobre redes de Conmutación Óptica de Ráfagas (OBS, *OPTICAL BURST SWITCHING*) mediante un prototipo basado en simulación"**, fue desarrollado en su totalidad por el señor Antonio Javier Venegas López, bajo nuestra dirección, como requerimiento previo a la obtención del título de MAGISTER EN REDES DE COMUNICACIONES.

Quito, 17 de abril de 2017

MSc. María Soledad Jiménez
DIRECTORA DEL PROYECTO

Ph. D. Gustavo Chafla
REVISOR DEL PROYECTO

MSc. Juan Francisco Chafla
REVISOR DEL PROYECTO

AUTORÍA

Yo, Antonio Javier Venegas López, portador de la cédula de ciudadanía Nº 171302604, declaro que el presente proyecto de investigación es de total responsabilidad del autor, y que se han respetado las diferentes fuentes de información realizando las citas correspondientes.

Ing. Antonio Venegas López

AGRADECIMIENTO

A Dios por su infinito amor y compañía en cada instante de mi vida, fortaleciendo y llenando cada espacio de mi espíritu con sabiduría, perseverancia y deseos de superación, que han sido los pilares fundamentales para alcanzar mis objetivos.

A mi adorada esposa, la señora MSc. Verónica Almeida y mis a amados hijos Paula Venegas y Sebastián Venegas por todo su apoyo y comprensión durante todo el tiempo invertido para el desarrollo y conclusión de este proyecto de investigación, ya que con su amor y ternura me impulsaban día a día para continuar adelante con el cumplimeinto de esta meta.

A mis padres, el señor Antonio Venegas y la señora Beatriz López por haberme educado con con mucho amor y cariño, inculcándome siempre valores y principios sólidos, y por haberme brindado la oportunidad de estudiar una carrera universitaria, que ha constituido mis primeros pasos de mi carrera académica y profesional.

A mis suegros, el señor Aparicio Almeida y la señora Susana Vargas por el apoyo brindado a mi familia, durante los espacios de tiempo en los que estuve dedicado al desarrollo de este proyecto de disertación.

A los abuelitos de mi esposa, la señora Blanca Mora y el señor César Vargas que en paz descanse, por la confianza y el apoyo brindado para que haya podido iniciar este programa de maestría.

A mi profesora y directora, MSc. María Soledad Jiménez, por su guía, apoyo y motivación brindados para la elaboración del presente proyecto de tesis.

A mis profesores y revisores de tesis, Dr. Gustavo Chafla y MSc. Juan Francisco Chafla por el conocimiento y experiencia impartidos, así como también por sus recomendaciones y sugerencias para la conclusión de este proyecto de estudio.

A la Pontifica Universidad Católica del Ecuador, institución superior de excelencia, por la oportunidad para realizar mis estudios de cuarto nivel y formar parte de ella, como un profesional con un nivel de preparación científica y académica.

DEDICATORIA

A Dios por su inmenso amor y bondad, y por llenar mi vida de muchas bendiciones y sabiduría, que han sido los pilares fundamentales que me han permitido llegar a este hito tan importante de mi carrera académica y profesional.

A mi adorada esposa, la señora MSc. Verónica Almeida y a mis amados hijos Paula Venegas y Sebastián Alejandro que han sido el principal motor que motiva mi vida y que me da fuerzas para seguir adelante.

A mis padres, el señor Antonio Venegas y la señora Beatriz López por su ejemplo y guía, que con amor, cariño, dedicación y sacrifico supieron siempre darme lo mejor de ellos, en cada instante de mi vida para llegar a lo que ahora soy.

A mis hermanos, el señor MSc. Carlos Venegas y la señorita MSc. Paola Venegas por el ejemplo de dedicación y superación que me motivaron para seguir adelante con este reto.

A mis aluelitos, el señor Manuel López, la señora Beatriz Paredes, el señor Oswaldo Venegas y la señora Piedad Gómez que en paz descansen y a quienes les recuerdo con mucho cariño y siempre les llevo presente en mi corazón.

CONTENIDO

LISTA DE FIGURAS

LISTA DE TABLAS

LISTA DE ANEXOS

RESÚMEN

El entorno dinámico en el cual el tráfico de datos es el principal protagonista, ha conducido a un crecimiento sustancial en la demanda de ancho de banda en los últimos años, y el paso del dominio de la voz al dominio de los datos hace que el costo de red de las tecnologías actuales, no soporten el crecimiento esperado de forma rentable, por lo tanto, es evidente una evolución de las redes de datos que permita satisfacer, al costo beneficio deseado, las necesidades actuales y futuras de los usuarios que cada día requieren mayores capacidades y esperan un menor precio. Es así que, el reto principal que deben afrontar los operadores de telecomunicaciones se centra en un aumento acelerado de las capacidades a transportar sobre sus redes y un crecimiento moderado de sus ingresos; tal escenario implica obviamente, cambios tecnológicos para proveer una plataforma de red multi-servicios que permita la adaptación de distintos formatos de datos, el soporte de altas tasas de transmisión, y una provisión flexible de anchos de banda, que se consideran factores clave para el Internet de próxima generación.

Ante esta realidad, la tecnología óptica ha permitido a través del desarrollo de DWDM (*Dense Wavelength Division Multiplexing*) incrementar significativamente el ancho de banda de las trasmisiones por fibra óptica alcanzando velocidades de hasta decenas de Terabits/seg en el nivel de transporte; sin embargo la conmutación de paquetes utilizada en Internet por los equipos de inter*networking*, efectúan su procesamiento de forma electrónica (donde es necesaria la conversión opto-eléctrica y electro-óptica), lo cual constituye un cuello de botella con relación a las altas velocidades de transmisión alcanzadas en el nivel óptico, en especial con DWDM que sigue en desarrollo con tendencia a alcanzar cada vez mayores velocidades al multiplexar más canales por hilo de fibra óptica; esta tecnología se puede considerar como la base para el transporte de información que se encuentra en su etapa de evolución hacia las denominadas redes todo ópticas, dando lugar a la aparición de técnicas de conmutación óptica como: conmutación óptica de circuitos (OCS, *Optical Circuit Switching*), conmutación óptica de paquetes (OPS, *Optical Packet Switching*) y conmutación óptica de ráfagas (OBS, *Optical Burst Switching*).

La tecnología OBS se considera una técnica de conmutación óptica de tercera generación dentro de la escala evolutiva del *networking* óptico, visualizada como un paradigma prometedor para el Internet Óptico de nueva generación, que podría ser adoptado para el transporte de información en el nivel de *core*, con el fin de minimizar el procesamiento electrónico para superar el cuello de botella de los enrutadores actuales, y a su vez enrutar y conmutar a nivel completamente óptico grandes volúmenes de información ensamblados en ráfagas. Esta tecnología se encuentra actualmente en estado de desarrollo e investigación y al respecto se han efectuado una serie de estudios que han conducido a la definición de modelos matemáticos, esquemas de señalización, algoritmos de planificación y técnicas de resolución de contenciones validados mediante simulaciones por computador, con el fin de evaluar el desempeño de este tipo de redes y su adaptabilidad al tráfico con naturaleza a ráfagas de Internet, encontrándose resultados interesantes así como también ciertas restricciones especialmente en cuanto al tema de contenciones, debido a la limitación del *buffer*ing óptico propio de la inmadurez de la tecnología fotónica actual.

Por otro lado, varios estudios han mostrado que aproximadamente el 90% de las aplicaciones en Internet utilizan TCP, por lo cual se lo puede considerar como, el mayor referente del volumen de tráfico que cursa a través del Internet y que se prevé continuará siendo el protocolo de transporte dominante. Por lo tanto, se considera importante evaluar su desempeño en redes que puedan emplear a futuro la tecnología de conmutación óptica de ráfagas, donde a simple vista es necesario evaluar el impacto de la agregación de paquetes en ráfagas y el efecto de la limitación de almacenamiento temporal (*buffering*) óptico en los nodos de *core*, sobre el desempeño de TCP, ya que actualmente no existe un equivalente óptico a la memoria RAM electrónica, indispensable para la conmutación de paquetes utilizada en las redes de datos existentes.

En este contexto, se evalúa el desempeño del protocolo TCP sobre un modelo de red OBS mediante el simulador de redes *ns-2* (*network simulator 2*), utilizando las variantes más comunes basadas en pérdidas de este protocolo, tales como TCP Reno, Newreno y SACK; y se contrastan los resultados con los modelos analíticos basados en los procesos regenerativos y teoría de renovación de Markov, propuestos en la litertura.

La realización de este proyecto de tesis se justifica técnicamente porque pretende efectuar un estudio moderado de la tecnología OBS para redes ópticas de nueva generación que se encuentran en desarrollo e investigación, complementado el fundamento teórico con un componente práctico que consiste en la implementación de un prototipo basado en simulación, que permitirá ampliar la comprensión de esta tecnología y modelar un determinado sistema a fin de obtener una aproximación lo más cercana a la realidad.

ABSTRACT

The dynamic environment in which data traffic is the main protagonist has led to a substantial growth in the bandwidth demand during the last years, and the change from voice dominance to data dominance makes that cost of current technologies, doesn't support the expected growth in a profitable way, therefore, an evolution of the data networks is evident that allows to satisfy, at desired cost-benefit balance, the current and future needs of the users that requires every day higher capacities and expect a lower price. Thus, the main challenge facing telecommunications providers is the accelerated increase in the capacity to carry over their networks and a moderate increase in their revenues; Such a scenario obviously involves technological changes to provide a multi-service network platform that allows the adaptation of different data formats, the support of high transmission rates, and a flexible provision of bandwidth which are considered key factors for the Next-generation Internet.

Faced with this reality, optical technology has enabled the development of DWDM (*Dense Wavelength Division Multiplexing*) to significantly increases the bandwidth of optical fiber transmissions, reaching rates up to tens of Terabits/sec at the transport level; However, the packet switching used by the inter*networking* equipment is electronically processed (where conversion opto-electronic and electro-optical is necessary), which is a bottleneck in relation to high transmission rates accomplished at optical level, especially with DWDM that is still developing, with a tendency to reach higher rates by multiplexing more channels per fiber optic cable; this technology can be considered as the basis for transport of information, which is in its evolution stage towards the so-called all-optical networks, giving rise to the appearance of optical switching techniques such as: OCS (*Optical Circuit Switching*), OPS (*Optical Packet Switching*) and OBS (*Optical Burst Switching*).

OBS technology is considered a third-generation optical switching technique within the evolutionary scale of optical *networking*, visualized as a promising paradigm for the new generation Optical Internet, which could be adopted for transport of information at the *core* level, in order to minimize the electronic processing to overcome the bottleneck in current routers and in turn routing and switchig at full optical level, large information volumes

assembled in bursts. This technology is currently under development and research and in this respect a series of studies have been carried out, leading to the development of mathematical models, signaling schemes, planning algorithms and contention resolution techniques validated by computer simulations, in order to evaluate the performance of this network type and its adaptability to the Internet traffic, with interesting results as well as certain restrictions, especially regarding the contention issue due to the limitation of optical *buffer*ing, characteristic of immaturity of the current photonic technology.

On the other hand, several studies have shown that approximately 90% of the applications on Internet use TCP, so it can be considered as the largest referent of the volume of traffic through the Internet and is expected to continue to be the dominant transport protocol. Therefore, it is considered important to evaluate its performance in networks that may be used in the future optical burst switching technology, where at a glance, it is necessary to evaluate the impact of packet aggregation in bursts and effect of temporary storage limitation (*optical buffering*) at the *core* nodes, on TCP performance, since there is currently no optical equivalent to electronic RAM, essential in the packet switching t used in existing data networks.

In this context, the performance of TCP protocol on an OBS network model is evaluated by *ns-2* (*network Simulator 2*), using the most common variants based on losses of this protocol such as TCP Reno, Newreno and SACK; and the results are contrasted with the analytical models based on the Markov regenerative processes and renewal theory, proposed in the literature.

The development of this thesis project is technically justified because it intends to carry out a moderate study of the OBS technology for new generation optical networks that are in development and research, complementing the theoretical foundation with a practical component which consists in the implementation of a simulation-based prototype, that will allow to extend the understanding of this technology and modeling a certain system in order to obtain an approximation as close to reality as possible.

OBJETIVO GENERAL

- Estudiar y evaluar el desempeño de TCP sobre redes OBS mediante una herramienta de simulación por ordenador.

OBJETIVOS ESPECÍFICOS

- Presentar las características principales de las diferentes técnicas de conmutación óptica, haciendo énfasis en la tecnología OBS en cuanto al proceso de ensamblado de ráfagas, esquemas de reserva y liberación de recursos, planificación de ráfagas, resolución de contenciones, mecanismos para la recuperación de pérdidas y demostradores implementados en el transcurso del tiempo.

- Validar el modelo analítico de TCP Reno sobre OBS basado en la teoría de procesos regenerativos de Markov definido para fuentes de clase lenta y rápida; y contrastar los resultados mediante la herramienta de simulación *Network Simulator (ns-2)*, ampliamente adoptada en el ámbito académico y de investigación.

- Validar el modelo analítico de TCP sobre OBS basado en la teoría de renovación, definido para una fuente de velocidad genérica; y contrastar los resultados mediante la herramienta de simulación *Network Simulator (ns-2)*, enfocando el análisis a las tres implementaciones más comunes de TCP basadas en pérdidas como son Reno, New Reno y SACK.

CAPÍTULO I INTRODUCCIÓN A LAS ARQUITECTURAS DE TRANSPORTE ÓPTICO DE NUEVA GENERACIÓN

La demanda continuamente creciente de mayor ancho de banda ha experimentado una evolución exponencial en los últimos años, debido principalmente a la proliferación del acceso a Internet y a la adopción cada vez mayor de aplicaciones intensivas en banda ancha como transferencias de archivos de gran tamaño, streaming de audio/video, juegos multi-participante en línea, *cloud computing*, distribución de contenido y muchas otras aplicaciones multimedia con exigencias en tiempo real, que precisan mayores capacidades y tasas de transmisión en el *backbone* de la red para poder satisfacer los requerimientos actuales y futuros.

En este capítulo se presenta una introducción a la evolución que ha tenido el *networking* óptico desde sus inicios, describiendo las tres diferentes generaciones a las que ha conducido el desarrollo de la tecnología óptica, empleada mayoritariamente en el ámbito de las redes troncales, y que en el transcurso de los años se ha ido expandiendo también al entorno de las redes metropolitanas y de acceso. Además, se presentan algunos desarrollos tecnológicos recientes y desafíos técnicos en los sistemas de transmisión de alta capacidad, finalizando con una breve descripción y comparativa entre las diferentes tecnologías de conmutación óptica propuestas para sistemas basados en WDM (*Wavelength Division Multiplexing*).

1.1 Evolución del *networking* óptico [5][12][27][36-37][47][61-62][75] [80,82,84,88][102,106,108,113]

El entorno dinámico en el cual el tráfico de datos es el principal protagonista, ha conducido a un crecimiento sustancial en la demanda de ancho de banda en los últimos años, y el paso del dominio de la voz al dominio de los datos, hace que el costo de red de las tecnologías actuales, no soporten el crecimiento esperado de forma rentable, por lo tanto, es evidente una evolución de las redes de datos que permita satisfacer, al costo beneficio deseado, las necesidades actuales y futuras de los usuarios que cada día requieren mayores capacidades y esperan un

menor precio. Es así que, el reto principal que deben afrontar los operadores de telecomunicaciones se centra en un aumento acelerado de las capacidades a transportar sobre sus redes y un crecimiento moderado de sus ingresos; tal escenario implica obviamente, cambios tecnológicos para proveer una plataforma de red multi-servicios que permita la adaptación de distintos formatos de datos, el soporte de altas tasas de transmisión y una provisión flexible de anchos de banda, que se consideran factores clave para el Internet de próxima generación.

La Figura 1.1 muestra el crecimiento global del tráfico mundial de Internet desde sus inicios, que superó al tráfico TDM (*Time Division Multiplexing*), principalmente llamadas telefónicas, en el año 2002. Para el año 2015 el tráfico IP global se prevé que sea aproximadamente ocho órdenes de magnitud mayor al que era en 1990 [62]. Dado que el tráfico IP tiene tan elevado crecimiento mientras que el tráfico TDM tiene sólo unos cuantos puntos porcentuales de la tasa de crecimiento anual, la mayoría del tráfico en estos días es casi todo en el formato de datos IP, que en el año 2011 alcanzó a nivel global un valor de 30 exabytes por mes [62].

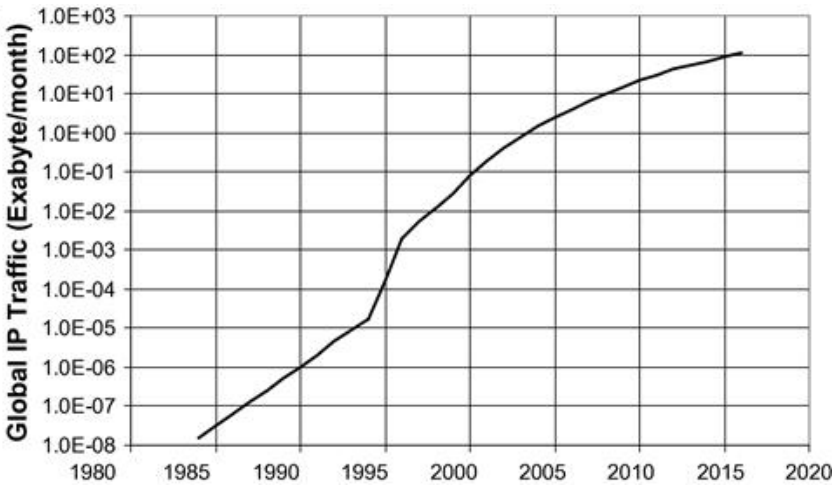


Figura 1.1Demandas de tráfico IP global, los datos a partir del 2011 son estimados [62]

Además de las crecientes demandas de usuarios residenciales y empresariales, las nuevas necesidades de usuarios científicos están impulsando el desarrollo de nuevas tecnologías para las redes ópticas del futuro. Ejemplos de estos consumidores de ancho de banda son [84]:

Usuarios residenciales.- Los requerimientos actuales de ancho de banda para los usuarios domésticos fijos se pueden estimar en 100 Mbps y hasta unas cuantas centenas de Mbps (de descarga), derivados principalmente de la demanda que surge de los servicios *"Triple Play"* que incluyen voz, datos y video (en alta definición). En cuanto a los usuarios móviles, con la introducción de la tecnología 4G LTE que permite incrementar el ancho de banda de acceso se estiman tasas superiores a 20 Mbps y hasta 75 Mbps en este modelo, que hoy en día se pueden encontrar en distintos paquetes comerciales ofrecidos por diferentes operadores de telecomunicaciones en varios países del mundo. Cabe mencionar que estas cifras serían mucho mayores si la televisión de ultra alta definición (UHDTV, *Ultra High Definition TeleVision*) también llegara a ser un servicio de difusión, moviendo los requerimientos de velocidad de acceso al régimen de los gigabits por segundo.

Grandes usuarios empresariales.- Este tipo de usuarios requieren acceso a altos anchos de banda simétricos (p. ej. hasta 10 Gbps) para redes privadas virtuales, recuperación de desastres, almacenamiento, etc.

Usuarios científicos.- Algunos ejemplos de aplicación incluyen los siguientes:

1) *Física de partículas de gran energía* que puede generar conjuntos de datos medidos en decenas de peta bytes por año, que sólo se pueden procesar y analizar mediante recursos informáticos distribuidos globalmente, para lo cual se necesita el transporte de un conjunto de datos de 10 – 100 TB, estos últimos con un requerimiento en *throughput* de 10 Gbps para la entrega de resultados en 24 horas.

2) *Interferometría de muy larga base* (VLBI, *Very Long Baseline Interferometry*) empleada por los radio astrónomos para obtener imágenes detalladas de fuentes de radio cósmicas, donde la combinación de señales de dos o más radio telescopios pueden crear eficientemente un instrumento con una potencia de resolución proporcional a su separación espacial. *e-VLBI* utilizará redes de alta velocidad para transferir datos del telescopio a un correlador a varios gigabits por segundo (10 Gbps – 40 Gbps).

3) *e-health* que supone un desafío para las tecnologías de la información debido al tamaño y cantidad de imágenes, con redes requeridas para transportar 1.2 GB de datos cada 30 seg, así

como también la disponibilidad de servicios de redes ópticas que ofrezcan garantías en tiempo real.

Ante esta realidad, los operadores de telecomunicaciones han desplegado redes de transporte óptico de alta capacidad, basadas principalmente en el desarrollo de la tecnología de multiplexación por división de longitud de onda, particularmente DWDM (*Dense Wavelength Division Multiplexing*), que ha permitido incrementar significativamente el ancho de banda de las transmisiones por fibra óptica alcanzando velocidades de hasta decenas de Terabits/seg en el *backbone* de la red. Sin embargo, los equipos de *internetworking* que basan su funcionamiento en la técnica de conmutación de paquetes utilizada actualmente en Internet, realizan el procesamiento, enrutamiento y almacenamiento de información en el dominio electrónico; y aunque ha habido un gran esfuerzo en el desarrollo de enrutadores IP electrónicos de alta velocidad basados en hardware, que pueden gestionar un tráfico agregado en el rango de Terabits por segundo, la tecnología DWDM puede ofrecer capacidades superiores, y su tendencia es conseguir cada vez mayores velocidades al multiplexar más canales por hilo de fibra óptica, o incrementar a su vez la capacidad de transmisión que se puede transportar sobre cada portadora óptica. Por lo tanto, existe un desajuste entre la capacidad de la línea y la capacidad del enrutador, lo cual puede conducir a que estos dispositivos electrónicos gradualmente sean ineficaces para cumplir con los requerimientos de tráfico necesarios y más bien se puede esperar un cuello de botella en términos de escalabilidad y crecimiento en comparación con las altas velocidades de transmisión alcanzadas en el nivel óptico.

Aunque el procesamiento electrónico permite una elevada flexibilidad y puede incluir una gran variedad de funciones avanzadas, tiene una necesidad de consumo de potencia que aumenta con la velocidad de procesamiento. Según el estudio realizado en [88], la potencia consumida por un enrutador es proporcional a $C^{2/3}$, donde C la capacidad del enrutador en Mbps. El problema se debe a que mientras el rendimiento de los semiconductores, pieza clave de la electrónica, se duplica cada 18 meses (por la conocida Ley de Moore), el tráfico se duplica anualmente. Por lo tanto, la tecnología óptica y en concreto la aparición de los conmutadores ópticos ha supuesto una alternativa mucho más eficiente que su contraparte electrónica para la conmutación de datos, puesto que necesitan menos energía para conmutar altas tasas de bits y para interconectar subsistemas más distantes [12]. Es así que la fabricación

de enrutadores IP Terabit/s electrónicos está sujeta a altos costos y no se considera como una solución viable para las redes troncales.

Por el contrario, los enrutadores ópticos basados en el paradigma IP sobre WDM a través del uso de nuevas tecnologías, tendrán una mejor escalabilidad y serán más simples ya que IP sobre WDM evita cierta redundancia encontrada en otras tecnologías de transporte, tales como IP sobre ATM e IP sobre SONET/SDH [37].

Hoy en día, el término "redes ópticas" denota a redes de telecomunicaciones de alta capacidad basadas en las tecnologías y componentes ópticos que pueden proporcionar ancho de banda, aprovisionamiento, enrutamiento, *grooming*[1], o restauración a nivel de longitud de onda. Es bien conocido que el *networking* óptico a través del incremento de capacidad que ofrece a un coste reducido por bit transmitido por kilómetro, en comparación con otras tecnologías de *networking* (como redes inalámbricas o redes cableadas basadas en cobre, cable coaxial o líneas eléctricas) ha permitido la revolución del Internet que inició a mediados de 1990 y ha resultado en un enorme impacto en nuestra sociedad. Las soluciones de *networking* óptico abarcan toda la gama de las redes de transporte, y sin duda está dominando los segmentos de *core* y *metro*, y más recientemente también ha iniciado su expansión al segmento de la red de acceso. Las redes troncales y metropolitanas transportan tráfico de datos agregado de gran velocidad cubriendo distancias que pueden abarcar desde el tamaño de una ciudad hasta todo un continente. Las redes de acceso transportan diferentes tipos de flujos de datos hacia y desde clientes particulares y/o empresariales, que son multiplexados/demultiplexados en nodos que tienen conexiones fijas de *backhaul*[2] con la red troncal de transporte.

La historia del *networking* óptico se remonta a la invención del láser por Schawlow y Townes en 1958 seguido por el trabajo de Kao y Hockham en fibras ópticas en 1965 y la subsecuente demostración de la fibra óptica como un medio de comunicación práctico por Maurer, Keck, Schultz, y Zimar en 1970, dio vida a una plataforma tecnológica capaz de soportar los

[1] Se refiere a la multiplexación eficiente, utilizada normalmente para agrupar flujos de tráfico de menor velocidad en flujos de tráfico de mayor velocidad con el fin de minimizar los requerimientos de capacidad de toda la red.

[2] En telecomunicaciones, un *backhaul* es un enlace de interconexión entre redes de datos o redes de telefonía móvil (celular). Pueden ser llevados a cabo utilizando conexiones de baja, media o alta velocidad y por medio de tecnologías alámbricas o inalámbricas.

requerimientos de comunicación nacionales y mundiales para el siglo XXI y más allá. A finales de 1970, la fibra empezó a reemplazar el cable coaxial como el medio de transmisión en los sistemas troncales de las redes de telecomunicaciones trayendo muchas ventajas tanto técnicas como económicas. La creación del Internet (con TCP/IP) en 1983, y posteriormente la red mundial global, WWW (*World Wide Web*) en 1993 desencadenó el crecimiento del tráfico de datos en la red, que como se mencionó anteriormente superó al tráfico de voz alrededor del año 2002 [61]. En la década de 1985 a 1995, cuatro eventos significativos anunciaron la posibilidad del *networking* óptico donde tanto la transmisión como la conmutación podían ser basadas en el nivel óptico. Estos fueron 1) la invención de los amplificadores ópticos que permitió 2) el desarrollo económico de WDM, 3) la demostración de un conmutador óptico capaz de permitir una rápida configuración de caminos de luz basados en canales de longitud de onda, y 4) la convergencia de servicios y velocidad de transmisión de transporte óptico alrededor del año 2000 (ver Figura 1.2a) a una tasa de 10 Gbps, que abrió la posibilidad de interconexión directa entre una red IP y una red de transporte óptico empleando WDM, donde la granularidad de la red se adaptó directamente a la tasa de la interfaz del enrutador [61]. Este fue un paso importante en la evolución de las redes ópticas ya que el flujo de salida de un enrutador podía circular directamente sobre un canal de longitud de onda. La Figura 1.2a también muestra la convergencia a 40 Gbps que ocurrió alrededor del año 2005 con la disponibilidad de enrutadores de 40 Gbps y sistemas de transmisión DWDM diseñados para 40 Gbps, que para el caso de los equipos de datos fue estandarizada posteriormente junto con 100 Gbps en el año 2010 [61].

La historia de la implementación de los estándares de la capacidad del canal de transporte óptico y los estándares de la velocidad del puerto *Ethernet* se muestra en la Figura 1.2b, donde se puede apreciar que la velocidad del puerto *Ethernet* ha crecido desde 10 Mbps a 100 Gbps con un aumento de 10 veces (excepto para el estándar 40 Gbps) mientras que los estándares de capacidad del canal óptico han crecido desde 155 Mbps (OC-3) a 100 Gbps (OTU-4) con un incremento de 4 veces (excepto para el estándar 100 Gbps), y a medida que la demanda de tráfico ha seguido aumentando, los proveedores de servicios de redes de telecomunicaciones han introducido sistemas de transporte coherentes con capacidades de 100 Gbps e incluso

superiores, desplegados recientemente en sus redes para satisfacer los requerimientos de tráfico actuales y futuros [62].

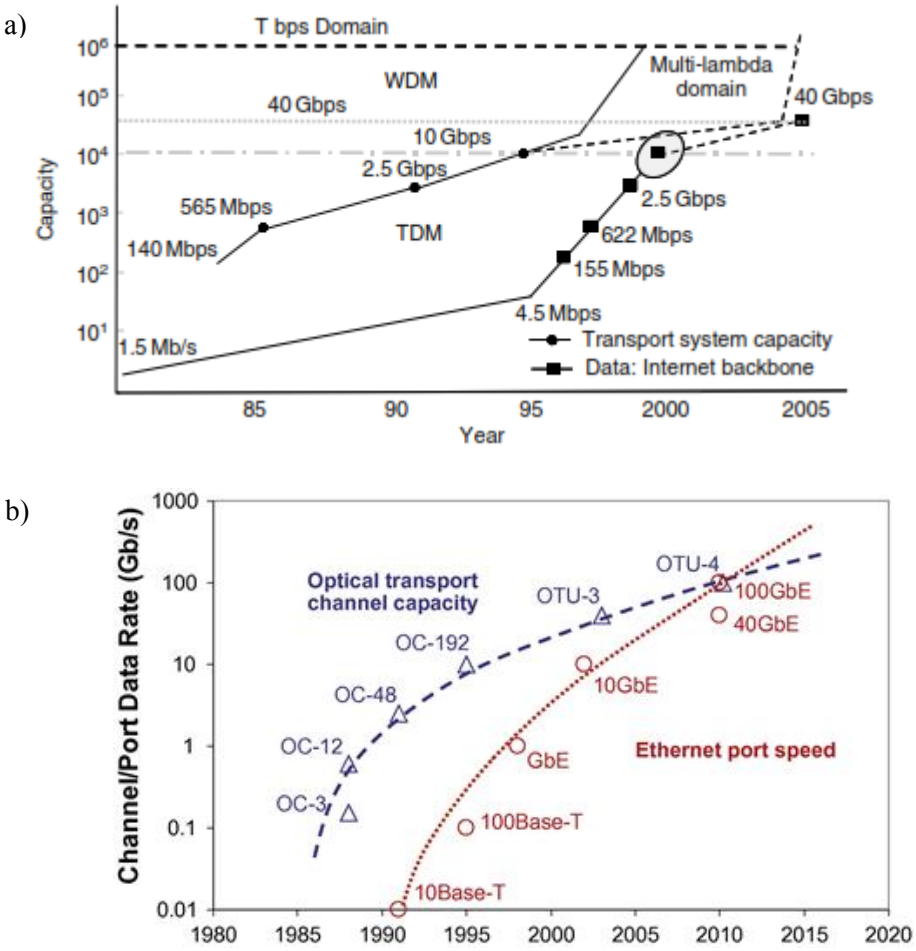


Figura 1.2 a) Servicios y velocidades de línea [61]; b) Estándares de las capacidades del canal de transporte y velocidades del puerto Ethernet [62]

La tecnología óptica ofrece muchos beneficios potenciales que pueden ser apreciados en la Figura 1.3, entre los cuales se pueden mencionar los siguientes [61]:

Transmisión.- Los métodos ópticos permiten sistemas de alta capacidad a un bajo costo en base a canales de alta velocidad combinados con la tecnología DWDM.

Conmutación.- El principal beneficio de la conmutación óptica es la energía y tamaño reducidos en comparación que su contraparte electrónica para el mismo *throughput*. Este es un

factor muy importante ya que las estaciones de intercambio de tráfico son a menudo limitadas en espacio, especialmente dentro de los entornos urbanos.

Interoperabilidad.- Existen muchas comunidades diferentes de usuarios, distribuidas globalmente que desean interconectarse (p. ej. *grid computing*[3]) para propósitos de experimentos o compartición de recursos. La interoperabilidad es un asunto clave ya que cada dominio de red puede utilizar diferentes tecnologías a nivel de los planos de datos y de control. Es probable que las redes basadas en tecnologías ópticas ofrezcan mayor facilidad de interoperabilidad, por ejemplo a través de, explotar la transparencia (las señales permanecen en el dominio óptico a través de la red), permitir que las longitudes de onda en la frontera de la red sean alineadas dinámicamente a otro dominio, o soportar el control distribuido mediante el uso de rápidos conmutadores ópticos.

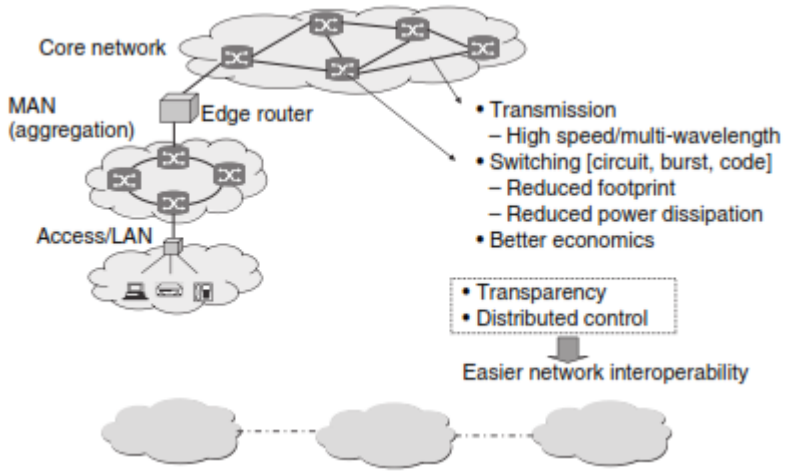


Figura 1.3 Tecnología óptica y la red [61]

[3] *Grid computing* es una arquitectura que permite utilizar coordinadamente los recursos informáticos de hardware y software (p. ej. potencia de procesamiento, almacenamiento, bases de datos, aplicaciones y servicios funcionales, etc.) de distintos sistemas de computación como supercomputadoras, clústers, etc., que pueden estar localizados geográficamente en diferentes ubicaciones, de manera que trabajen conjuntamente como un solo sistema, para el soporte de tareas o aplicaciones intensivas en cálculos y datos, que de otra manera serían imposibles de realizar o tomarían demasiado tiempo empleando un sistema individual.

El continuo desarrollo de las técnicas de transmisión por fibra óptica, ha conducido a la definición de tres niveles en la escala evolutiva del *networking* óptico (ver Figura 1.4), en concreto redes de primera, segunda y tercera generación que se describen en las siguientes secciones, en las que la granularidad de tráfico se refiere tanto al volumen de tráfico como al tamaño de cada unidad de tráfico [82].

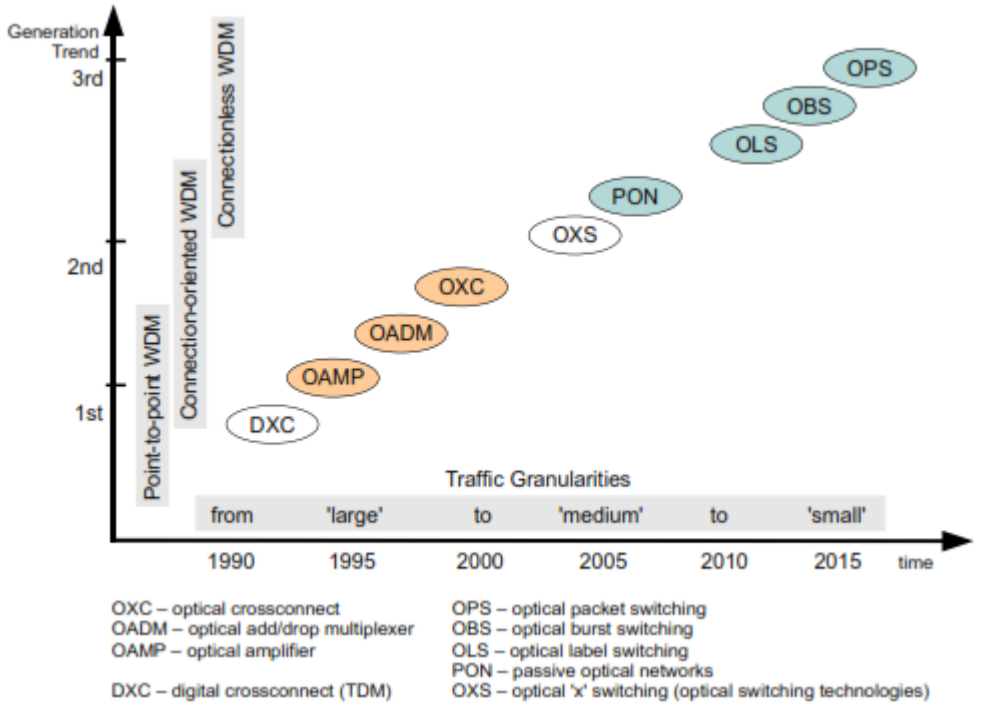


Figura 1.4 Evolución del networking óptico [27]

1.1.1 Redes ópticas de primera generación

La fase inicial del *networking* óptico, referida usualmente como redes ópticas de primera generación, inició su desarrollo en 1997 y enfocó su atención únicamente en utilizar la fibra óptica como medio de transmisión para incrementar la capacidad de la red, a través de la provisión de enlaces ópticos estáticos punto a punto para interconectar los distintos nodos de la red, donde todo el tráfico que arriba a un nodo es extraído y sometido a conversión opto-

electrónica para su procesamiento, y posteriormente convertido nuevamente al formato óptico (conversiones O/E/O) antes de ser enviado a través del puerto de salida correspondiente.

Inicialmente, cada una de las fibras ópticas de la red soportaba una única longitud de onda transportada a través de sistemas basados en la multiplexación por división de tiempo, TDM como SONET/SDH, ampliamente desplegada en los operadores de telecomunicaciones a nivel mundial y diseñada originalmente para el tráfico de voz; esta tecnología ha experimentado una migración hacia una solución mejorada de nueva generación (NG-SONET/SDH) capaz de transportar más eficientemente el tráfico de datos, pero con la aparición de la tecnología WDM la capacidad de transmisión de los enlaces aumentó considerablemente, permitiendo en esta generación de redes, la provisión de enlaces estáticos WDM punto a punto manteniendo la necesidad de conversiones O/E/O en cada nodo de la red para todo el tráfico, como se puede apreciar en la Figura 1.5. Los principales aspectos técnicos relacionados con esta implementación primitiva de WDM incluyen el diseño y desarrollo de lásers, y amplificadores ópticos (OAMP, *Optical Amplifier*), siendo estos últimos los elementos clave que permitieron la viabilidad comercial y el despliegue inicial de los sistemas WDM.

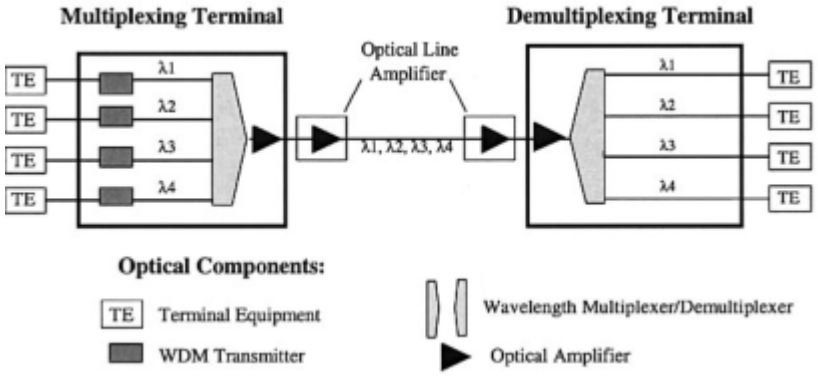


Figura 1.5 Sistema de transmisión WDM para n longitudes de onda [82]

La principal desventaja de esta generación de redes es el procesamiento electrónico, que conduce a una sobrecarga considerable e innecesaria, dado que aproximadamente el 85% del tráfico que atraviesa un nodo en una red es de tránsito o *bypass* [36], es decir, que no va destinado a dicho nodo, dando como resultado un desperdicio de recursos y componentes

electrónicos en los equipos, que incrementan los costos de la red no únicamente debido al gran número de transponders necesarios, sino también al complejo procesamiento de la señal que se debe realizar en los nodos intermedios; además constituye un factor que limita la escalabilidad de la red puesto que la velocidad de procesamiento en el dominio eléctrico no aumenta al mismo ritmo que la velocidad de transmisión en el dominio óptico.

En resumen, las redes ópticas de primera generación se caracterizan por utilizar la fibra óptica únicamente como medio de transmisión de alta capacidad, en consecuencia, todo el procesamiento, enrutamiento y conmutación se realiza en el dominio eléctrico. Con el fin de trasladar algunas de estas funciones al dominio óptico para superar las limitaciones derivadas del llamado cuello de botella electrónico y minimizar los costos de operación de la red, se introducen multiplexores de adición/extracción de señales, que permiten la extracción del tráfico destinado a cada nodo, dejando pasar el resto intacto, evitando así el procesamiento innecesario de todo el tráfico en los nodos de la red; dando lugar a la definición de las redes ópticas de segunda generación conocidas también como redes de enrutamiento por longitud de onda o redes de conmutación de circuitos ópticos descritas a continuación.

1.1.2 Redes ópticas de segunda generación

Las redes ópticas de segunda generación consisten en redes interconectadas en anillo o malla con cada nodo equipado con multiplexores ópticos de adición/extracción, OADM (*Optical Add/Drop Multiplexer*) conocidos también como multiplexores de adición/extracción de longitudes de onda, WADM (*Wavelength Add/Drop Multiplexer*), donde se pueden añadir y/o extraer selectivamente canales ópticos y a la vez dejar pasar el tráfico de *bypass*. En esta generación de redes, se pretende realizar funciones adicionales a sólo las de transmisión punto a punto en el dominio óptico. En concreto, el enrutamiento y conmutación dentro del dominio óptico son funciones que como se mencionó en la subsección anterior pueden conducir a un ahorro considerable de equipos electrónicos dentro de los nodos de las redes que emplean transmisión por fibra óptica, que aparte de elevar su costo conllevan a la implementación de equipamiento que requiere mayor espacio físico y consumo de energía. Por estas razones, y

teniendo en cuenta la premisa de que la mayor parte de tráfico que pasa por un nodo es del tipo *bypass*, si este se pudiera mantener en el dominio óptico a medida que atraviesa el nodo en lugar de incurrir en sucesivas conversiones OEO para su procesamiento y retransmisión, se reducirían sustancialmente los costos asociados en proporcionar la conmutación y enrutamiento de alta capacidad en cada nodo. Además, el traslado de otra serie de funciones al dominio óptico relacionadas con el control, la gestión y la protección de la red pueden también aportar notables ventajas [36].

Para poder llevar a cabo en el dominio óptico estas funciones y otras más que pudieran requerirse en el futuro, es necesario establecer una nueva capa dentro del modelo de capas de la red, denominada capa óptica, que ocupa el nivel más bajo en la jerarquía de la red, y tiene como función principal proporcionar una serie de servicios a las capas situadas por encima de ella denominadas capas clientes tales como SDH, ATM e IP, a través del establecimiento de una serie de circuitos o caminos ópticos, denominados *lightpaths*[4], orientados a la conexión extremo a extremo en la capa óptica, operados y administrados en base a una topología virtual sobre la topología física, que pueden ser configurados de forma manual o dinámica, en este último caso en respuesta a cambios de tráfico. En los nodos intermedios, los caminos ópticos son enrutados y/o conmutados entre diferentes enlaces de salida pudiendo mantenerse la misma longitud de onda a través de todo trayecto (característica conocida como restricción de longitud de onda), o cambiarse la longitud de onda si la red incluye capacidades de conversión de longitud de onda. Las longitudes de onda se pueden reutilizar espacialmente en diferentes enlaces, conduciendo a una mejor utilización del ancho de banda. Así, un enlace físico de fibra puede transportar varios caminos ópticos a la vez, siempre y cuando no empleen la misma longitud de onda, o dicho de otra forma, distintos caminos ópticos pueden emplear la misma longitud de onda siempre que no compartan enlaces físicos comunes [36].

La introducción de la capa óptica implica una modificación en el modelo de capas de una red de telecomunicaciones que viene recogida en la recomendación G.872 sobre la arquitectura de la red de transporte de la Unión Internacional de Telecomunicaciones (ITU, *International*

[4] Un *lightpath* se define como un canal óptico punto a punto donde la transmisión entre los nodos extremos e intermedios se realiza empleando una portadora óptica con una frecuencia específica (longitud de onda).

Telecommunication Union). La capa óptica en las redes de segunda generación está formada principalmente por dispositivos fotónicos y se estructura en tres subcapas como se muestra en la Figura 1.6.

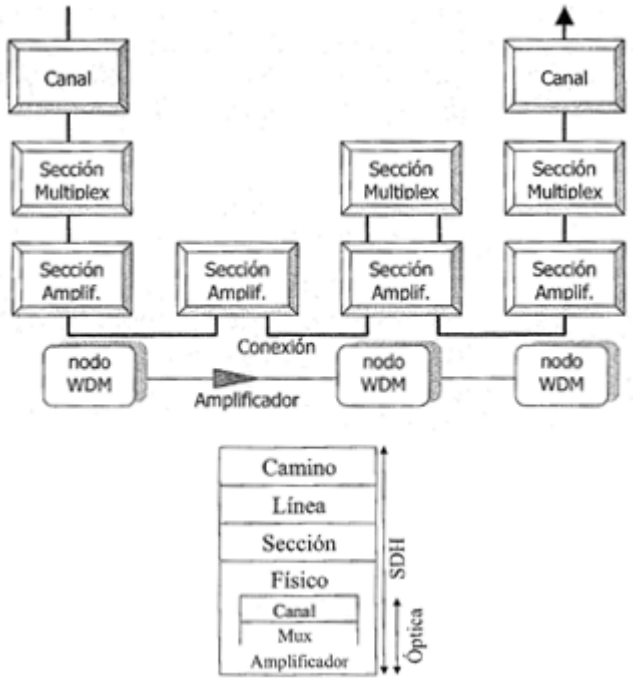


Figura 1.6 Arquitectura de la capa óptica definida en la Rec. G.872 de la ITU (parte superior). Ejemplo de anidamiento de capas cuando SDH es cliente de la capa óptica (parte inferior) [36]

La subcapa de canal se encarga de la transmisión extremo a extremo del camino óptico, la de multiplexación se ocupa de definir las secciones de multiplexación constituidas por los enlaces que son compartidos por varios caminos ópticos multiplexados en longitud de onda. Finalmente, la subcapa de amplificador incluye las secciones de amplificación constituidas por las partes de los enlaces WDM comprendidas entre dos amplificadores ópticos [36]. En la Figura 1.7 se muestra un esquema sencillo de una red de segunda generación donde se pueden observar sus elementos principales y varios caminos ópticos establecidos entre usuarios o clientes finales.

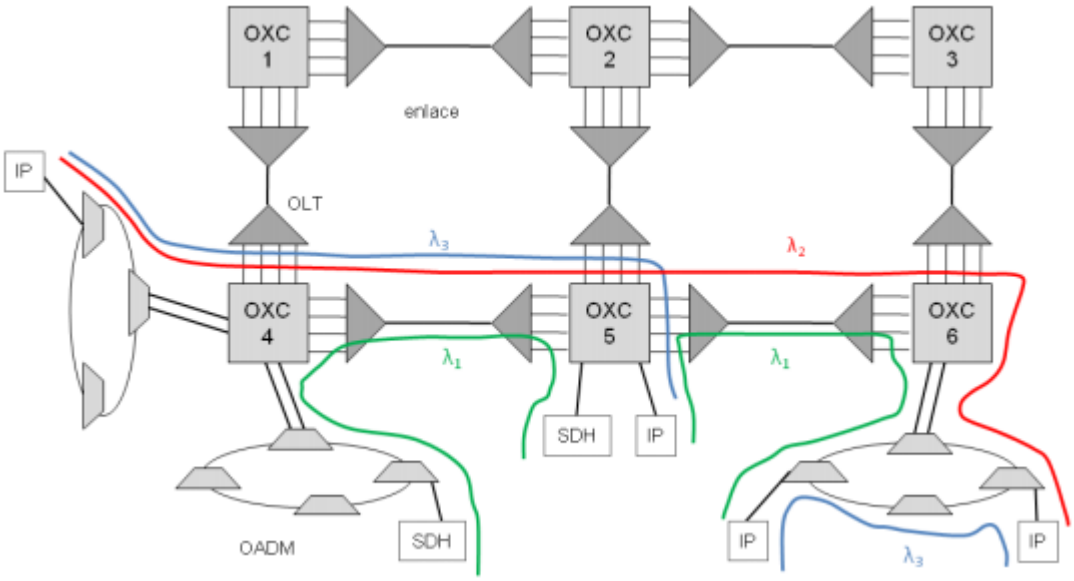


Figura 1.7 Esquema sencillo de una red óptica de segunda generación [36]

Las tecnologías clave que condujeron a esta evolución de redes que hizo posible la conmutación de datos en el dominio completamente óptico, fueron el desarrollo de los multiplexores ópticos de adición/extracción (OADM, *Optical Add-Drop Multiplexer*), y las matrices de conmutación óptica, OXC (*Optical Cross-Connect*) descritos brevemente a continuación.

Multiplexor óptico de adición/extracción (OADM).- Es un dispositivo que toma una señal compuesta WDM, que consiste de múltiples longitudes de onda, desde un puerto de entrada y puede extraer selectivamente una o varias longitudes de onda hacia sus puertos locales dejando pasar el resto del tráfico hacia el puerto de salida. Al mismo tiempo, puede insertar selectivamente longitudes de onda, antes de que la señal compuesta pueda dejar el puerto de salida correspondiente.

Matriz de conmutación óptica (OXC).- Es un dispositivo con múltiples puertos de entrada y salida. En adición a la capacidad de adición/extracción, también puede conmutar una longitud de onda desde cualquier puerto de entrada a cualquier puerto de salida. La conmutación de señales ópticas a través de un dispositivo completamente óptico (O/O/O) corresponde al

segundo enfoque[5] de implementación de un OXC. Tal conmutador se denomina usualmente OXC transparente o matriz de conmutación fotónica, PXC (*Photonic Cross-Connect*).

Tanto los OADMs como OXCs pueden incorporar o no capacidades de conversión de longitud de onda, de modo que el tráfico de una longitud de onda se puede cambiar a otra longitud de onda en un puerto de salida. Además de la conversión de longitud de onda, otros aspectos técnicos de esta segunda generación de redes WDM incluyen: el problema de enrutamiento y asignación de longitud de onda, RWA (*Routing and Wavelength Assignment*), interoperabilidad entre redes WDM, control y administración de la red, etc. [61].

El término OADM hace referencia a un multiplexor con capacidad de conexión únicamente en dos direcciones (o grados de libertad), que corresponde a su desarrollo inicial y es apropiado para topologías en anillo. Sin embargo, para topologías en malla que requieren más de dos grados de libertad para sus múltiples conexiones y el concepto de dotar a la red de flexibilidad e inteligencia en la capa óptica, el desarrollo de estos dispositivos condujo a los multiplexores de adición/extracción reconfigurables, ROADM (*Reconfigurable Optical Add/Drop Multiplexer*) en particular los basados en el conmutador selectivo de longitud de onda, WSS (*Wavelength Selective Switch*), utilizados hoy en día en equipos comerciales que permiten ocho grados de libertad, y que conjuntamente con la tecnología OTN (*Optical Transport Network*) definida principalmente en la recomendación ITU-T G.709 (implementación de la capa de canal óptico de acuerdo a los requerimientos de la Rec. ITU-T G.872 [113]), proveen una solución integral que combina la flexibilidad de SONET/SDH con la transmisión de alta capacidad ofrecida por DWDM.

La Rec. ITU-T G.709 mejora el desempeño de la red de transporte y facilita la evolución a mayores anchos de banda en el *backbone*. OTN G.709 se define como una envoltura digital que encapsula diversas tramas de datos desde diferentes fuentes en una sola entidad independiente de su protocolo nativo, que incluye el transporte de cabeceras que proveen capacidades de operación, administración y mantenimiento (OA&M, *Operation, Administration & Management*) y corrección de errores mediante FEC (*Forward Error Correction*), formando una jerarquía

[5] El primer enfoque es implementado en el dominio eléctrico y se denomina OXC electrónico u OXC opaco.

multiplexada de unidades de datos ópticos, para tasas de bits desde 1 Gbps hasta 100 Gbps como se puede observar en las Figuras 1.8 y 1.9.

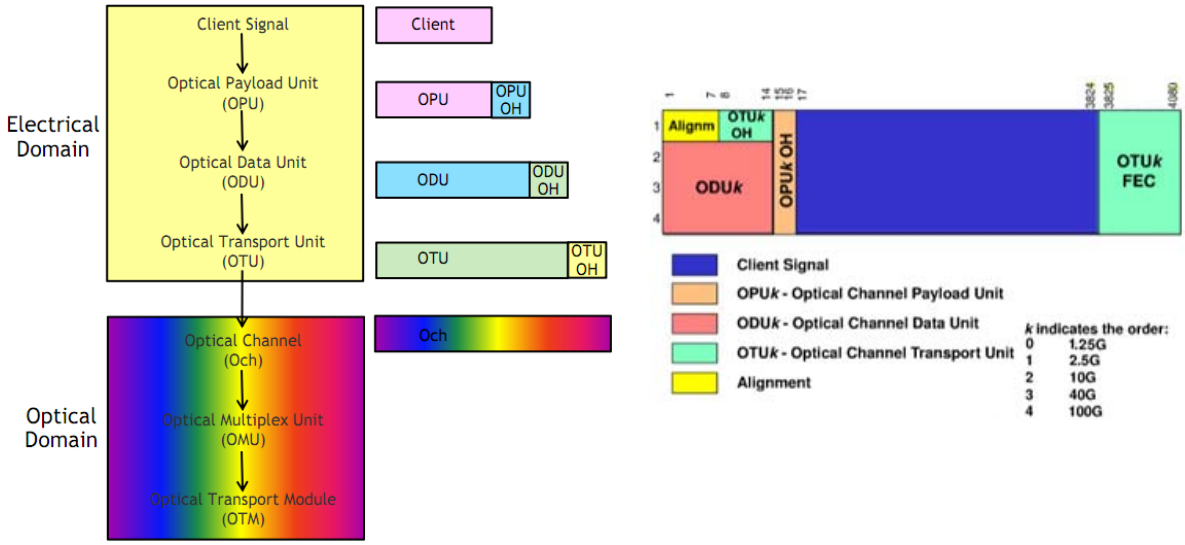


Figura 1.8 Encapsulamiento de un contenedor OTN y formato de la trama OTN G.709 [108]

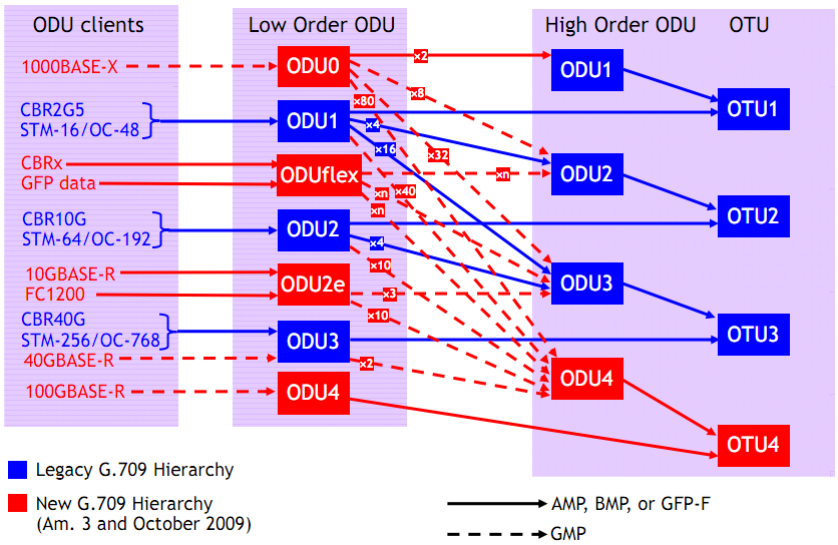


Figura 1.9 Jerárquica OTN [108]

No obstante, en las redes OTN la asignación y liberación de conexiones se realiza manualmente de forma centralizada desde un sistema de gestión de red, NMS (*Network Management System*), dando como resultado una provisión de servicios propensa a errores, largos

tiempos de aprovisionamiento, utilización ineficiente de recursos, además de falta de flexibilidad del sistema para reaccionar de forma dinámica a los cambios de tráfico en la red. Esto, sumado a la necesidad cada vez mayor de proveer una infraestructura de red que permita superar las limitaciones antes mencionadas, así como también otros inconvenientes como: difícil interoperabilidad entre redes de paquetes de clientes y redes ópticas conmutadas por circuitos, compleja administración de la red, difícil interoperabilidad entre redes que pertenecen a diferentes operadores y falta de protección en redes ópticas tipo malla, permitió la introducción de inteligencia en la red por medio de un plano de control distribuido, reconocido como la mejora adicional para satisfacer los requerimientos de aprovisionamiento rápido y flexible de ancho de banda, descubrimiento automático de la topología y rápida restauración del tráfico. Estas redes emergentes, denominadas Redes Dinámicas de Conmutación Óptica de Circuitos (DOCS, *Dynamic Optical Circuit Switching*) conocidas también como redes dinámicas de conmutación de longitud de onda, son un paso adelante en la evolución hacia redes más inteligentes y flexibles [106].

El proceso de estandarización para dicho plano de control ha sido promulgado por dos organismos diferentes. Por un lado, la ITU-T desarrolló el paradigma denominado ASON (*Automatically Switched Optical Networks*), mientras que la IETF desarrolló una extensión generalizada de MPLS denominada GMPLS (*Generalized Multiprotocol Label Switching*). En principio, ASON y GMPLS no deberían ser competidores sino más bien trabajar de forma complementaria [5].

ASON se define como una red de transporte óptico que tiene la capacidad de conexión dinámica, que se logra a través de un plano de control que realiza las funciones de control de llamada y conexión. Se describe además, como una arquitectura de referencia debido a que presenta componentes funcionales e interfaces abstractas. En la Figura 1.10 se presenta una vista lógica de la arquitectura ASON, donde se pueden diferenciar claramente tres planos de trabajo. El plano de transporte, referido también como plano de datos, representa los recursos funcionales de la red que transmiten la información del usuario entre las distintas localidades. El plano de control realiza las funciones de control de llamada y control de conexión, que son automatizadas, en base a inteligencia en la red, que incluye el descubrimiento automático, enrutamiento y señalización. El plano de gestión realiza las funciones de administración para

el plano de transporte, plano de control, y el sistema en general, así como también coordina la operación de todos los planos [5].

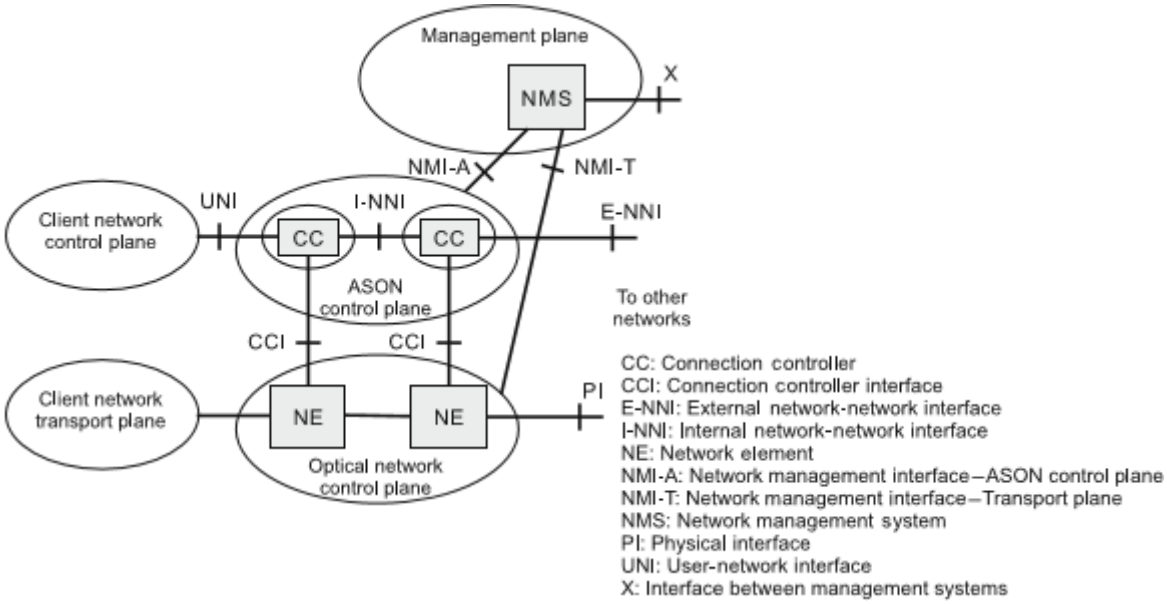


Figura 1.10 Vista lógica de la arquitectura ASON [5]

Por otro lado, GMPLS surge como una evolución de la arquitectura de ingeniería de tráfico de MPLS, con el fin de utilizar una arquitectura integral que permita la provisión de circuitos extremo a extremo a través de enrutadores, multiplexores SDH, y conmutadores ópticos, así como también funciones de protección, restauración, monitoreo, mantenimiento, etc., incluyendo todas las capas y dispositivos de conmutación involucrados en una conexión. Por lo tanto, GMPLS extiende el concepto de LSP (*Label Switched Path*) a circuitos en redes TDM y ópticas, y propone la extensión de los protocolos de control existentes de MPLS para controlar los conmutadores orientados a la conexión. Esto significa que los conmutadores TDM y ópticos deben tener conocimiento de los protocolos de control de GMPLS para traducir la operación de una entrada de reenvío de etiquetas en un cambio de estado de conmutación en los dominios de tiempo, longitud de onda y espacio. De esta manera, el plano de control de GMPLS es capaz de operar sobre un amplio rango de dispositivos de red heterogéneos (enrutadores IP/MPLS, elementos de red SONET/SDH, conmutadores ATM, así como

también elementos de red ópticos como OXCs y OADMs), que representan diferentes tipos de LSR que realizan distintos tipos de conmutación y que se pueden categorizar de acuerdo a la Capacidad de Conmutación de la Interfaz (ISC, *Interface Switching Capability*), como se muestra en la Figura 1.11.

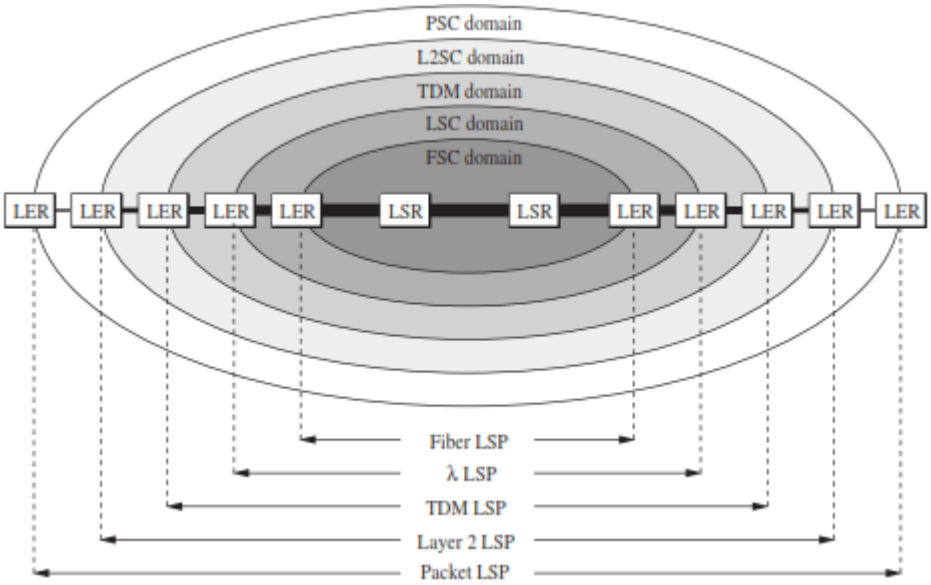


Figura 1.11 Túneles LSP (Label Switched Paths) de GMPLS [75]

Las redes DWDM de primera y segunda generación han sido desplegadas en varios operadores de telecomunicaciones, inicialmente sólo en la infraestructura de *long-haul* (larga distancia) y después se han ido acercando a los *edges* (extremos) de la red. La razón económica detrás de este movimiento es que a medida que uno se mueve más cerca del *edge*, el costo de una red en un nivel de jerarquía particular se amortiza en pocos usuarios finales mientras que aumenta el precio. Debido a esta diferencia en la sensibilidad del precio, existe usualmente una tendencia a desplegar nuevas tecnologías primero en el *backbone*, para luego extenderse gradualmente hacia el *edge* de la red conforme la tecnología madura y se pueden alcanzar precios más bajos. El desarrollo de WDM presentado en la Figura 1.12 es un ejemplo claro de esta tendencia [27].

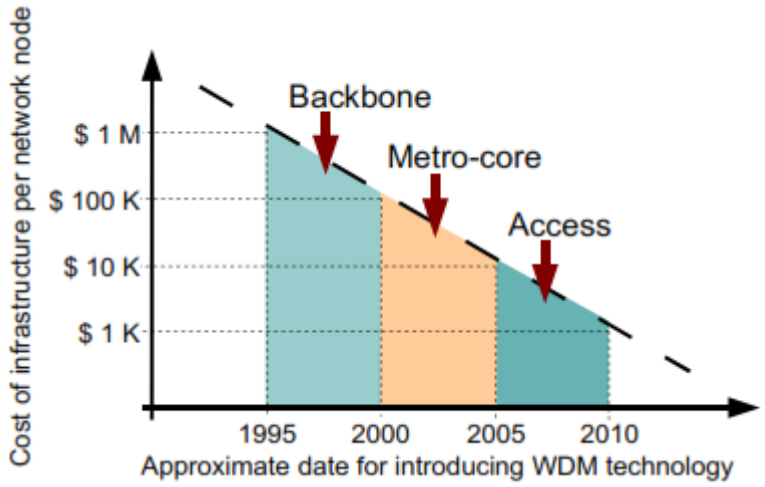


Figura 1.12 El efecto de la reducción de costos en infraestructura WDM [27]

1.1.3 Redes ópticas de tercera generación

Las redes ópticas de tercera generación se plantean como arquitecturas basadas también en OADM y OXC, pero con capacidad de soportar el *networking* óptico no orientado a la conexión así como también reconfigurabilidad, mientras proveen altos niveles de multiplexación estadística en el dominio óptico y una mayor granularidad. Dentro de este contexto, se proponen dos paradigmas denominados conmutación óptica de paquetes (OPS, *Optical Packet Switching*) y conmutación óptica de ráfagas (OBS, *Optical Burst Switching*) que serán descritos con mayor detalle más adelante.

Se visualiza que esta generación de redes ópticas tendría dos componentes funcionales: una red de nodos de conmutación óptica que soportan la transmisión de datos completamente óptica y varias redes de acceso que alimentan los datos a la red de *core* a través de los nodos de borde (ingreso y egreso). Estas redes de acceso podrían estar basadas en *Gigabit Ethernet*, redes ópticas pasivas PON (*Passive Optical Networks*), o cualquier otra tecnología de acceso de banda ancha que es la responsable de transportar el tráfico de los usuarios finales. Los nodos de ingreso que conectan las redes de acceso a la red de *core* son enrutadores de alta velocidad que pueden agregar el tráfico destinado al mismo nodo de egreso. La red de *core* se podría componer de una malla de fibras con múltiples longitudes de onda con elementos

reconfigurables (por ejemplo, OXCs) interconectados por medio de enlaces ópticos de larga distancia con alta capacidad. En la capa de acceso se han desarrollado tecnologías de banda ancha que permiten hoy en día brindar al usuario una gama de servicios integrados, comercializados como servicios *tripleplay* (telefonía fija, televisión e Internet) y en algunos países ahora también el servicio *cuadruplay* (que incluye la telefonía celular). El campo de las redes de acceso se encuentra fuera del alcance de este proyecto de tesis; sin embargo, debido a que son el origen del tráfico que transporta el *backbone*, se considera conveniente mencionar brevemente en esta subsección, las diferentes tecnologías involucradas, que se espera aporten en la próxima década, una demanda importante de ancho de banda, derivada principalmente por aplicaciones de video, que podrían seriamente desafiar la viabilidad de las arquitecturas de red convencionales. El ancho de banda disponible para los usuarios en las redes de acceso ha crecido exponencialmente en los últimos 20 años en un ciclo de auto-alimentación donde el apetito insaciable de nuevos servicios ha estimulado la necesidad de nuevas tecnologías innovadoras de mayor velocidad, y su posterior disponibilidad ha permitido la creación de nuevos servicios imaginativos [47]. Para soportar este crecimiento, las tecnologías de datos subyacentes han evolucionado desde módems de acceso telefónico a redes integradas de servicios digitales, ISDN (*Integrated Services Digital Network*), luego a generaciones sucesivas de líneas de abonado digital xDSL (*x Digital Subscriber Line*), y más recientemente, a fibras directas hacia el hogar, FTTx (*Fiber To The x*). Al mismo tiempo, los operadores de telecomunicaciones tradicionales que proporcionaban servicios de voz y datos a niveles residencial y corporativo a través de sus redes, están experimentado un importante movimiento a una infraestructura convergente "todo IP".

Históricamente, ha existido un incremento de 10 veces en el ancho de banda cada 6 años como se ilustra en la Figura 1.13, y hay evidencia de que esta tendencia continuará en el futuro [47]. En el año 2006, se ofrecían exitosos paquetes comerciales de servicios *triple-play* con un ancho de banda en el rango de 20 Mbps a 30 Mbps por usuario[6], por lo que se podía prever un conjunto de servicios que demanden más de dos o tres veces ese ancho de banda en el futuro

[6] El paquete estándar de video, voz y datos consistía de 1 señal de video IP HDTV (8 Mbps, MPEG-4), 1 señal de video IP SDTV (2 Mbps, MPEG-4), 1 línea telefónica (64 kbps), e Internet de alta velocidad (5 Mbps). Consumo de ancho de banda Total - 15 Mbps [80].

próximo[7]. Más recientemente, hoy en día, los operadores de telecomunicaciones de muchos países han lanzado ofertas residenciales de *triple-play* por fibra óptica con anchos de banda de algunas centenas de Mbps.

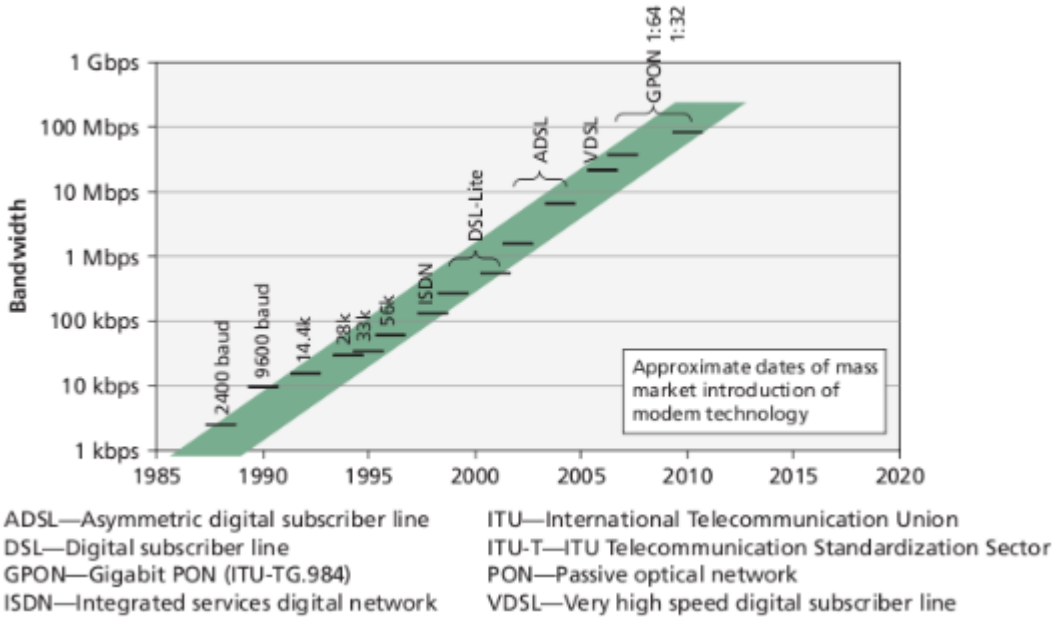


Figura 1.13 Crecimiento del ancho de banda para redes de acceso [27]

Aunque las tecnologías actuales basadas en xDSL son capaces de cumplir con estos requerimientos de ancho de banda, la tecnología FTTx se apunta como la única infraestructura de acceso de banda ancha que permitirá soportar mayores capacidades en el futuro, donde la situación puede ser muy diferente, sin duda habrán nuevos servicios y aplicaciones que demanden aún más ancho de banda, en particular las basadas en video, que podrían incluir [47]: 1) la transmisión de un video completo de televisión en alta definición (HDTV) como un archivo; 2) el desarrollo de televisores de pantallas grandes de más de 100 pulgadas por fabricantes de electrónica (como LG, Philips, Panasonic, Samsung, etc.) que impulsan la

7 En ese entonces se preveía que en los próximos 5 años, el paquete estándar de video, voz y datos probablemente podría consistir de 1 señal de video IP Super HDTV (32 Mbps, MPEG-4), 1 señal de video IP HDTV (8 Mbps, MPEG-4), línea de teléfono (64 kbps), e Internet de alta velocidad (20 Mbps). Consumo de ancho de banda Total - 60 Mbps [80].

innovación de servicios y un mayor uso del ancho de banda, con un posible formato super HD de 2160i (8 megapíxeles) que requiere de 32 Mbps a 60 Mbps por flujo de video y un formato de ultra alta definición de 4320i (32 megapíxeles) que puede precisar tanto como 256 Mbps hasta 480 Mbps por flujo; 3) la entrega de múltiples imágenes de video que ofrecen distintos ángulos de visión de eventos deportivos o un ángulo de visión más amplio (180 o 360 grados) análogo a la experiencia Imax; 4) video en 3D combinado con alta definición con un consumo de ancho de banda de 63 Mbps para televisión de definición estándar (SDTV), 187 Mbps para HDTV, 643 Mbps para super DTV, y 2.571 Mbps para ultra DTV; 5) experiencias de video en 3D interactivas podrían contemplarse como juegos en 3D, conferencias en 3D, viajes virtuales en 3D, *e-learning* en 3D, experiencias participativas de actuación en 3D en las que el espectador llega a ser un actor con otros en el espacio virtual.

Aunque estos conceptos de servicio son de hecho especulativos y su momento es incierto, no hay duda de que la competencia e innovaciones tecnológicas producirán servicios y aplicaciones imaginativas que explotarán el ancho de banda adicional disponible [47]. Por lo tanto, el potencial de crecimiento desde una perspectiva impulsada por servicios es masiva, posiblemente dos o tres órdenes de magnitud mayor que el ancho de banda de las redes de banda ancha actuales. Es así que, simplemente escalar la tecnología de las redes actuales no producirá soluciones viables y los operadores tendrán que considerar cuidadosamente nuevos enfoques arquitectónicos para la implementación de redes, donde el rol del *networking* óptico será crucial para satisfacer estas demandas sin precedentes en el futuro [27].

En base a lo expuesto anteriormente, en la Figura 1.14 se ilustran las tecnologías de redes ópticas actuales y posibles etapas adicionales en la evolución del *networking* óptico en los próximos años [61]. En dicha figura, (a) representa el paso a un modelo de conmutación más centrado en los datos y dinámico utilizando una arquitectura ASON que permitiría el aprovisionamiento automático de caminos ópticos y soporte de NG-SDH/SONET con cross-conectores digitales, DXC (*Digital Cross Connect*) o conmutación OXC (O/E/O) en el plano de datos. La Figura 1.14 (a) muestra también que se prevé el paso a una arquitectura IP/MPLS (p.ej. enrutadores IP) o GMPLS (con conmutadores de longitud de onda O/E/O u O/O/O), que proporciona una capacidad dinámica mejorada. GMPLS permite que todos los modos de transporte, circuitos, ráfagas y paquetes sean soportados y puede ser desplegado de una

manera centralizada o distribuida. Esto representa una de las opciones para construir una "red convergente", donde el *backbone* es una red multi-tecnología IP/GMPLS/OEO/OOO que soporta todos los servicios (voz, datos, video), que puede superar a la arquitectura ASON. También es el caso que en los últimos años *Carrier Ethernet* (basado en *Ethernet* nativo o MPLS) parece cada vez más atractivo en todas las capas de la red, y de hecho algunas empresas de telecomunicaciones ya operan redes nacionales convergentes con elementos de conmutación *Ethernet*. El movimiento hacia estándares de 100G ilustra la importancia de esta tecnología y pistas de futuros roles importantes en las redes de próxima generación. La Figura 1.14 (b) representa un movimiento hacia un diseño centrado en el usuario, basado en OBS con GMPLS (OBS/GMPLS); esta tecnología proporciona granularidad a nivel de sub-longitud de onda y también es de interés para las futuras redes ópticas para servicios *Grid* [102]. Finalmente, la Figura 1.14 (c) representa el paso a una red OPS donde MPLS provee un plano de control común (a través de dominios eléctricos/ópticos). OPS ofrece la granularidad más fina y es vista todavía como la técnica de conmutación final, pero su éxito dependerá de muchos desafíos tecnológicos, que se describen en la subsección 1.3.2.

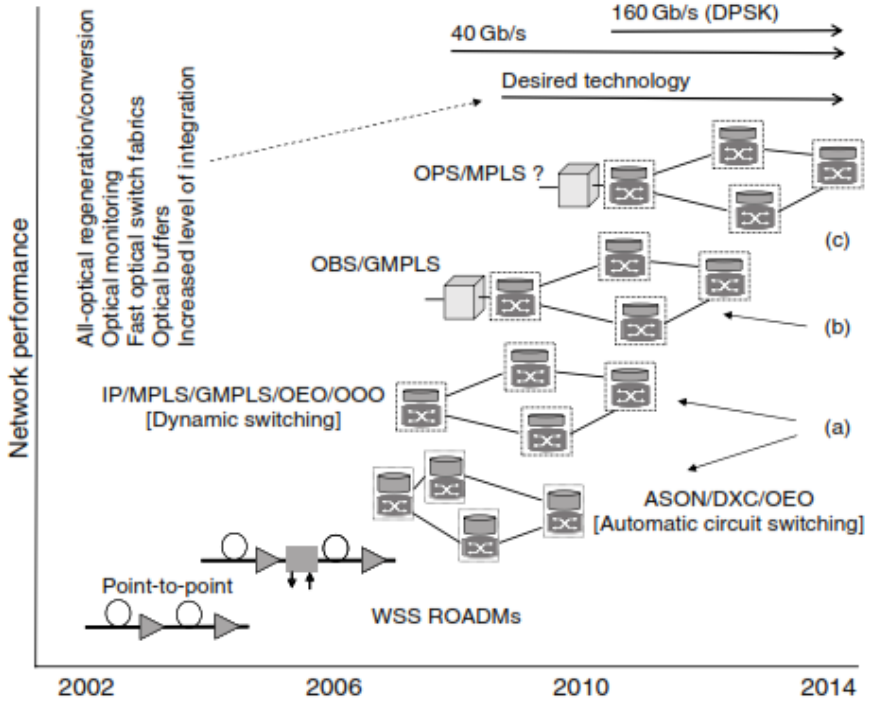


Figura 1.14 Tecnologías de red actuales y posibles etapas de evolución de las redes ópticas [61]

Como se expuso en las secciones anteriores, las redes ópticas han evolucionado desde el *networking* tradicional compuesto por enlaces WDM estáticos punto a punto a un *networking* óptico dinámicamente reconfigurable. Su continua evolución es probable que conduzca a un *networking* con capacidad de conmutación dinámica de ráfagas y paquetes, incluyendo elementos de red ópticos reconfigurables (ROADMs y OXCs) y enrutadores ópticos con un plano de control distribuido. De forma similar a esta evolución presentada comparativamente en la Figura 1.15, se ilustra también la evolución en cuanto a la tendencia de las pilas de protocolos del *networking* óptico en la Figura 1.16.

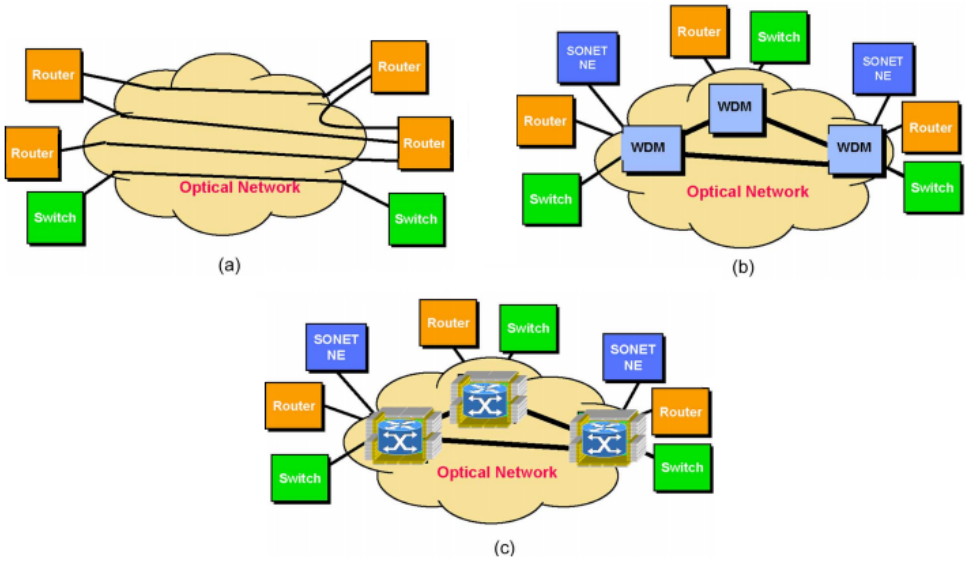


Figura 1.15 Evolución del internetworking óptico desde a) redes ópticas de primera generación WDM punto a punto, b) redes ópticas de segunda generación OCS, y c) redes ópticas de tercera generación OPS y OBS conectados con otros elementos de red [61]

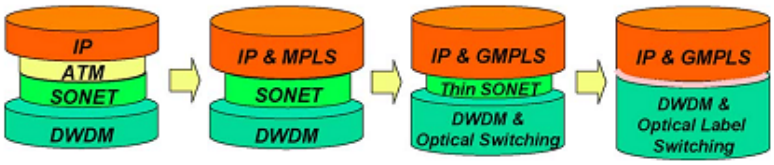


Figura 1.16 Evolución de la tendencia de las pilas de protocolos desde el tradicional IP/ATM/SONET/DWDM al nuevo paradigma IP sobre WDM con conmutación óptica de etiquetas (OLS) [61]

1.2 Recientes desarrollos y desafíos técnicos en sistemas de transmisión de alta capacidad [4][17][62][100][135,137][145]

La historia muestra que los proveedores de servicios de red han hecho un buen uso de cada etapa de una nueva capacidad de canal provista por los fabricantes de equipos. La Figura 1.17 presenta la línea de tiempo de los incrementos en la capacidad de los enlaces de fibra en las redes de los proveedores de servicios. A inicios de 1990, una capacidad de unos cuantos cientos de Mbps por enlace y sólo un canal por cada hilo de fibra en una red de transporte era típico. A medida que el correo electrónico llegó a ser una nueva herramienta de comunicación a mediados de 1990, la capacidad de la fibra incrementó gradualmente a unos pocos Gbps, y este crecimiento continuó al direccionar la demanda que las personas necesitaban para iniciar el acceso a Internet. A finales de 1990 la capacidad de la fibra continuó el crecimiento con el despliegue de canales 10 Gbps y técnicas WDM para multiplexar y amplificar un número pequeño de longitudes de onda (4-8) en un par de fibras.

A inicios del año 2000, el uso del Internet había llegado a ser algo habitual, pero el *networking* se mantenía al ritmo de la introducción de las técnicas de DWDM que podrían soportar cuarenta, ochenta, o incluso más longitudes de onda permitiendo a las capacidades de fibra estar cerca de los Tbps. Este extensivo incremento de la capacidad de la fibra ayudó a la red de transporte a soportar las demandas de usuarios continuamente crecientes. A finales del 2000, la introducción de canales 40G dio a la capacidad de las redes un nuevo impulso. Para el año 2010, la compartición de video en el Internet mediante aplicaciones como YouTube y otros servicios de video bajo demanda, VoD (*Video on Demand*) nuevamente empezaron a desafiar la capacidad de las redes existentes. La introducción de los canales 100 Gbps coherentes desarrollados recientemente ha proporcionado otro incremento de 10 veces en la capacidad de aproximadamente 10 Tbps por fibra. Esto debería direccionar a corto plazo los requerimientos de capacidad, pero avanzando hacia, el *cloud computing* y otras aplicaciones hambrientas de ancho de banda que continuarán consumiendo los recursos de la red, y nuevas técnicas ópticas para incrementar la capacidad del canal y la capacidad del enlace óptico serán introducidas progresivamente.

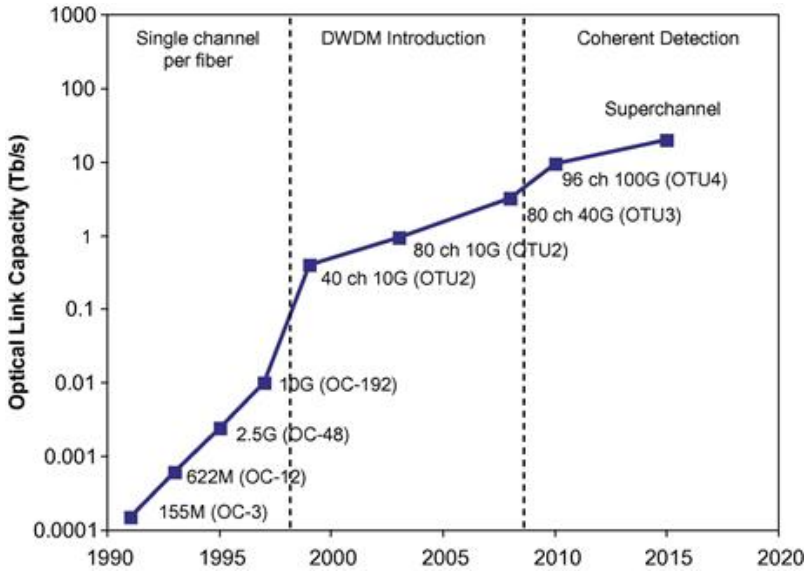


Figura 1.17 Capacidades de canal típicas y capacidades de fibra de las redes de transporte, cuando empezaron a ser populares [62]

Hoy en día las redes ópticas son redes de transporte realmente flexibles con enlaces de transmisión que son capaces de operar a capacidades de Terabits por segundo combinando muchas longitudes de onda, cada una operando actualmente a tasas tan altas como 100 Gbps y más, en una sola fibra. La evolución de los sistemas de transmisión óptica de alta capacidad ha sido posible gracias a innovaciones tecnológicas en varias áreas clave. Un área notable de avance significativo en los últimos años es la correspondiente a los formatos de modulación y esquemas de detección avanzados que permiten un alto rendimiento y transmisión óptica con alta eficiencia espectral.

La elección del formato de modulación y esquema de detección tiene fuertes implicaciones sobre el sistema tales como el requerimiento de OSNR (*Optical Signal to Noise Ratio*), eficiencia espectral alcanzada, tolerancia a la dispersión cromática (CD, *Chromatic Dispersion*) y dispersión por modo de polarización (PMD, *Polarization Mode Dispersion*), y la no linealidad de la fibra [135]. Los sistemas de transmisión óptica DWDM convencionales, se basan principalmente en canales de 10 Gbps con modulación de encendido-apagado OOK (*On Off Keying*) sobre una grilla de canales de 50 GHz con una eficiencia espectral de 0.2 b/s/Hz. Sin embargo, en la transmisión de velocidades superiores a 10 Gbps, los efectos de distorsión de la fibra óptica

como la dispersión cromática (CD), la dispersión por modo de polarización (PMD), el ruido y otros efectos no lineales se vuelven más severos, impactando fuertemente en el desempeño de la transmisión y afectando drásticamente la calidad de la señal. Por ejemplo, si para una comunicación a 40 Gbps se utilizara el mismo formato de modulación usado para 10 Gbps, se reduciría la eficiencia espectral y por ende se podrían transmitir menos longitudes de onda sobre un par de hilos de fibra óptica.

La mejora de capacidad y escalabilidad de los sistemas actuales requiere que canales de longitudes de onda de 40 Gbps y 100 Gbps, que fueron estandarizados en junio de 2010, sean transportados sobre la misma infraestructura de red existente, para lo cual es necesario abordar varios desafíos técnicos. En primer lugar, el ancho de banda espectral óptico de cada canal de 40 Gbps o 100 Gbps tiene que ser similar al del canal de 10 Gbps OOK para ser transportado sobre el mismo sistema DWDM, lo cual implica la necesidad de formatos de modulación espectralmente más eficientes para los canales con tasas de transmisión más altas. En segundo lugar, se prefiere que la distancia de transmisión de los canales de 40 Gbps y 100 Gbps sea comparable a la de los canales de 10 Gbps actuales, lo que significa que formatos de modulación y esquemas de detección avanzados son requeridos para que las tolerancias de la señal al ruido ASE (*Amplified Spontaneous Emission*), CD, PMD, y no linealidad de la fibra no se vean comprometidas con el incremento de la velocidad de datos por canal [135]. La Figura 1.18 muestra los diagramas de constelación de los cuatro formatos de modulación de alta eficiencia espectral basados en DPSK (*Differential Phase Shift Keying*) que han sido demostrados experimentalmente, todos compatibles con detección directa.

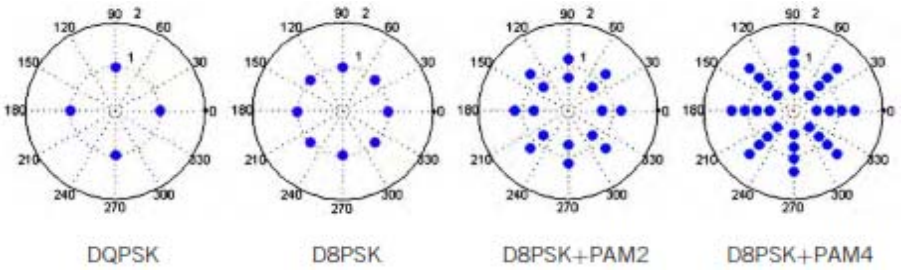


Figura 1.18 Diagramas de constelación de cuatro formatos de modulación basados en DPSK de alta eficiencia espectral demostrados con detección directa [135]

La introducción de canales WDM a mayor velocidad de datos ha proporcionado un gran crecimiento en la capacidad de los sistemas de transmisión, y dado que las velocidades del canal están creciendo por encima de 40 Gbps o 100 Gbps y más, uno de los desafíos clave como se mencionó anteriormente es incrementar la eficiencia espectral de los sistemas de comunicación por fibra óptica que constituye una manera efectiva de satisfacer la demanda siempre creciente para la capacidad de transmisión y explotar completamente el potencial de la actual infraestructura de fibra desplegada. En este punto, la multiplexación por división de polarización (PDM, *Polarization Division Multiplexing*) es un medio eficiente para duplicar la eficiencia espectral de un formato de modulación dado, que transmite dos canales con estados ortogonales de polarización en la misma longitud de onda. Con el uso de un receptor coherente digital con diversidad de polarización, PDM puede ser soportado sin complejidad adicional en el hardware del receptor, por lo tanto la detección digital coherente es naturalmente compatible con PDM. De hecho, la mayor parte de demostraciones recientes con detección digital coherente fueron realizadas utilizando PDM [135].

Por lo antes mencionado, la transición en la tasa de bit del canal a 100 Gbps y más, requiere que el esquema de modulación convencional OOK y el esquema de detección directa sean abandonados en favor de formatos de modulación más avanzados, tales como la modulación por desplazamiento de fase en cuadratura con multiplexación por polarización, PM-QPSK (*Polarization Multiplexed Quadrature Phase Shift Keying*) y esquemas de detección coherente. En contraste con la tecnología existente de detección directa, un esquema de detección coherente puede detectar no sólo la amplitud de la señal óptica sino además la fase y la polarización. Con un sistema de detección coherente se incrementa la capacidad de detección y la eficiencia espectral, lo cual se traduce en que mayor información puede ser transmitida con el mismo ancho de banda óptico. Además, como la detección coherente permite detectar la fase y la polarización de la señal óptica y luego ser medida y procesada, los impedimentos en la transmisión que anteriormente presentaban retos, pueden, en teoría, ser mitigados electrónicamente. A pesar de los múltiples desafíos (respecto a frecuencias, ruidos, amplificación, etc.), un sistema de detección coherente tiene muchas ventajas sobre las tecnologías de detección tradicionales, entre las cuales se pueden mencionar las siguientes [4]:

- Un incremento de la sensibilidad del receptor de 15 a 20 dB comparado con sistemas no coherentes.
- Compatibilidad con formatos de modulación complejos como DPSK o DQPSK.
- La detección simultánea de la amplitud, fase y polarización de la señal óptica permite obtener información más detallada, que se transportará y extraerá, lo que aumenta la tolerancia a las deficiencias de la red, tales como la dispersión cromática, y mejora el rendimiento del sistema.
- Mejor rechazo a la interferencia de canales adyacentes en sistemas DWDM, permitiendo que más canales sean transportados dentro de la banda de transmisión.

La Figura 1.19 muestra los diagramas de constelación de los formatos de modulación comúnmente populares utilizados con detección digital coherente.

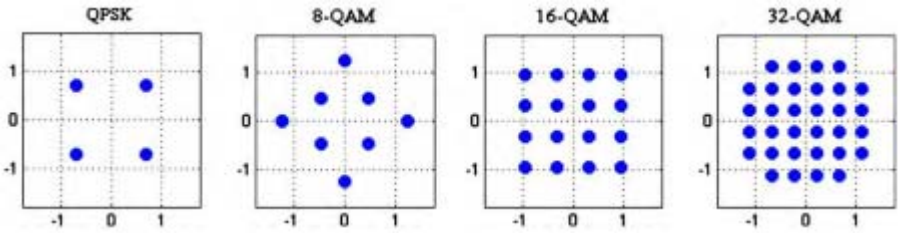


Figura 1.19 Diagramas de constelación de cuatro formatos de modulación comúnmente utilizados con detección coherente digital [135]

Adicional a estos esquemas de modulación de portadora única, se pueden utilizar también métodos de modulación multi-portadora basados en la multiplexación por división de frecuencias ortogonales, OFDM (*Ortogonal Frequency Division Multiplexing*)[8], que habilitada para detección coherente, OFDM óptica coherente (CO-OFDM) trae beneficios similares a los de los sistemas coherentes basados en portadora única descritos anteriormente, tales como alta

[8] OFDM es una tecnología de modulación/multiplexación ampliamente utilizada en comunicaciones inalámbricas y de datos, y ha sido introducida recientemente en las comunicaciones por fibra óptica [135].

eficiencia espectral y alta sensibilidad del receptor, mientras que ofrecen adicionalmente capacidad de adaptación del transmisor y estimación y compensación eficiente del canal [135].

La Figura 1.20a muestra el espectro de canales WDM, cada uno con modulación CO-OFDM, mientras que la Figura 1.20b Muestra el espectro óptico ampliado para cada canal óptico. Además, en la Figura 1.20c se ilustra el espectro de un canal OFDM que consiste de n sub-canales. Cuando el espaciamiento de banda de guarda (Δf_G) entre sub-canales adyacentes es igual a un múltiplo del espaciamiento de subportadora (Δf), cada sub-canal puede ser recibido individualmente sin sufrir diafonía coherente de sub-canales adyacentes debido a que la ortogonalidad entre todas las sub-portadoras (f_1,… f_i,… $f_{j,...}$) es preservada.

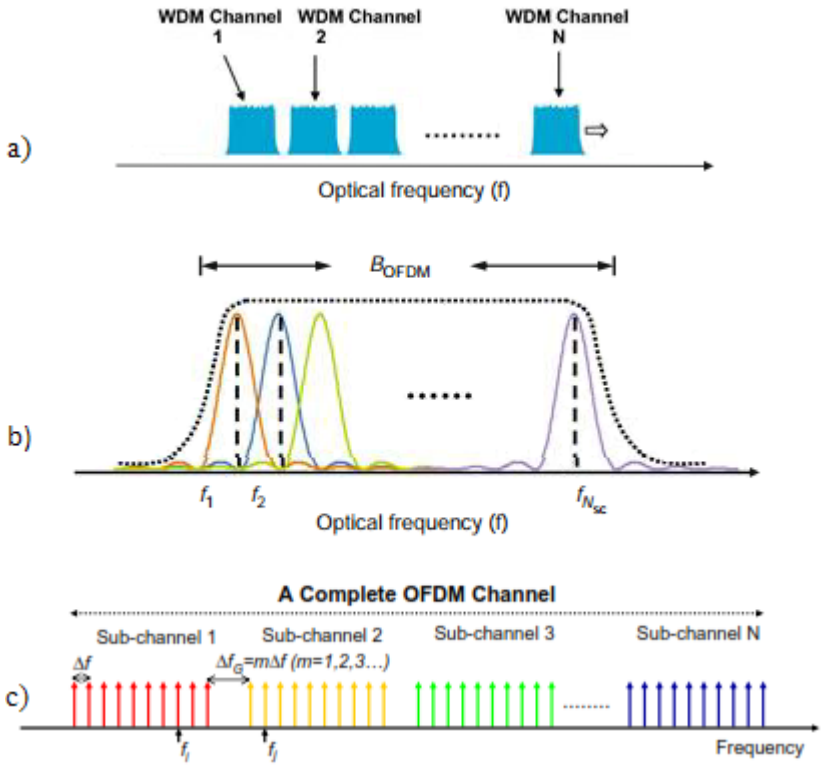


Figura 1.20 Espectro óptico para a) N canales WDM CO-OFDM, b) señal OFDM ampliada para una longitud de onda, c) canal OFDM que consiste de n sub-canales con la ortogonalidad entre todas las sub-portadoras, preservada para permitir acceso a sub-canales libres de diafonía [100]

De hecho, esta característica ha sido explotada recientemente para demostrar la transmisión CO-OFDM sobre 1 Tbps con acceso de sub-canales a 107 Gbps y 121 Gbps. El beneficio de OFDM en la reducción de la diafonía coherente entre sub-canales también ha sido demostrado en OOK multi-portadora, comúnmente referida como WDM coherente, DQPSK de dos portadoras, y PDM-QPSK [135].

La detección coherente también plantea importantes retos para el desarrollo de chips de alta velocidad, tanto para conversión analógica-digital como procesamiento digital de señales. Para la transmisión a estas tasas de bits, cada dB de mejora en OSNR será crítico para conseguir una transmisión de alta capacidad sobre distancias terrestres, y el mejoramiento de la figura de ruido de los amplificadores en línea, por ejemplo, mediante el uso de amplificadores híbridos Raman/EDFA será importante para estos sistemas.

La Tabla 1.1 resume el estado del arte de algunos de los resultados de transmisión de alta capacidad por orden de eficiencia espectral (SE, *Spectral Efficiency*) alcanzada. Un indicador de desempeño clave es el producto SE-distancia (SEDP, *Spectral Efficiency Distance Product*), que está directamente relacionado con el producto de la capacidad de transmisión y la distancia para la misma asignación de ancho de banda óptico.

Tabla 1.1 Estado del arte de las demostraciones de transmisión de alta capacidad [135]

SE (b/s/Hz)	Format/detection	Reach (km)	Fiber type	Optical amplification	Number of channels	SEDP (km-b/s/Hz)
1.4 [50]	43G DQPSK and 107G PDM-DQPSK /DD	1280	SSMF	EDFA	20	**1,792**
2 [49]	111G 2-Carrier NGI-CO-OFDM /DCD	6248	PSCF	2nd-order DRA and EDFA	135	**12,496**
2 [51]	112G PDM-QPSK/DCD	7040	LCF	DRA and EDFA	72	**14,080**
4 [24]	114G PDM-8QAM/DCD	580	Low-loss SSMF	EDFA	320	**2,320**
6.2 [25]	104G PDM-16QAM/DCD	630	SSMF	EDFA and DRA	10	**3,906**
7 [26]	65.1G PDM-OFDM-32QAM /DCD	240	SSMF	EDFA and DRA	8	**1,680**

DD: direct detection with real-time BER measurement; DCD: digital coherent detection with offline DSP; SSMF: standard single-mode fiber; PSCF: low-loss (0.16 dB/km) low-nonlinearity pure silica core fiber; LCF: large-core fiber with 120 μm^2 effective area and 0.184 dB/km loss; EDFA: erbium-doped fiber amplifier; DRA: distributed Raman amplifier

Por otro lado, puesto que la calidad de la señal en una transmisión amplificada ópticamente, tiene una fuerte dependencia con el nivel de OSNR, donde el valor requerido para alcanzar un

BER determinado en un canal óptico depende de su tasa de transmisión, formato de modulación, y esquema de detección.

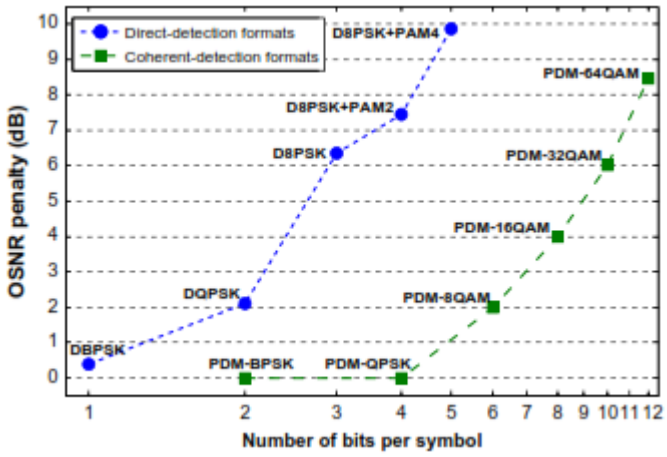


Figura 1.21 Penalidades de OSNR de formatos de detección directa y detección coherente [135]

La Figura 1.21 muestra las penalidades de OSNR[9] de los formatos de detección diferencial directa de alta eficiencia espectral y formatos de detección coherente, utilizando como referencia la modulación por desplazamiento de fase binaria, BPSK (*Binary Phase-Shift Keying*) con detección coherente. Los resultados también se resumen en la Tabla 1.2.

Tabla 1.2 Penalidades de OSNR y margen de fase (ϕ) de modulaciones de alta eficiencia espectral [135]

Differential-detection formats				
	DQPSK	**D8PSK**	**D8PSK+PAM2**	**D8PSK+PAM4**
Bits/symbol	2	3	4	5
OSNRpenalty (dB)	2.1	6.4	7.4	9.9
ϕ-margin (degree)	45	22.5	22.5	22.5

Coherent-detection formats (w/ PDM)					
	QPSK	**8-QAM**	**16-QAM**	**32-QAM**	**64-QAM**
Bits/symbol	4	6	8	10	12
OSNRpenalty (dB)	0	1.98	3.98	6.02	8.45
ϕ-margin (degree)	45	22.5	16.9	10.9	7.7

[9] Nivel de OSNR requerido adicionalmente en dB para un BER dado.

Existen dos observaciones importantes de la Figura 1.20; en primer lugar, los formatos de detección coherente superan sustancialmente a los formatos de detección directa, esto es porque la detección coherente ofrece una mayor sensibilidad del receptor o un menor requerimiento de OSNR que la detección directa, y PDM permite a los formatos de detección coherente duplicar el número de bits por símbolo sin penalidad de OSNR. En segundo lugar, la penalidad de OSNR aumenta rápidamente con el incremento del número de bits por símbolo para ambos esquemas de detección. Por ejemplo, para alcanzar 5 bits/símbolo con detección directa D8PSK+PAM4[10] se tiene que pagar una penalidad de casi 10 dB, mientras que para alcanzar 12 bits/símbolo con PDM-64QAM[11] la penalidad de OSNR es de aproximadamente 8.5 dB. Esto significa que existe un compromiso entre el rendimiento de OSNR y la eficiencia espectral alcanzada. En la Tabla 1.2 se muestra también otro indicador de desempeño de un formato de modulación que es el margen de fase[12] [135].

Las actuales demandas de tráfico pueden ser satisfechas por sistemas de 100G, y aunque constituyen sólo el comienzo de la fase de fabricación y despliegue, los principales grupos de investigación del *networking* óptico se han estado enfocando en estándares y tecnologías con capacidades superiores. La transmisión de datos a una velocidad de 400G, es el siguiente paso después de la llegada al mercado de 100G que supuso un importante logro en materia de innovación. Para muestra, el caso de la empresa multinacional Alcatel-Lucent que en el año 2012 lanzó la primera solución comercial DWDM con funcionalidades OTN de 400G (Gbps), correspondiente a su plataforma PSS 1830 (*Photonic Service Switch*) soportada en el desarrollo de un nuevo chip para redes ópticas denominado PSE (*Photonic Service Engine*), basado en la innovación de los Laboratorios Bell y la experiencia comercial de la compañía con el gran despliegue de soluciones 100G coherentes; que apunta a mejorar la economía y las prestaciones de los actuales sistemas 100G, ampliando y acelerando su adopción en el mercado.

[10] PAM-4 es una técnica avanzada de modulación de 4 niveles para transportar la información y ha sido adoptada para algunos aspectos del estándar IEEE P802.3bs para 400 Gbps.
[11] 64-QAM es una técnica de modulación avanzada de 6 niveles que provee alta eficiencia.
[12] Rotación de fase máxima permitida en la constelación de la señal antes de que un símbolo sea confundido con otro símbolo en ausencia de ruido.

El PSE 400G se puede utilizar en una variedad de formas dependiendo de los requerimientos de los proveedores de servicios. Cuando se configura para el transporte de 100G, puede optimizar el rendimiento, extendiendo el alcance en más del 50% (de 2.000 a más de 3.000 km) sin la necesidad de una costosa regeneración eléctrica reduciendo de esta manera el consumo de energía y el espacio físico en una tercera parte. El PSE 400G permite a los proveedores de servicios alcanzar la capacidad de longitud de onda completa de la fibra de cualquier calidad, reduciendo la necesidad de comprometerse en longitudes de onda o velocidad. En aplicaciones de 400G, el PSE incrementa la capacidad de tráfico por fibra en más de 2,6 veces y reduce el consumo de energía por gigabit en un 33% [145], como se muestra a continuación en la Tabla 1.3.

Tabla 1.3 Comparativa entre los actuales sistemas 100G y el nuevo chip PSE 400G [17]

	STATE-OF-THE-ART 100G	400G PSE
Year	2010	2012
Speed	100G	400G
Line rates	40G, 100G	40G, 100G, 400G
Reach	2000 km	> 3000 km
Line capacity	8.8Tbps	> 23Tbps

Con respecto a este avance tecnológico, France Telecom-Orange y Alcatel-Lucent (ahora Nokia) anunciaron en el mes de febrero de 2013 el despliegue del primer enlace óptico del mundo que ofrece una capacidad de 400 Gigabits por segundo (Gbps) por longitud de onda, en una red con tráfico real. Tras el éxito de las pruebas de campo, ahora se encuentra operativo un enlace óptico de 400 Gbps entre París y Lyon. Este enlace, que se ha desplegado en el entorno operativo de France Telecom-Orange, representa todo un hito en la tecnología de redes terrestres de larga distancia. Con una velocidad cuatro veces superior a la actualmente disponible y utilizando 44 longitudes de onda, este nuevo enlace óptico puede transmitir un tráfico total de hasta 17,6 Terabits por segundo (Tbps) [137].

1.3 Tecnologías de conmutación para WDM [22,24,27,29][36][41,43][59] [61,66][75][81-83][91][110]

El desarrollo de las tecnologías de conmutación óptica se conoce genéricamente como Optical 'X' Switching (OXS), donde X = {C, P, B, L, F} para circuitos, paquetes, ráfagas, etiquetas, y flujos, respectivamente [82]. Las tres técnicas principales propuestas para transportar tráfico IP sobre redes ópticas basadas en WDM son OCS, OPS y OBS, las mismas que se describen y comparan a continuación.

1.3.1 Conmutación Óptica de Circuitos (OCS)

Es una tecnología propuesta dentro del contexto de las redes completamente ópticas, donde los circuitos son conmutados por los nodos intermedios en la granularidad de un canal de longitud de onda. Por consiguiente, las redes de conmutación óptica de circuitos (OCS, *Optical Circuit Switching*) se denominan también redes de enrutamiento por longitud de onda [75], que proveen un servicio de transporte orientado a la conexión, mediante el establecimiento de caminos ópticos dedicados, conocidos como *lightpaths*, entre un par de nodos origen y destino a través de canales DWDM sobre cada enlace a lo largo de la ruta física, que se configuran previamente para permitir la transmisión de datos extremo a extremo, que permanece en el dominio óptico (sin necesidad de conversiones OEO en los nodos intermedios), hasta que termina la conexión establecida, luego de lo cual se liberan los recursos reservados para poder ser utilizados posteriormente por otras conexiones. En las redes OCS todo el ancho de banda de cada *lightpath* está dedicado a una conexión entre un par de nodos origen y destino, y el ancho de banda no utilizado no puede ser demandado por otros nodos listos para enviar los datos. Por lo tanto OCS, no permite la multiplexación estadística y conduce por tanto a una baja utilización del ancho de banda, como se menciona más adelante.

La provisión de circuitos completamente ópticos en redes OCS son transparentes en términos de tasa de transmisión, esquema de modulación y protocolo, donde los nodos de conmutación son referidos usualmente como matrices de conmutación óptica (OXC), que son responsables

de la conmutación completamente óptica de datos transportados de una longitud de onda en el puerto de entrada hacia una longitud de onda en el puerto de salida, como se ilustra en la Figura 1.22. Si los OXC son equipados con convertidores de longitud de onda, un *lightpath* puede ser convertido de una longitud de onda a otra a lo largo del camino. En caso contrario, será necesario utilizar la misma longitud de onda en todos los enlaces de la red, propiedad conocida como restricción de continuidad de longitud de onda. Un canal óptico DWDM puede ser utilizado por diferentes *lightpaths* siempre y cuando no compartan ningún enlace en común, permitiendo de esta manera el reúso espacial de longitudes de onda en diferentes tramos de la red [24].

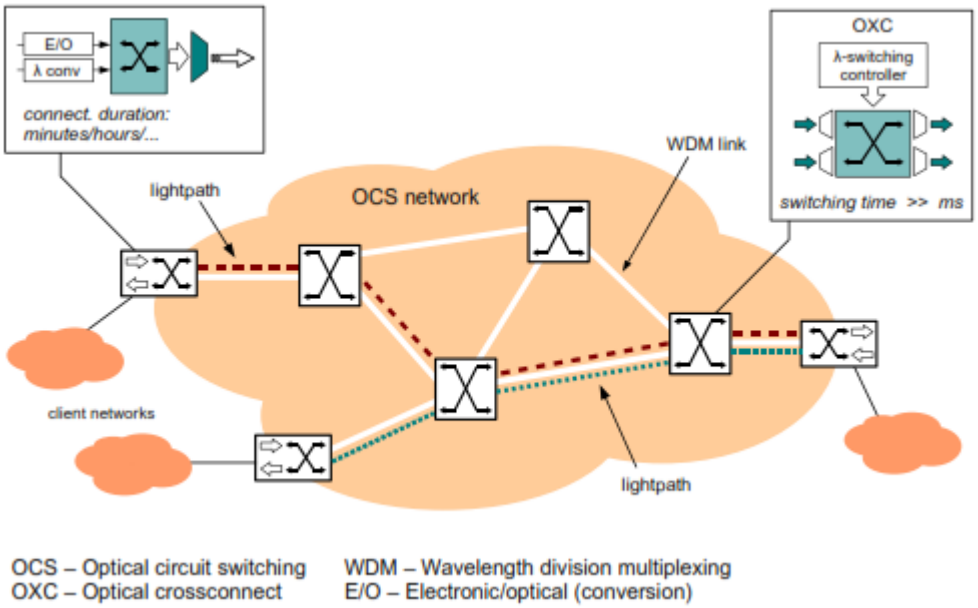


Figura 1.22 1.22 Red OCS [27]

El establecimiento de caminos ópticos involucra varias tareas que incluyen el descubrimiento de la topología y recursos, enrutamiento, asignación de longitudes de onda, señalización y reserva de recursos. El descubrimiento de la topología y recursos comprende el mantenimiento y distribución de la información de estado de la red (p. ej. topología física y disponibilidad de ancho de banda de los enlaces). La búsqueda para encontrar rutas y asignar longitudes de onda a lo largo de la red, se conoce como problema de enrutamiento y asignación de longitud de

onda (RWA, *Routing and Wavelength Assignment*), que no es trivial, y se traduce en un problema combinatorio conocido como NP-completo[13] [36], en el cual la determinación del enrutamiento es normalmente resuelto mediante algoritmos que calculan la ruta más corta, rutas alternativas o esquemas multi-camino, mientras que la asignación de longitudes de onda se realiza utilizando coloreado de grafos o métodos heurísticos.

Por lo general, las solicitudes de conexión pueden ser de dos tipos, estáticas o dinámicas. En el primer caso, los requerimientos de tráfico se conocen de antemano, y el problema se traduce a configurar los *lightpaths* manualmente para estas conexiones, con el fin de minimizar los recursos de red como el número de longitudes de onda o el número de fibras. En el segundo caso, los *lightpaths* se configuran automáticamente a medida que llegan las solicitudes de conexión y se liberan después de una cierta cantidad finita de tiempo, cuyo objetivo es minimizar el bloqueo de conexión y por tanto maximizar el número de conexiones establecidas en la red en cualquier momento [59].

La técnica de OCS es una alternativa viable y suficientemente madura, por lo cual es utilizada en las redes totalmente ópticas actuales, sin embargo, tiene algunas limitaciones. En primer lugar, los *lightpaths* son conexiones bastante estáticas y fijas en ancho de banda que no son capaces de adaptarse eficientemente a la naturaleza altamente variable y a ráfagas del tráfico de Internet. En tal caso, esta técnica conduce más bien en una subutilización del ancho de banda debido a que el tráfico a ráfagas requiere una capacidad reducida (a nivel de sub-longitud de onda) en comparación con la ofrecida por una longitud de onda y además porque se desperdicia la capacidad del circuito reservado durante los instantes inactivos en los cuales no se transmiten datos. Por esta razón, la baja granularidad de conmutación a nivel de longitud de onda de OCS llega a ser cada vez más ineficiente e impráctica, siendo más apropiada para largas transmisiones de datos con perfiles de tráfico estable que demandan gran ancho de banda, cuyo tiempo para mantener una larga conexión en el orden de unos cuantos minutos, horas, días, semanas, o incluso meses justifican el *overhead* de reservación en dos vías para establecer o liberar un *lightpath*, que puede tomar unos pocos cientos de milisegundos [75].

[13] Problema que no ha podido ser resuelto de manera exacta por medio de algoritmos deterministas eficientes, pero que puede ser resuelto por algoritmos no deterministas [36].

Otro inconveniente de esta técnica es que, el número de conexiones en una red es usualmente mucho mayor que el número de longitudes de onda, por lo tanto esta limitación de recursos se traduce en problemas de flexibilidad y escalabilidad, puesto que a pesar del reúso espacial de longitudes de onda, no es posible ni eficiente asignar una longitud de onda a cada conexión.

La investigación de métodos alternativos para transportar datos a través de las redes ópticas que permitan un uso más flexible y eficiente del ancho de banda, dio lugar a dos nuevos paradigmas de redes todo ópticas descritos a continuación.

1.3.2 Conmutación Óptica de Paquetes (OPS)

Debido a que el tráfico de datos está dominando en la red y que la conmutación de circuitos no constituye una alternativa optimizada para el tráfico de datos, ha habido un considerable interés por parte de la comunidad de investigación en la técnica que conmutación óptica de paquetes (OPS, *Optical Packet Switching*), donde la conmutación se realiza en la granularidad de paquetes fotónicos, los cuales son conmutados y enrutados de forma independientemente a través de la red sin ningún tipo de reserva previa, como se ilustra en la Figura 1.23, constituyéndose en una alternativa más adecuada para manejar el tráfico con naturaleza a ráfagas.

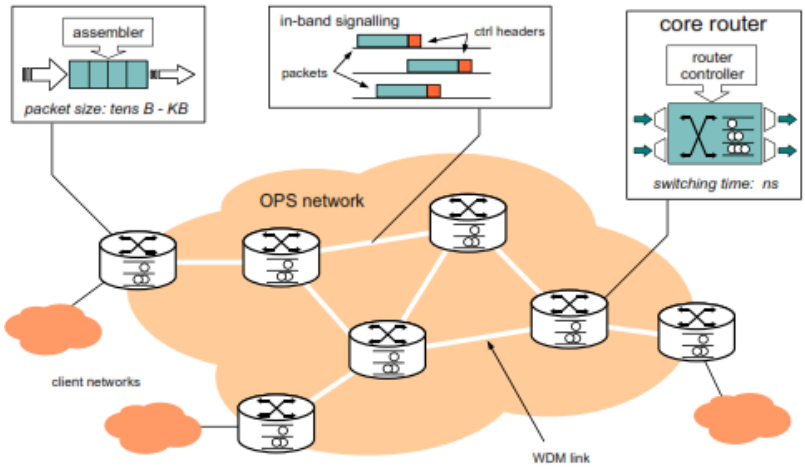


Figura 1.23 Red OPS [27]

A diferencia de OCS, OPS es capaz de proporcionar una ganancia significativa de multiplexación estadística sobre los enlaces de fibra óptica, debido al hecho de que el ancho de banda no está dedicado a una sola conexión sino que puede ser compartido por múltiples flujos de datos [75].

El objetivo principal de OPS es permitir capacidades de conmutación de paquetes a velocidades comparables con las de los enlaces ópticos y de esta manera reemplazar el enrutamiento por longitud de onda en las redes ópticas de próxima generación [24], con el fin de dar una solución a la incompatibilidad existente entre la capacidad de transmisión ofrecida por la capa óptica de DWDM y la capacidad de procesamiento electrónico de los conmutadores y enrutadores actuales, ampliamente referida como cuello de botella opto-electrónico [75].

El datagrama OPS debe ser completamente óptico para alcanzar plenamente todas sus ventajas fundamentales sobre OCS; sin embargo, esto requiere que el reconocimiento/procesamiento de las cabeceras, así como también las operaciones de control, se realicen en el dominio completamente óptico sobre una base paquete por paquete. No obstante, esta forma deseada de OPS está más allá del alcance tecnológico de hoy en día, debido a que requiere técnicas y componentes que todavía no se encuentran disponibles por completo a tal grado de que sean comparables con la conmutación de paquetes electrónicos ampliamente utilizada en las redes de datos actuales. Por lo tanto, OPS se plantea inicialmente como un esquema basado en el control electrónico, e incluso bajo esta consideración enfrenta grandes desafíos, tales como la necesidad de tecnologías de conmutación óptica ultra-rápidas en escalas de tiempo de nanosegundos, memorias/*buffers* ópticos, delineación y sincronización de paquetes ópticos. Este tipo de implementación particular de OPS es conocida también como conmutación óptica de etiquetas (OLS*, Optical Label Switching*), donde la cabecera, referida como etiqueta, es procesada electrónicamente para propósitos de enrutamiento, mientras que la carga útil es conmutada en el dominio óptico. Adicionalmente, OLS provee una plataforma unificada para la facilitar la interoperabilidad entre las distintas técnicas de conmutación de circuitos, paquetes y ráfagas que es posible mediante el uso de GMPLS.

Únicamente tecnologías con muy alta velocidad de conmutación han sido consideradas como candidatas a ser utilizadas en redes OPS, como es el caso de los amplificadores ópticos de

semiconductor (SOA, *Semiconductor Optical Amplifier*) que proveen tiempos de conmutación menores a 1 ns [59]. Debido a que los recursos de la red no son reservados de forma previa para la transmisión de los paquetes, estos pueden experimentar contención dentro del dominio OPS (cuando dos o más paquetes compiten por el mismo recurso de salida al mismo tiempo), que en redes tradicionales se resuelve mediante *buffers* electrónicos; sin embargo, debido a que no existe el equivalente óptico de la memoria RAM electrónica, el *buffering* en el dominio óptico se realiza utilizando líneas de retardo de fibra (FDL, *Fiber Delay Line*) o líneas de retardo conmutadas (SDL, *Switched Delay Line*) que hacen uso de la dimensión temporal, aunque existen también técnicas de resolución de contenciones que explotan las dimensiones de longitud de onda (conversión de longitud de onda), y/o espacio (enrutamiento por deflexión). Por otro lado, la sincronización y alineación de paquetes en los puertos de entrada del conmutador se podría utilizar con el fin de minimizar las contenciones.

Cuando un paquete óptico llega a un nodo OPS, una pequeña porción de la potencia óptica del paquete es desviada hacia el plano de control, donde se extrae la cabecera para convertirla al dominio electrónico a fin de que pueda ser procesada, y que en base a la información contenida se configurará la matriz de conmutación para permitir que el paquete atraviese el nodo hacia el siguiente salto para continuar el tránsito hacia su destino. Antes de que el paquete abandone el puerto de salida del conmutador, se inserta una nueva cabecera óptica (y la original es eliminada), como se puede observar en la Figura 1.24.

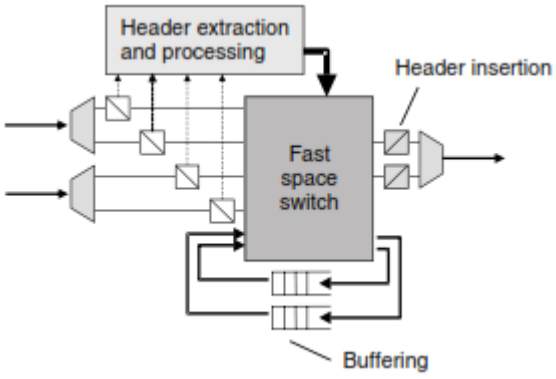
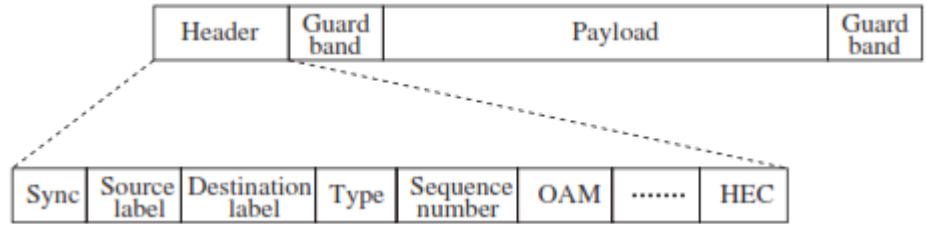


Figura 1.24 Extracción, procesamiento e inserción de cabecera en un nodo OPS [61]

Además, debido a que es necesario un cierto período de tiempo para procesar la cabecera y reconfigurar el conmutador, la carga útil del paquete se retrasa utilizando normalmente líneas de retardo de fibra.

La Figura 1.25 muestra el formato genérico de un paquete óptico que consiste de una cabecera y carga útil con bandas de guarda antes y después de los datos, utilizadas para soportar las incertidumbres en tiempo de los paquetes ópticos que llegan a los nodos OPS.



Sync: Delineation and synchronization bits.
Source Label: Source node address.
Destination Label: Destination node address.
Type: Type and priority of packet and carried payload.
Sequence Number: Packet sequence number to reorder packets arriving out of order and guarantee in-order packet delivery.
OAM: Operation, administration, and maintenance functions.
HEC: Header error correction.

Figura 1.25 Formato genérico de un paquete óptico [75]

Debido a que la cabecera óptica necesita únicamente un número relativamente pequeño de bits, la tasa de transmisión de la cabecera óptica puede ser mucho menor que la de los datos ópticos, lo cual permite simplificar el procesamiento electrónico. Existen varios métodos para el etiquetado los paquetes ópticos como: multiplexación por división de tiempo (TDM, *Time Division Multiplexing*), multiplexación por división de código óptico (OCDM, *Optical Code Division Multiplexing*), multiplexación por subportadora (SCM, *Subcarrier Multiplexed*), modulación ortogonal (*Orthogonal Modulation*) y etiquetado con multiplexación por división de onda (*Wavelength-Division Multiplexing Labeling*), los cuales que se describen brevemente a continuación [81]:

En los primeros cuatro métodos, la etiqueta se asigna al paquete de información en el mismo canal espectral, mientras que en el último se utiliza un canal diferente para transportar la etiqueta. El etiquetado TDM combina la etiqueta óptica en el dominio del tiempo al inicio de la carga útil, formando un paquete óptico que se codifica en la misma longitud de onda, lo que hace necesario utilizar bandas de guarda y bits de sincronización.

El etiquetado OCDM inserta la etiqueta mezclando mediante *scrambling* la carga útil con un código específico que contiene la información de la etiqueta, y aunque es una técnica que permite el reconocimiento de etiquetas para enrutamiento en lugar de operaciones en tablas de búsqueda, su implementación es bastante compleja, puesto que si una longitud de onda soporta N códigos OCDM, un banco de N autocorreladores ópticos por longitud de onda es necesario para cada canal y una réplica del canal para cada uno de los autocorreladores.

En el etiquetado SCM la etiqueta es modulada en una subportadora de RF que se multiplexa con la carga útil en el mismo canal óptico. En el etiquetado ortogonal la información de la etiqueta se inserta en la fase o la frecuencia (DPSK o FSK) de la portadora de los datos que está modulada en amplitud (ASK, *Amplitude Shift Keying*), aprovechando la ortogonalidad de estas variables para transmitir la información del paquete por separado. En el etiquetado WDM las etiquetas de cada canal se multiplexan y se envían por un canal dedicado para este propósito utilizando una o múltiples longitudes de onda. De las técnicas anteriores, la que ofrece mejores prestaciones es el etiquetado SCM ya que permite la posterior separación de la etiqueta por medio de técnicas comunes de filtrado óptico y por otra parte no requiere un sincronismo exacto entre cada paquete de datos y su correspondiente etiqueta, algo que si es necesario en la técnica de etiquetado TDM y que se traduce en elevados costes de la electrónica necesaria para mantener completamente sincronizado el sistema [91].

Se pueden diferenciar dos categorías de redes ópticas conmutadas por paquetes: *ranuradas* o sincrónicas y *no ranuradas* o asincrónicas. En el primer caso, sobre el que se enfocó inicialmente OPS, se requiere una sincronización a nivel de bit y una rápida recuperación del reloj para el reconocimiento de la cabecera y alineación de los paquetes, donde se considera una ranura de tiempo de longitud fija que contiene la carga útil y cabecera del paquete, cuya duración es mayor que el paquete completo para proveer un tiempo de guarda, como se observa en la Figura 1.26, para el caso del proyecto ACTS KEOPS (*KEys to Optical Packet Switching*), que

utilizó una tasa de transmisión fija para codificar la cabecera de 622 Mbps y una velocidad de línea para la carga útil que podía variar desde cientos de Megabits por segundo hasta 10 Gbps [43].

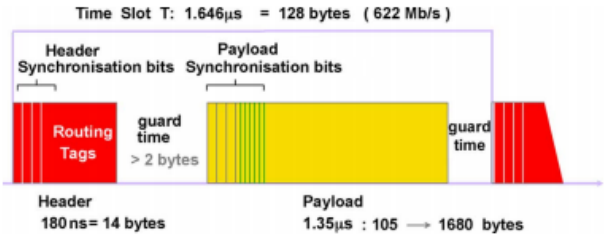


Figura 1.26 Formato del paquete óptico KEOPS [75]

En las *redes ranuradas*, todos los paquetes que llegan a los puertos de entrada (ver Figura 1.27) necesitan ser alineados en fase antes de ingresar a la matriz de conmutación y como se mencionó anteriormente, un derivador (*tap*) divide una pequeña cantidad de potencia de la señal entrante para que la unidad de control, luego del reconocimiento del preámbulo, realice la lectura de la cabecera.

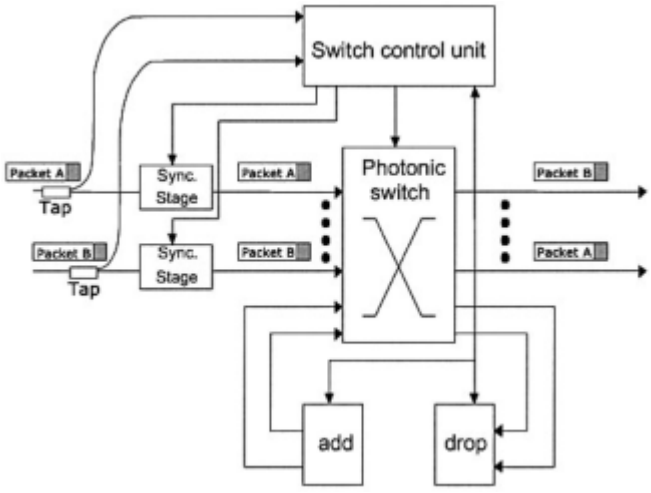


Figura 1.27 Arquitectura de un nodo genérico de la red sincrónica [82]

Adicionalmente, la información de temporización del paquete entrante pasa hacia la unidad de control para configurar las etapas de sincronización y la matriz de conmutación. Una vez que el procesador de cabecera reconoce el patrón de bits iniciales para intentar realizar el proceso de alineación, identifica el instante inicial del paquete y la unidad de control se encarga de calcular el retardo necesario a través de las líneas de retardo conmutadas que consisten básicamente de conmutadores 2x2 y líneas de retardo de fibra establecidas como una secuencia exponencial que corresponden con una fracción de la ranura de tiempo, como se ilustra en la Figura 1.28. Cabe mencionar que podría también ser o no necesaria, una etapa de sincronización a la salida del conmutador para compensar la variación rápida de tiempo (*jitter*) inducido dentro del nodo que depende principalmente de la arquitectura del conmutador.

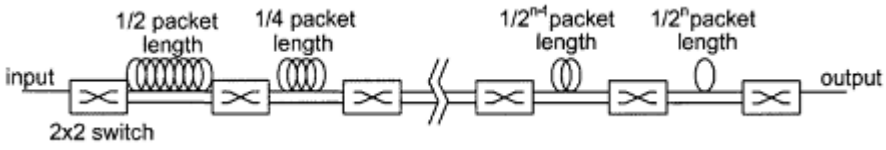


Figura 1.28 Esquema para la etapa de sincronización de entrada en un nodo [82]

Para el caso de las *redes no ranuradas*, los paquetes pueden o no tener el mismo tamaño y arriban al conmutador sin ser alineados en fase, por lo cual, la operación de conmutador paquete por paquete podría tener lugar en cualquier instante de tiempo.

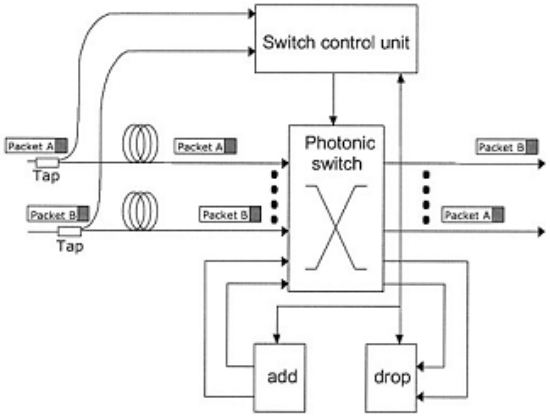


Figura 1.29 Arquitectura genérica de un nodo de la red asincrónica [82]

En la Figura 1.29 se presenta la arquitectura general de un nodo en una red asincrónica que no requiere de etapas sincronización y ni alineación como en el caso anterior. Las líneas de retardo de fibra mantienen el paquete mientras se lleva a cabo el procesamiento de la cabecera y reconfiguración del conmutador.

En una red asincrónica, la probabilidad de contención es mayor que en una red sincrónica debido a que el comportamiento de los paquetes es más impredecible; sin embargo, las redes asincrónicas son más flexibles comparadas con sus contrapartes sincrónicas, ya que permiten adaptar de mejor manera paquetes de longitud variable [82].

Finalmente, es importante mencionar que gracias a grandes esfuerzos de investigación y progresos que se ha tenido en el ámbito de la tecnología óptica, se han logrado desarrollar esquemas que han sido probadas en escenarios de laboratorio, y que se pueden consideran constituyen los primeros pasos para la implementación de esta tecnología en equipos comerciales que se prevé pueda ser una realidad en el largo plazo. En contraste, el paradigma de conmutación óptica por ráfagas (OBS) descrito brevemente a continuación y el plano de control de GMPLS se visualizan como alternativas más realistas en el mediano o corto plazo.

1.3.3 Conmutación Óptica de Ráfagas (OBS)

La tecnología utilizada actualmente en los sistemas de transporte óptico DWDM comerciales, permite velocidades de línea de 10 Gbps, 40 Gbps, 100 Gbps e incluso superiores con 80 o más canales ópticos de transmisión simultánea. Esta enorme capacidad puede ser manejada exclusivamente en la capa óptica cuando se utilizan ROADMs con múltiples grados para la conmutación en los nodos de la red, basada normalmente en la tecnología de conmutación óptica de circuitos (OCS), que tiene como principal desventaja su granularidad puesto que ofrece conexiones ópticas con grandes anchos de banda determinados, equivalentes a una longitud de onda, que no son adecuados para el tráfico de datos dominante en las redes actuales, ya que los servicios y aplicaciones exigen velocidades de datos comparativamente más bajas en el rango de Mbps, o algunos Gbps, lo cual sumando a la falta de multiplexación estadística, resulta en una subutilización del ancho de banda.

Con el fin de utilizar de una manera más eficiente el ancho de banda, surge la tecnología OTN definida en la norma ITU-T G.709 como una envoltura digital que ofrece una mayor granularidad y agregación de tráfico, al encapsular distintas tramas de datos de diferentes fuentes en unidades de información que pueden ser procesadas y transmitidas como entidades individuales, bajo una jerarquía multiplexada de unidades de datos ópticos ODU-k organizadas en unidades de transporte óptico OUT-k que son transportadas sobre los canales de longitud de onda ópticos provistos a través de la red OCS.

El objetivo principal es reducir el costo del manejo de tráfico en la red, haciendo un mejor uso de la capacidad disponible y en particular la utilización de este tipo de *grooming* de tráfico intermedio a nivel de ODU-k (sub-longitud de onda), puede reducir el uso de longitudes de onda en un 40% [83] como se ilustra en la Figura 1.30, ya que permite empaquetar más flujos dentro de cada canal óptico; sin embargo, al ser un enfoque que emplea el dominio electrónico adolece de las desventajas expuestas anteriormente. Por lo tanto, la investigación actual en redes ópticas apunta hacia alternativas que permitan la conmutación de paquetes de forma completamente óptica, con el fin de obtener una infraestructura más eficiente en cuanto a costo, espacio físico y consumo energético por bit.

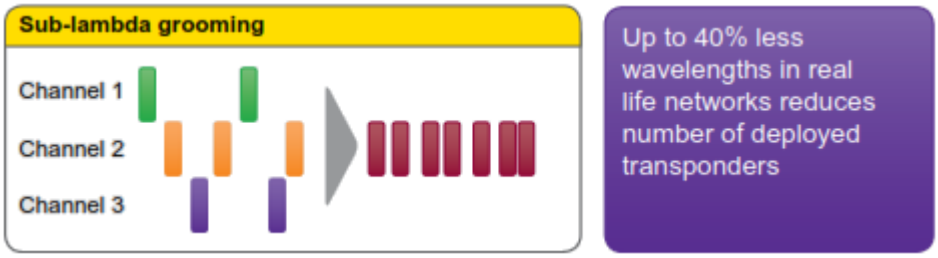


Figura 1.30 El grooming a nivel de sub-lambda empaqueta más flujos en cada longitud de onda y por lo tanto, reduce eficazmente el número de longitudes de onda distintas requeridas [83]

Es así que la visión a largo plazo apunta a la conmutación óptica de paquetes (OPS), donde parte del desafío que encara esta tecnología se debe a la granularidad de conmutación extremadamente pequeña y al deseo de leer/escribir los encabezados de los paquetes en el dominio óptico [29]; y a pesar de los continuos avances que ha tenido, su grado de desarrollo

tecnológico es aún bajo quedando todavía mucho camino por recorrer para conseguir una realización madura que esté lista para implementaciones prácticas.

En respuesta a la complejidad de la implementación de OPS y a la necesidad de mejorar la flexibilidad y el desempeño de las soluciones basadas en OCS, surge como una solución intermedia la tecnología denominada conmutación óptica de ráfagas (OBS, *Optical Burst Switching*), propuesta inicialmente por Qiao [22] y Turner [110] en 1999 como un paradigma prometedor para las redes ópticas de nueva generación basadas en WDM, que combina las ventajas de la conmutación óptica de circuitos (OCS, *Optical Circuit Switching*) y la conmutación óptica de paquetes (OPS, *Optical Packet Switching*) mientras que evita sus respectivas desventajas, permitiendo aprovechar el ancho de banda más eficientemente gracias a que ofrece una mayor granularidad y ganancia de multiplexación estadística, al utilizar de forma transparente la enorme capacidad de las fibras para la transmisión/conmutación y la capacidad de procesamiento sofisticado de la electrónica, con el fin de reducir los costos de la red y explotar los avances tecnológicos de los mundos óptico y electrónico, de manera que permitan apalancar a OBS como una alternativa para la arquitectura de un *core* de conmutación IP sobre WDM para el Internet Óptico de nueva generación.

La tecnología OBS se caracteriza por un enfoque híbrido, cuya premisa fundamental es la separación de los planos de control y de datos, y la segregación de sus funcionalidades dentro de los dominios electrónico y óptico respectivamente, con lo cual se evita la necesidad de lectura/escritura óptica de las cabeceras, que es parte del desafío que encara OPS. En contraste al esquema tradicional de *store-and-forward* utilizando en los equipos actuales, OBS es vista como una tecnología del tipo *cut-through*, es decir, que transfiere los datos directamente a través de la red sin necesidad de almacenamiento en los nodos intermedios, permitiendo además que múltiples flujos de datos puedan compartir un canal de longitud de onda, lo cual se traduce en que ofrece granularidad de sub-longitud de onda que constituye un nivel intermedio en algún punto entre un paquete y un circuito (camino óptico) [29]. La característica de permitir la compartición de un canal óptico será cada vez más importante ya que las tasas de bit han incrementado de 40 Gbps a 100 Gbps e incluso más.

El principio de operación de OBS se fundamenta en su unidad de transporte básica denominada ráfaga, que es un contenedor de mayor tamaño, normalmente de longitud

variable, compuesto de varios paquetes de datos provenientes de las redes de clientes que son ensamblados en el nodo de ingreso en base a su destino y/o requerimientos de calidad de servicio. El tamaño de la ráfaga puede variar desde un único paquete hasta un gran conjunto de datos, con una duración típica de algunos microsegundos a varios cientos de milisegundos. Cada ráfaga es transmitida en la forma de dos componentes hacia el núcleo de la red OBS a través de canales de longitud de onda separados, un pequeño paquete de control o cabecera de ráfaga (BHP*, Burst Header Packet*) y una ráfaga de datos (DB, *Data Burst*), donde únicamente los paquetes de control se someten a conversiones OEO en cada nodo intermedio para su procesamiento electrónico a fin de llevar a cabo las decisiones de conmutación/enrutamiento en el plano de datos, mientras que las ráfagas de datos se transmiten asincrónicamente en un conjunto separado de canales de datos que son conmutados de forma completamente óptica dentro del dominio OBS, eliminando de este modo la complejidad de la sincronización.

La transmisión todo-óptica que se logra gracias a que los nodos troncales son configurados con anterioridad a la llegada de la ráfaga, asegura una comunicación transparente dentro el núcleo de la red, independientemente del contenido, protocolo, tasa de transmisión, formato de modulación y codificación de los datos originales. Además, dado que un paquete de control tiene un tamaño significativamente menor que una ráfaga de datos, un canal de control es suficiente para transportar los paquetes de control asociados con múltiples (p.ej. cientos) canales de datos [22].

La Figura 1.31 muestra una red OBS con sus tres componentes básicos: un nodo de ingreso, un nodo de egreso, y una red de nodos de *core*. En el nodo de ingreso, varios tipos de datos de clientes provenientes de la red de acceso se agregan en una ráfaga de datos que se transmite completamente en el dominio óptico. Los paquetes destinados al mismo nodo de egreso (nodo destino de borde) y que requieren el mismo nivel de servicio (si se admiten múltiples clases) se agregan en una ráfaga en una cola de ensamblado de ráfagas. Para evitar el almacenamiento y procesamiento óptico en los nodos intermedios denominados nodos de *core*, antes de la transmisión de cada ráfaga, el nodo de ingreso genera y envía un paquete de control en un canal óptico dedicado, que contiene información sobre la longitud y el tiempo de llegada de la ráfaga, en base a la cual, los nodos intermedios pueden configurar sus matrices de conmutación para reservar los recursos (puerto, longitud de onda, etc.) a lo largo de un

determinado camino, proporcionando un canal óptico a través del cual las ráfagas de datos pueden ser transmitidas desde la fuente hasta el destino final permaneciendo el dominio óptico, después de un lapso de tiempo denominado *offset*[14], para compensar el retardo de procesamiento de los paquetes de control y la configuración de los conmutadores en los nodos intermedios.

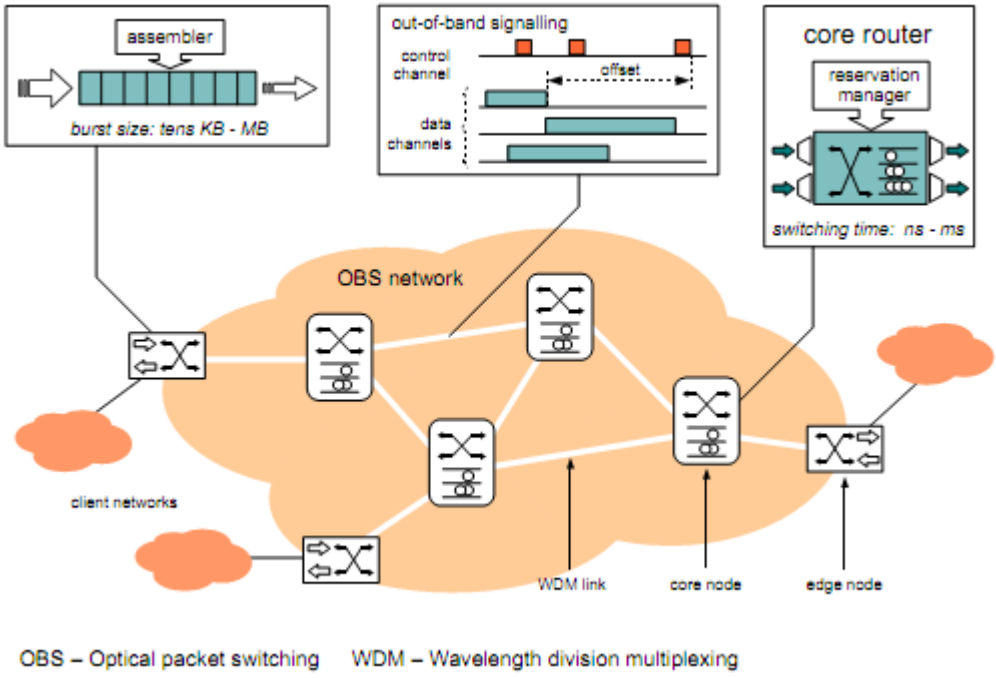


Figura 1.31 Red OBS [27]

Mediante una selección apropiada del tiempo de *offset*, un algoritmo de planificación puede garantizar que un canal de longitud de onda de salida esté configurado el momento que la ráfaga de datos llega a un nodo de conmutación intermedio. Una vez que una ráfaga alcanza el nodo de egreso, ésta se desensambla en sus paquetes originales y estos últimos son enrutados hacia sus destinos respectivos fuera del dominio OBS. Durante el proceso de ensamblado/desensamblado de ráfagas los datos de los clientes son almacenados

[14] Lapso durante el cual, los nodos intermedios tienen tiempo para reservar recursos para el transporte de la ráfaga que se va a transmitir.

temporalmente en el borde de la red en memorias RAM electrónicas que son económicas y abundantes.

Con esta tecnología, los recursos de la red pueden ser reservados dinámicamente por períodos pequeños y muy específicos de tiempo, permitiendo asignar el ancho de banda de una manera más eficiente y obtener un mayor grado de multiplexación estadística, que no era posible mediante la tecnología OCS. Además, ya que los datos se transmiten en entidades más grandes, el *overhead* de procesamiento puede ser reducido y los requerimientos tecnológicos para la conmutación son menos exigentes si se comparan con la necesidad de rápidos conmutadores ópticos esenciales para OPS. Esto se debe principalmente al hecho de que las ráfagas, al tener una mayor longitud que un simple paquete, permiten reducir la frecuencia con la que se deben conmutar y procesar las cabeceras, y es más viable con la tecnología actual como se verá más adelante al abordar los componentes ópticos habilitantes de OBS en la sección 2.2.

Las principales ventajas de OBS en comparación con los otros esquemas de conmutación óptica son que a diferencia de las redes ópticas conmutadas por longitud de onda, el ancho de banda óptico es reservado únicamente para la duración de la ráfaga, y a diferencia de las redes OPS, puede operar sin almacenamiento y debido a que goza de un overhead de procesamiento reducido, gracias al ensamblado de múltiples paquetes en ráfagas, necesita únicamente una velocidad de reconfiguración del conmutador en el orden de microsegundos, que es un aspecto importante ya que tiene un impacto en el tamaño de la ráfaga que se puede manejar. A pesar de que estas diferencias y otras imponen consideraciones especiales de diseño y operación, sus aspectos de implementación como: procesamiento más fácil, operación de almacenamiento reducida/ninguna, y requerimientos de conmutación relativamente lentos [61], hacen que OBS sea una de las tecnologías que probablemente haya recibido la mayor parte de atención de la comunidad científica en los últimos años, y que pueda ser vista como una solución práctica que puede ser considerada como el siguiente paso en la evolución futura del *networking* óptico.

La tecnología de conmutación óptica de ráfagas se podría emplear tanto en redes de transporte metropolitanas o regionales como en redes de *backbone*. En el primer caso, OBS debería que ser capaz de transmitir datos hasta 200 km y soportar un tráfico dinámico. A corto plazo, se puede emplear OBS en redes en anillo, por lo que la conmutación en los nodos se

simplifica. A medio/largo plazo, se debería migrar hacia redes en malla para ofrecer una mayor tolerancia a fallos. En el segundo caso, OBS tendría que transmitir grandes volúmenes de datos a cientos o miles de kilómetros de distancia; y a diferencia de las redes metropolitanas, el tráfico agregado previamente de las conexiones a transportar, es más estable y no tan dinámico [41]. Por lo tanto, OBS podría ser adoptada en un futuro para soportar el transporte de información a través de Internet, ya que permitiría minimizar el procesamiento electrónico superando el cuello de botella de los enrutadores actuales y a su vez enrutar y conmutar a nivel completamente óptico grandes volúmenes de información encapsulados en ráfagas.

Es necesario mencionar que esta tecnología se encuentra actualmente en estado de desarrollo e investigación y al respecto se han realizado varias simulaciones y prototipos para evaluar su desempeño y adaptabilidad sobre el tráfico de Internet, encontrándose resultados interesantes y también ciertas restricciones especialmente en cuanto al tema de contenciones, debido a la limitación del *buffering* óptico propio de la inmadurez de la tecnología actual. Incluso se han desarrollado algunas soluciones comerciales que se expondrán más adelante.

Aunque la conmutación óptica de ráfagas parece ofrecer ventajas con respecto a la conmutación óptica de circuitos y la conmutación óptica de paquetes, también tiene aspectos tecnológicos y desafíos arquitectónicos propios, por lo cual, varios asuntos deben ser considerados antes de que esta tecnología pueda ser implementada en redes en producción. En particular, estos asuntos incluyen el ensamblado de ráfagas, esquemas de señalización, planificación de ráfagas y resolución de contenciones, que se abordarán más adelante.

Se requiere un esquema de ensamblado para determinar cómo se ensamblan los paquetes en ráfagas. Los aspectos incluyen principalmente cuándo ensamblar una ráfaga y cuántos paquetes añadir en una ráfaga. El esquema de ensamblado afecta a la longitud de la ráfaga así como también a la cantidad de tiempo que un paquete debe esperar antes de ser transmitido. Los esquemas de ensamblado basados en temporizador, longitud de ráfaga e híbridos se discuten en la sección 2.4.

Por otro lado, un esquema de señalización es necesario para la reserva de recursos y la configuración de los conmutadores para una ráfaga entrante. Los esquemas más comunes de

señalización en redes OBS son: TAG (*Tell-And-Go*), TAW (*Tell-And-Wait*) y JET (*Just-Enough-Time*), siendo éste último la alternativa más popular en OBS, superando a otras opciones [59]. Las técnicas de señalización para redes OBS se estudian en detalle en la sección 2.5.

Cuando un nodo intermedio recibe el BCP, elige un enlace de salida, y a continuación, mediante un algoritmo de planificación, escoge una longitud de onda de salida para la ráfaga asociada. Los investigadores han desarrollado varios esquemas de planificación de canal que tratan de maximizar su utilización, los mismos que se discuten con mayor detalle en la sección 2.6.

En los esquemas TAG y JET, el origen no espera recibir un acuse de recibo antes de enviar una ráfaga. Por lo tanto, es posible que las reservas no sean exitosas en algún nodo del camino, en cuyo caso, una ráfaga que está en tránsito experimentará contención. Dicho de otra manera, puede darse el caso de que dos o más ráfagas necesiten el mismo recurso en el mismo instante de tiempo, en cuyo caso se produce una contención. En las redes clásicas, este problema se solventa mediante el almacenamiento electrónico en una memoria habitualmente denominada *buffer*, hasta que el recurso quede libre. Sin embargo, cuando la transmisión se realiza por medios ópticos, el almacenamiento es más complicado y en vista de que la tecnología de almacenamiento óptico es todavía inmadura, y no existe el equivalente de una memoria RAM óptica, una solución a la contención es el empleo de líneas de retardo de fibra óptica (FDLs) que retrasan la ráfaga un tiempo determinado. Otras soluciones consisten en cambiar la longitud de onda de una de las ráfagas (de forma que puedan transitar simultáneamente por la misma fibra de salida), desviar una de las ráfagas por otro puerto de salida distinto al que le correspondería originalmente (enrutamiento por deflexión), o descartar la ráfaga cuando las técnicas de resolución de contenciones no son exitosas. Los esquemas de resolución de contenciones se discuten en la sección 2.7.

En caso de que no sea posible evitar una contención con los esquemas antes mencionados, las ráfagas se descartarían y en este punto se tendría como alternativas, que el tratamiento de dichas pérdidas sea llevado a cabo por los protocolos de capas superirores (p. ej. TCP) o por la propia red OBS; para el segundo caso se han popuesto ciertos mecanismos para la recuperación de pérdidas que se describen en la sección 2.8.

Por otro lado, en la sección 2.9 se presentan los principales prototipos o *"testbeds"* implementados a lolargo de tiempo, que constituyen los primeros pasos reales para la puesta en producción de esta tecnología.

1.3.4 Comparación de las técnicas de conmutación OXS

Para aclarar las características distintivas de los tres paradigmas de conmutación que se acaban de presentar (OCS, OPS y OBS), se realizará una comparación considerando los siguientes cinco criterios utilizados comúnmente en la literatura [24][59]: requerimientos de ancho de banda, latencia de configuración, velocidad de conmutación, complejidad de procesamiento y adaptabilidad de tráfico.

Requerimientos de ancho de banda.- Como se mencionó anteriormente, las redes OCS, permiten el establecimiento de un camino óptico de forma estática o dinámica utilizado longitudes de onda completamente dedicadas a lo largo de la ruta entre un par de nodos origen y destino, por lo cual, el ancho de banda sin uso en un camino óptico no puede ser empleado por otros nodos para el envío de datos, es decir, OCS no ofrece ganancia de multiplexación estadística. Por lo tanto, la utilización de ancho de banda es bastante baja en este tipo de redes. En contraste, las redes OPS y OBS permiten una utilización más eficiente del ancho de banda gracias a que emplean multiplexación estadística en el dominio óptico, permitiendo que el tráfico entre diferentes pares de nodos origen y destino pueda compartir el ancho de banda disponible sobre los distintos enlaces de la red.

Latencia de configuración.- En las redes OCS, es necesario un intercambio de mensajes de señalización en dos vías entre cada par de nodos origen y destino antes de establecer o liberar un camino óptico, conduciendo a largos tiempos de reconfiguración en comparación con OPS y OBS. En OPS, los datos se envían normalmente sin ninguna reserva previa, mientras que en OBS se requiere únicamente señalización en una vía antes de que los datos sean transmitidos, reduciendo así la latencia extremo a extremo, pero incrementando a su vez la probabilidad de bloqueo producto de las contenciones que se pueden presentar en la red. Por lo tanto, en OPS y OBS es fundamental el uso de esquemas de resolución de contenciones que permitan

garantizar la transferencia de datos a un nivel mínimo aceptable de probabilidad de pérdida de ráfagas.

Velocidad de conmutación.- En OPS se requieren conmutadores de muy alta velocidad para permitir la conmutación de los paquetes ópticos rápidamente hacia los distintos puertos de salida a medida que los paquetes arriban a la red. En OCS, debido a que la conmutación se realiza a nivel de longitud de onda y los caminos ópticos se establecen por períodos relativamente largos, la velocidad de conmutación requerida es baja. En el caso de las redes OBS, debido a la mayor granularidad de las ráfagas ópticas y a los tiempos de *offset* asociados, los requerimientos de conmutación se pueden considerar moderados. La Figura 1.32 ilustra las exigencias tecnológicas que requiere cada tipo de conmutación óptica, donde de acuerdo a la duración de las conexiones, los requerimientos de conmutación en los nodos son diferentes. Por lo tanto, existen diferentes tecnologías que proporcionan las capacidades de conmutación, las cuales se abordarán en capítulo II.

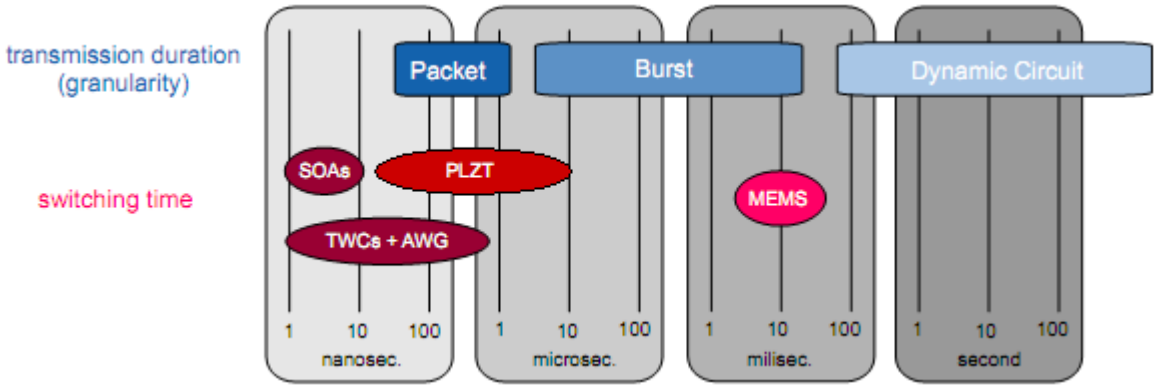


Figura 1.32 Información general sobre los principales parámetros que determinan la granularidad de circuitos/ráfagas/paquetes y la tecnología de conmutación requerida [66]

Complejidad de procesamiento.- Este criterio está estrechamente relacionado con la granularidad de conmutación. En redes OCS, como el tráfico no puede conmutar a una granularidad más baja que una longitud de onda y los caminos ópticos tienen una duración relativamente larga, el nivel de procesamiento es relativamente bajo en comparación con las redes OPS y OBS. En redes OPS, puesto que la entidad de conmutación es un paquete óptico

individual y la información de control está contenida en cada paquete, la complejidad de procesamiento es muy alta. En redes OBS, debido a que la entidad de conmutación es una ráfaga de datos individual que agrega múltiples paquetes de datos, la complejidad de procesamiento se encuentra en el medio de OCS y OPS.

Adaptabilidad del tráfico.- Como se mencionó anteriormente, OPS y OBS mejoran la utilización de los recursos de la red mediante la multiplexación estadística de flujos de tráfico, en particular, cuando éste presenta un comportamiento altamente variable dada su naturaleza a ráfagas, permitiendo una mejor adaptación al tráfico de datos dominante en las redes actuales, en comparación con las redes OCS que son poco flexibles y eficientes para adaptarse a las dinámicas de este patrón de tráfico, debido a su alta latencia de configuración y baja granularidad de conmutación a nivel de longitud de onda.

Las diferencias y similitudes mencionadas se resumen en la Tabla 1.4, donde se evidencia las ventajas de la tecnología OBS, especialmente si se consideran conjuntamente con las innovaciones tecnológicas desarrolladas en el campo óptico que se discutirán más adelante. Esta tabla muestra también, porqué OBS se presenta usualmente como una tecnología que combina las ventajas tanto de OCS como de OPS, mientras que evita sus deficiencias.

Tabla 1.4 Comparación de los diferentes tipos de conmutación óptica [24]

Tipo de conmutación óptica	Utilización de ancho de banda	Latencia de establecimiento	Velocidad de conmutación	Complejidad de procesamiento	Adaptabilidad del tráfico
OCS	Baja	Alta	Lenta	Baja	Baja
OPS	Alta	N/A	Rápida	Alta	Alta
OBS	Alta	N/A	Media	Media	Alta

CAPÍTULO II CONMUTACIÓN ÓPTICA DE RÁFAGAS

La conmutación óptica de ráfagas (Qiao y Yoo, 1999; Turner, 1999) [22][110] es una de las tecnologías más prometedoras para transmitir el tráfico a ráfagas sobre una infraestructura completamente óptica. Basado en un concepto de conmutación de ráfagas inicialmente introducido en Amstutz (1983) y Kulzer y Montgomery (1984), su desarrollo en comunicaciones ópticas ha surgido como una alternativa a una operación de OCS poco flexible y a una solución OPS tecnológicamente inmadura todavía. La tecnología OBS se soporta en los avances de varios elementos clave de la red, incluyendo conmutadores totalmente ópticos, receptores en modo de ráfaga óptica y convertidores de longitudes de onda [59]. Este tipo de conmutación ha recibido considerable atención en los últimos años, y se han propuesto y analizado diferentes soluciones en un intento de mejorar su rendimiento, incluyendo técnicas de ensamblado de ráfagas, esquemas de planificación de canal, métodos de resolución de contenciones y provisión de QoS [22]. Aunque aún se encuentra en desarrollo (no ha sido estandarizada todavía), OBS está actualmente siendo considerada una de las arquitecturas todo ópticas más prometedoras para el Internet de próxima generación [10]. Entre otras razones, el éxito de OBS se soporta en dos características fundamentales: se basa en esquemas de señalización de una vía para disminuir el retardo asociado con el tiempo de ida y vuelta, y en su pequeño encabezado de control para una gran cantidad de datos de carga útil.

En este capítulo se presenta una descripción de la arquitectura de los nodos de conmutación óptica de ráfagas, incluyendo sus diferentes componentes funcionales y ciertas tecnologías clave que han permitido la definición de este nuevo paradigma propuesto para el Internet Óptico de próxima generación. Se discuten además, distintos asuntos relacionados con la capa física que pueden afectar el desempeño de este tipo de redes, así como también sus principales características, protocolos, beneficios y algunos de sus desafíos. Por último, se realiza un recorrido por los *testbeds* de OBS que han servido de prueba de concepto y que han demostrado su viabilidad, finalizando con los primeros productos comerciales basados en la conmutación óptica de ráfagas.

2.1 Arquitectura de la red OBS [59][75,78][111]

Una red OBS está compuesta de nodos de conmutación óptica de ráfagas interconectados mediante enlaces de fibra óptica, que soportan múltiples canales de longitud de onda utilizando WDM. Los nodos en una red OBS se pueden clasificar en nodos frontera (*edge*) de ingreso o egreso y nodos de *core*, como se expuso anteriormente. Los nodos de *edge* conocidos también como enrutadores de borde, tal y como su nombre lo indica, se encuentran en el límite de la red OBS, que actúa como interfaz para proporcionar las funcionalidades necesarias para la interoperación entre el mundo externo (redes de clientes) y la red OBS. Por otro lado, los nodos de *core* referidos también como enrutadores centrales o nodos intermedios, componen el *core* interno de la red OBS.

En la frontera de la red se pueden distinguir claramente dos tipos de funcionalidades provistas en los nodos de *edge*, que dan lugar a que se los pueda clasificar para efectos didácticos como nodos de ingreso y nodos de egreso, que actúan como interfaz de comunicación entre las redes electrónicas convencionales de clientes y el dominio OBS.

El nodo de ingreso es responsable principalmente de ensamblar los paquetes provenientes del mundo exterior en bloques de mayor tamaño denominados ráfagas, y planificarlas para su envío a través de canales ópticos de salida. Antes de transmitir una ráfaga, el nodo de ingreso genera un paquete de control (BHP, *Burst Header Packet*), que permite la reserva de recursos a lo largo de la red y contiene toda la información necesaria para poder transmitir la ráfaga correctamente, la cual se envía luego del paquete de control después de un lapso de tiempo predefinido denominado *offset*. Por otro lado, el nodo de *core* se encarga principalmente de conmutar las ráfagas que llegan a un puerto de entrada, hacia un puerto de salida en base a la información proporcionada por los paquetes de control y de controlar las contenciones de ráfagas que se transmiten en el dominio completamente óptico a lo largo del *core* de la red, sin ningún tipo de almacenamiento en los nodos intermedios. Finalmente, el nodo de egreso se encarga de recibir las ráfagas de datos y desensamblarlas en sus paquetes originales, para posteriormente reenviarlos hacia las redes cliente de destino.

La Figura 2.1 presenta un diagrama de bloques que resume las distintas funcionalidades que se llevan a cabo dentro de una red óptica conmutada por ráfagas, haciendo referencia a cada uno de los diferentes nodos involucrados.

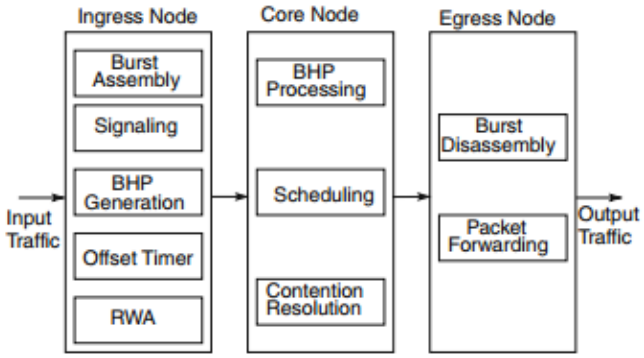


Figura 2.1 Diagrama funcional de OBS [111]

La arquitectura del enrutador de *edge* (ver Figura 2.2) consta de un Módulo de Enrutamiento (RM, *Routing Module*), un ensamblador de ráfagas y un planificador, que cumplen distintas funciones como pre-clasificación de paquetes, almacenamiento de paquetes en *buffers* electrónicos, ensamblado de paquetes en ráfagas y desensamblado de ráfagas.

El módulo de enrutamiento selecciona el punto de salida apropiado para cada paquete que lo envía al módulo ensamblador de ráfagas correspondiente, el cual se encarga de agregar en ráfagas ópticas los paquetes destinados a un mismo enrutador de egreso, para enviarlas dentro de la red. El proceso de conformación de ráfagas se realiza de acuerdo a distintas políticas de ensamblado basadas en un umbral establecido de forma predefinida o adaptativa, en función de uno de los siguientes criterios: tamaño máximo de ráfaga, temporizador o una mezcla de ambos (esquema híbrido).

El módulo ensamblador de ráfagas, incluye una cola de paquetes independiente para cada clase de tráfico. El planificador se encarga de la creación de ráfagas en base al algoritmo de ensamblado seleccionado y las transmite a través del puerto de salida correspondiente. En el enrutador de egreso, un módulo de desensamblado de ráfagas las desmonta en sus paquetes originales y los envía fuera del dominio OBS hacia el equipo destino, propiedad del cliente,

que posteriormente se encargará de transportar la información hacia las capas superiores de la red.

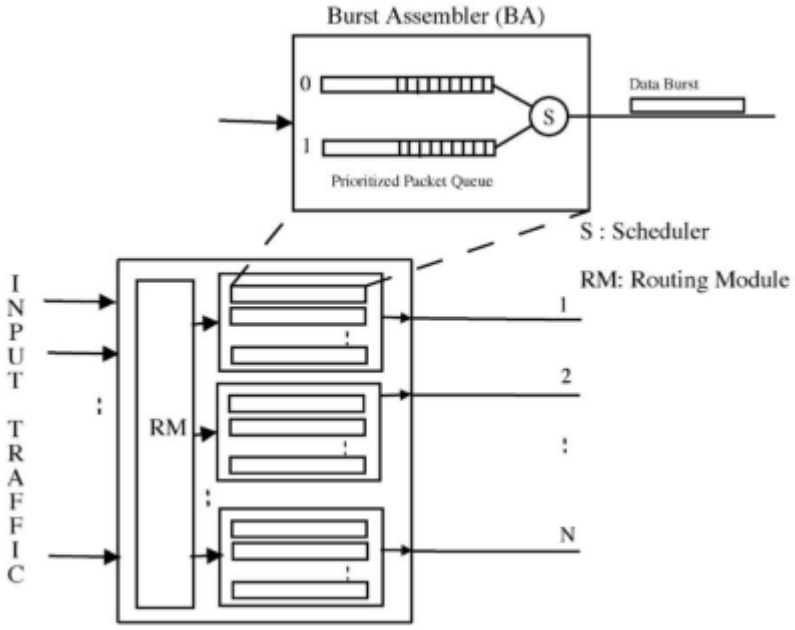


Figura 2.2 Arquitectura de un enrutador de borde [59]

La arquitectura del enrutador de *core* (ver Figura 2.3) se compone de una Unidad de Control de Conmutación (SCU, *Switch Control Unit*) y un conmutador óptico (OXC, *Optical Cross Connect*). La SCU crea y mantiene una tabla de reenvío y es responsable de la configuración del OXC. Cuando la SCU recibe un paquete de control, consulta los procesadores de enrutamiento y señalización para identificar el puerto de salida en base al destino. Si el puerto de salida está disponible cuando llega la ráfaga de datos, el OXC se configura de manera que permita el paso de dicha ráfaga; por el contrario, si el puerto no está disponible, el OXC se configura en función de la política de resolución de contenciones implementada. En el caso de que una ráfaga de datos ingrese al OXC antes que su paquete de control, simplemente se descarta. Este problema es referido como llegada anticipada de ráfagas, que podría suceder debido a una insuficiencia en el tiempo de *offset*.

Adicional a lo anterior, la SCU tiene también otras responsabilidades importantes que incluyen el procesamiento de la cabecera, planificación de ráfagas, detección y resolución de contenciones, reescritura de la cabecera, y control de los convertidores de longitudes de onda.

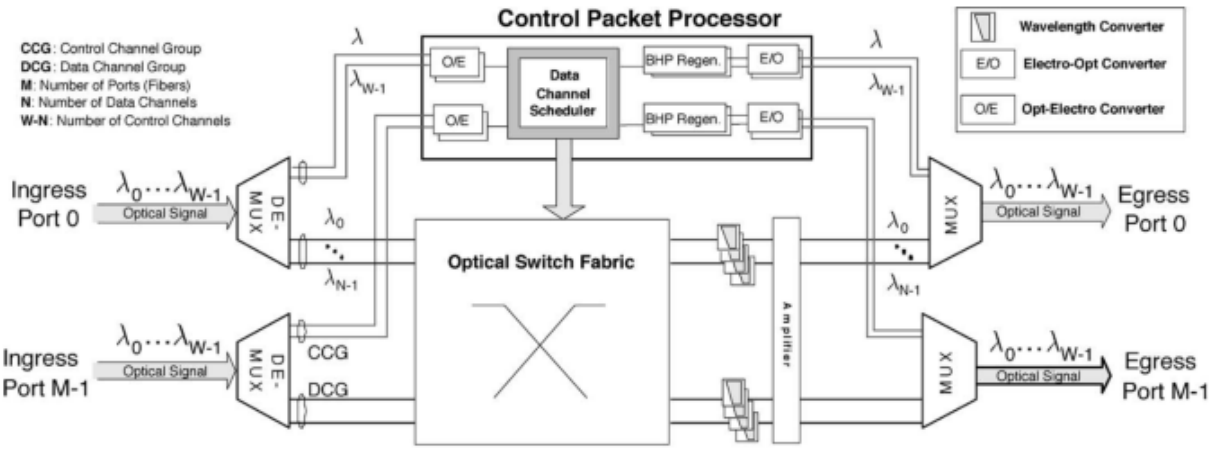


Figura 2.3 Arquitectura de un enrutador de core [59]

Para proveer la funcionalidad básica de la conmutación óptica de ráfagas, se requieren varias tecnologías de dispositivos ópticos. En los nodos centrales y de frontera, el OXC debe ser implementado utilizando una matriz de conmutación óptica de alta velocidad. Los nodos frontera también deben disponer de receptores en modo ráfaga que sean capaces de adquirir la señal de una ráfaga entrante rápidamente. Cada nodo también debería tener alguna forma de conversión de longitud de onda con el fin de reducir la contención en los enlaces de salida.

OBS introduce una serie de aspectos de diseño de red específicos que deben ser abordados adecuadamente con el fin de implementar las funciones antes mencionadas, para lo cual se requiere una capa de control de acceso al medio MAC (*Medium Access Control*) entre la capa IP y la capa óptica [75,78]. La Figura 2.4 representa los bloques funcionales necesarios tanto en la capa MAC OBS, como en la capa óptica para la implementación de redes OBS.

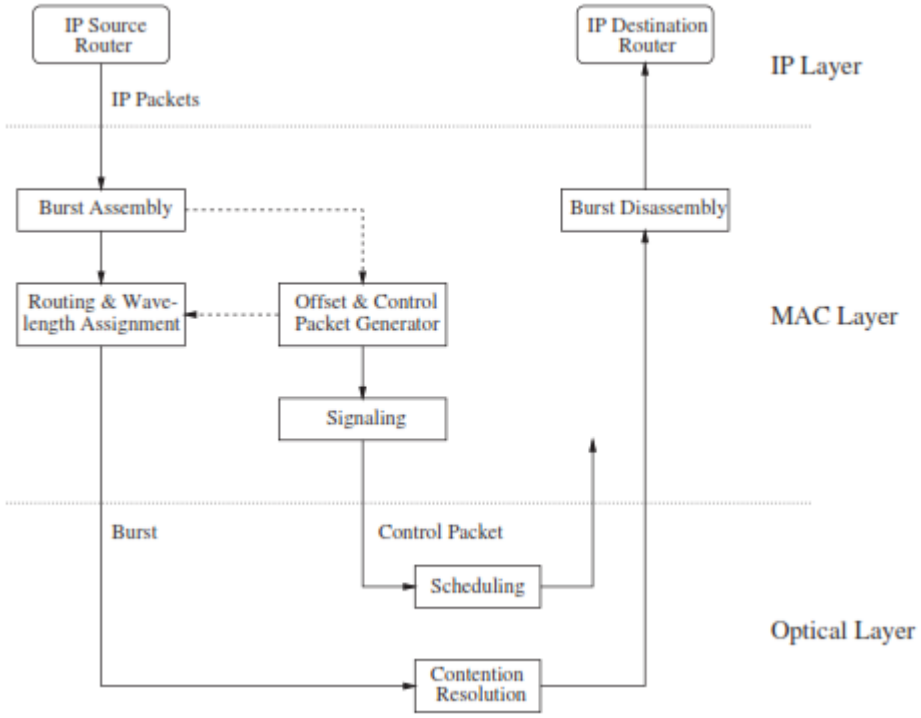


Figura 2.4 Diagrama de bloques de redes OBS que consisten de capas IP, MAC y óptica [75]

Los bloques funcionales corresponden a las funciones antes mencionadas que deben ser ejecutadas por los nodos OBS de borde y los nodos OBS de *core*, en la capa MAC y capa óptica respectivamente. En la capa MAC OBS, los nodos origen y destino localizados en la frontera de la red OBS realizan las funciones de ensamblado/desensamblado de ráfagas, cálculo del *offset*, generación del BHP, enrutamiento y asignación de longitud de onda (RWA), y señalización. En la capa óptica, los nodos OBS de *core* intermedios son los responsables de la planificación y de la resolución de contenciones de las ráfagas en tránsito. La capa MAC OBS junto con la capa óptica subyacente, ofrecen servicios que garantizan cierta probabilidad de bloqueo de ráfagas a los enrutadores IP de clientes de la capa superior [75]. A partir de la sección 2.4, se describen con mayor detalle los bloques funcionales de la capa OBS MAC y la capa óptica de la Figura 2.4, y se provee una discusión a profundidad del estado del arte de los algoritmos y las técnicas propuestas para redes OBS.

El enrutamiento en redes OBS se puede realizar sobre una base salto por salto utilizando rápidos algoritmos de búsqueda en una tabla de enrutamiento en los nodos intermedios OBS,

de manera similar a las redes IP, o mediante la implementación de protocolos de enrutamiento de GMPLS (*Generalized Multi-Protocol Label Switching*) para calcular rutas explícitas o basadas en restricciones en los nodos OBS de borde. A lo largo de la ruta seleccionada, a cada enlace se le debe asignar una longitud de onda en la cual las ráfagas son transportadas. En redes OBS, es posible la asignación de longitudes de onda con y sin conversión de longitud de onda en los nodos intermedios [75].

2.2 Tecnologías habilitantes [2,5,7,9][15,18][29][36][41,45][50,59][60,61] [63][70,78][86][99][104][121,123][131]

La definición de la conmutación óptica de ráfagas se basa en el desarrollo exitoso de varias tecnologías clave, que incluyen conmutadores completamente ópticos, receptores en modo ráfaga, y convertidores ópticos de longitud de onda, pero independientemente del tipo de tecnología que se utilice finalmente en el diseño de redes ópticas conmutadas por ráfagas, se deben tener en cuenta las limitaciones de la capa física impuestas por las tecnologías de los dispositivos y componentes seleccionados.

En los últimos años, se ha centrado mucha investigación sobre la tecnología OBS que es adecuada para el tráfico a ráfagas, y si bien su avance ha tenido un importante progreso, se requiere mayor investigación especialmente en cuanto a temas como la optimización a nivel de enrutamiento, capacidad de almacenamiento óptico, mecanismos para provisión de calidad de servicio, esquemas para enrutamiento redundante, etc., antes de que dicha tecnología esté completamente desplegada y estandarizada en sistemas en producción, que se prevé sea una realidad en el corto o mediano plazo.

2.2.1 Tecnología de conmutación óptica

Los conmutadores ópticos pueden operar por medio de métodos mecánicos, tal como el desplazamiento físico de una fibra óptica para accionar una o más fibras alternativas de salida

o por medio de efectos que ocurren en algún material bajo ciertas condiciones. A partir de esto, se han estudiado y desarrollado varios tipos de conmutadores, que se agrupan principalmente en dos clases a la hora de considerar su tecnología de fabricación base: espacio libre y guía de onda. La primera clase comprende dispositivos que realizan la conmutación al trabajar en la óptica de espacio libre con haces colimados; ejemplos de esta tecnología son: MEMS (*Micro Electrical Mechanical Systems*), cristales líquidos, dispositivos basados en electro-holografía, o ESBG (*Electrically Switchable Bragg Gratings*). En la segunda clase, la conmutación se lleva a cabo recurriendo a efectos que ocurren en las guías de onda; ejemplos de éstos son: termo-ópticos, electro-ópticos, acusto-ópticos, basados en gel/aceite, amplificador óptico de semiconductor (SOA, *Semiconductor Optical Amplifier*), y dispositivos ferromagnéticos [5]. Estas dos clases se presentan esquemáticamente en la Figura 2.5.

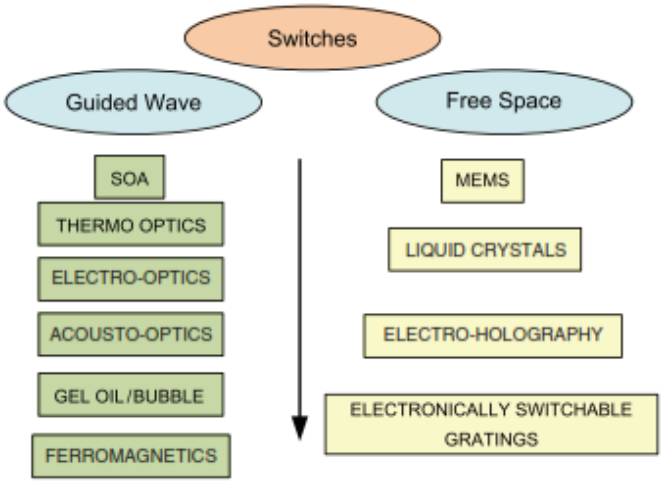


Figura 2.5 Tecnologías de conmutación óptica: conmutadores basados en guía de onda y conmutadores basados en el espacio libre [5]

Aunque los requerimientos de conmutación para OBS se pueden considerar moderados debido a que el tiempo de reconfiguración de los conmutadores es menor que la longitud de las ráfagas de datos, se debe tener en cuenta que la demanda de nuevos servicios (p. ej. servicios grid: edición de video en alta resolución, renderizado en tiempo real, *e-health*, TV interactiva en HD, etc.) trae nuevas restricciones en cuanto a la velocidad de conmutación y requerimientos tecnológicos, que llegan a ser particularmente importantes al considerar la velocidad de

transmisión. Por ejemplo, un conmutador basado en sistemas micro-electromecánicos MEMS (*Micro Electrical Mechanical Systems*) con un tiempo de conmutación de alrededor de 20 ms proporciona un factor de utilización del 93.7% en un sistema de 10 Gbps, donde la duración de la ráfaga es de 300 ms. Si la misma ráfaga fuera transmitida a 160 Gbps, entonces la duración de la ráfaga sería de 18.75 ms y el enrutamiento a través del mismo conmutador disminuiría el *throughput* del sistema a menos del 50% [5]. Esta reducción en la utilización se vuelve más severa para ráfagas más pequeñas con cortos tiempos de *offset*. Por esta razón, el desarrollo de tecnologías de conmutación rápida es esencial para las futuras redes OBS de alta velocidad con soporte para tal evolución de servicios en demanda de ancho de banda.

Por lo general, la tecnología de guía de onda en comparación con la de espacio libre presenta mayores pérdidas, pero velocidades de conmutación más altas. Por lo tanto, cuando se escoge una tecnología de conmutación, se debe llegar a un compromiso entre estos dos parámetros. Los conmutadores ópticos lentos, como los que emplean fibras en movimiento, pueden ser utilizados para el enrutamiento alternativo de una ruta de transmisión óptica, tal como la conmutación de tráfico de la fibra de trabajo a la fibra de protección ante la eventualidad de una falla, conocido como restauración y protección. Conmutadores ópticos rápidos, como los que utilizan efectos electro-ópticos o magneto-ópticos, se pueden utilizar para llevar a cabo las operaciones lógicas y de conmutación en el orden del tamaño de un bit como el caso de la multiplexación/demultiplexación por división de tiempo óptica, OTDM (*Optical Time Division* Multiplexing) [5].

Las técnicas que permiten la conmutación en el dominio completamente óptico son muy importantes en el desarrollo de las matrices de conmutación, que constituyen un elemento fundamental en los nodos de *core* dentro de una red OBS, por lo cual, se deben tener en cuenta distintos parámetros como el tiempo de reconfiguración del conmutador, bajas pérdidas de inserción, dependencia con la polarización (PDL[15] y PMD), bajo consumo de energía, facilidad de interconexión, fiabilidad, y escalabilidad, que pueden limitar el desempeño global de la red. Se han desarrollado muchas tecnologías a lo largo de los años, cada una con sus

[15] Si la pérdida del conmutador no es igual para ambos estados de polarización de la señal óptica, se dice que el conmutador tiene una pérdida dependiente de la polarización (PDL, *Polarization-dependent loss*) [5].

propias limitaciones y beneficios, adaptándose en mayor o menor medida a cada una de las funciones que realizan, tal y como se muestra en la Tabla 2.1.

Tabla 2.1 Resumen de las principales características de algunas de las tecnologías disponibles [5]

Tecnología	Ventajas	Desventajas	Aplicaciones
Fibras móviles	Bajos niveles de pérdidas y diafonía	Largos tiempos de conmutación y estabilización, baja escalabilidad	Protección, OADMs
MEMS	Tamaño reducido	Baja confiabilidad debido a sus partes móviles	OXC de gran escala
Burbuja	Fácil de integrar	Largos tiempos de conmutación (bajo 10 ms)	OADM, OXC de escala media
Termo-óptico	Fácil de integrar	Largos tiempos de conmutación, alto nivel de pérdidas y diafonía, alto consumo de energía	Protección/restauración, OADM, OXC de escala media
Cristal líquido	Buena confiabilidad	Dependiente de la temperatura, bajos tiempos de conmutación (ms)	Protección/restauración, OXC de baja escala y OADM
Electro-óptico	Conmutación rápida	Pérdidas medias y alta diafonía Dependiente de la polarización y baja escalabilidad	Protección/restauración, OADM, paquetes/ráfagas
Acusto-óptico	Conmutación flexible	Pérdidas medias y complejidad	Protección/restauración, OXC de baja escala y OADM
Electro-holografía	Altamente flexible y posible de incorporar en la demultiplexación de longitudes de onda	Pérdidas medias y alta potencia	Protección/restauración, OXC de baja escala y OADM
SOA	Rápida conmutación, ganancia de amplificación	Adición de ruido, moderadamente costosos	Protección/restauración, OADM, conmutación de paquetes o ráfagas

A continuación se describen las tecnologías de conmutación más comunes, algunas de las cuales se han utilizado en la implementación de ciertos prototipos desarrollados para demostrar la viabilidad tecnológica de OBS.

Opto-mecánicos.- La tecnología opto-mecánica fue la primera tecnología disponible comercialmente para la conmutación óptica [86], principalmente para las funciones básicas de protección y restauración. La función de conmutación se realiza por algún medio mecánico, que puede incluir prismas, espejos y acopladores direccionales. Los conmutadores mecánicos presentan bajos niveles en cuanto a pérdidas de inserción, pérdidas dependientes de la polarización y diafonía (*crosstalk*), y su costo de fabricación es bajo. Sin embargo, sus principales desventajas son su tiempo de conmutación en el orden de ms, la cual puede no ser aceptable para algunos tipos de aplicaciones, y su falta de escalabilidad. Los conmutadores opto-mecánicos son utilizados principalmente en aplicaciones de protección de fibra y adición/extracción de longitudes de onda con un número muy reducido de puertos [5].

Sistemas Micro-electromecánicos (MEMS).- Esta tecnología hace referencia al desarrollo de pequeñas máquinas que combinan piezas mecánicas, sensores, actuadores, dispositivos electrónicos y ópticos, que se fabrican normalmente utilizando sustratos de silicio a una escala micrométrica; tuvo sus primeras aplicaciones comerciales en pantallas digitales que empleaban arreglos de diminutos espejos y en acelerómetros para sensores de airbags en automóviles.

Los sistemas MEMS se utilizan actualmente en un amplio ámbito de campos que van desde teléfonos móviles, tablets y pico-proyectores hasta monitores médicos, sistemas de seguridad en automóviles, plataformas de juegos y robótica, incluyendo también una serie de aplicaciones en el campo de las comunicaciones ópticas, donde es una de las tecnologías de dispositivos más maduras para realizar la conmutación completamente óptica y bajo este contexto los sistemas MEMS se refieren a diminutos espejos con dimensiones del orden de cientos de micrómetros a milímetros, organizados normalmente en un arreglo o matriz de espejos encargados de transferir las señales ópticas en el espacio libre dentro del conmutador, redirigiendo la luz desde un puerto de entrada determinado hacia el puerto de salida deseado. Los espejos cambian de una posición a otra utilizando técnicas de variación electrónica, tales como métodos electromagnéticos, electrostáticos o piezoeléctricos. Su tamaño, posición y

ángulo de inclinación se establecen durante el proceso fotolitográfico. Las principales ventajas que ofrecen son su pequeño tamaño, bajo consumo de potencia, fabricación por lotes y alta velocidad de conmutación con tiempos de respuesta en las proximidades de 10^{-6} s [61].

El tipo más sencillo de conmutador MEMS ilustrado en la Figura 2.6, se denomina 2D y consta de espejos organizados en una estructura del tipo barra cruzada (*cross-bar*) que pueden conformar arreglos de NxN, aunque en la práctica sus dimensiones están restringidas al tamaño del sustrato de silicio, que típicamente permite un máximo de 32x32. Como se puede apreciar en la Figura 2.6, cada espejo puede moverse de forma independiente pero únicamente entre dos posiciones. En la primera, no existe una desviación de la luz debido a que el espejo se encuentra abajo (inactivo), mientras que en la segunda, el espejo se ubica en una posición vertical (activo) permitiendo la redirección de la luz al punto de interés. La manipulación del movimiento de los espejos, limitada a estos dos estados de funcionamiento, se consigue fácilmente utilizando un circuito de control digital.

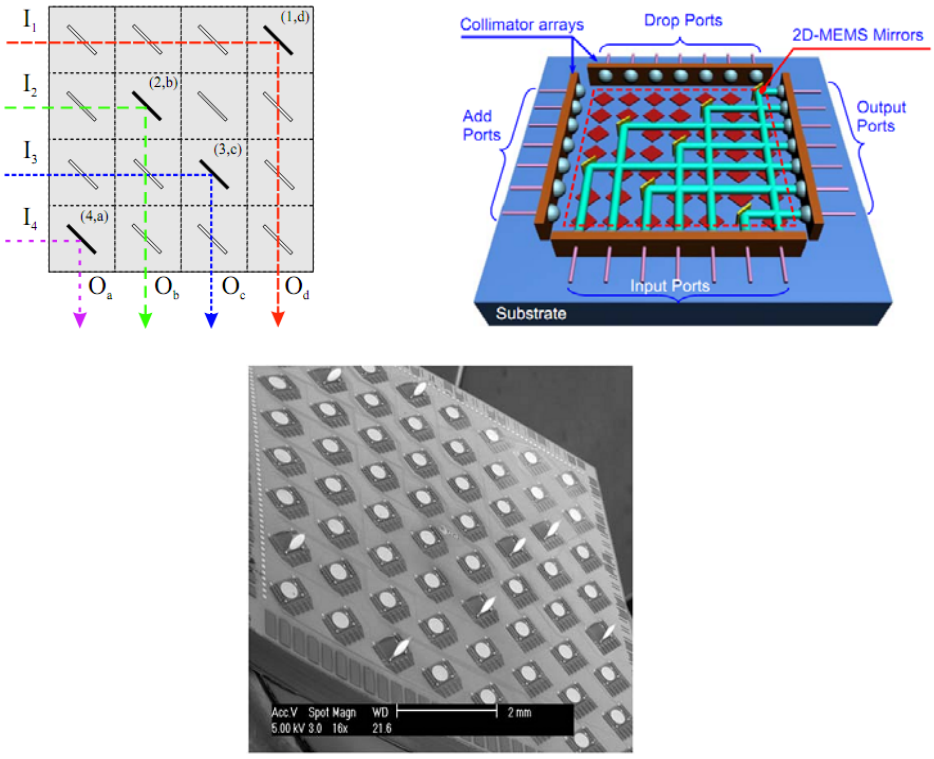


Figura 2.6 Conmutador óptico MEMS 2D NxN con estructura cross-bar [9][68]

Otro tipo de conmutador óptico es el denominado 3D, que incluye un arreglo de espejos organizados en forma de matriz, como se puede observar en las Figuras 2.7 y 2.8 donde cada espejo está conectado y rota sobre un marco interior, que a su vez está conectado y puede rotar sobre un marco exterior, es decir existen dos ejes independientes de giro que permiten un movimiento rotacional del espejo, que puede detenerse en múltiples posiciones, que se traducen en un rango continuo de deflexiones angulares. El control de movimiento de estos espejos se puede realizar mediante una señal analógica, que incluye sofisticados mecanismos de servo control, necesarios para desplazar los espejos hasta su posición correcta y mantenerlos en ella.

Los MEMS 3D ofrecen mejores prestaciones que los anteriores dado que son compactos, con muy buenas propiedades ópticas (bajas pérdidas, pérdidas uniformes y dispersión despreciable) y pueden consumir muy poca potencia [36]. Sin embargo, debido a que su control llega a ser complejo si N es grande, los asuntos de confiabilidad y estabilidad son las principales preocupaciones de este tipo de conmutadores ópticos [70].

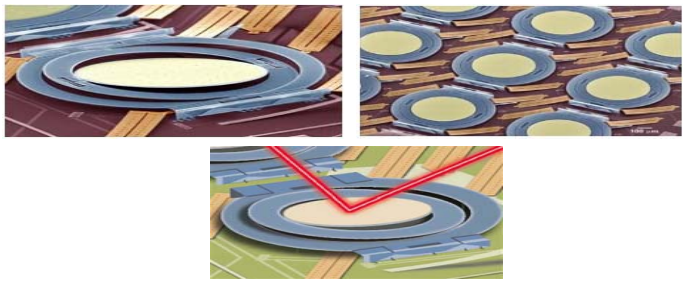


Figura 2.7 Espejos MEMS utilizados en un conmutador óptico [45]

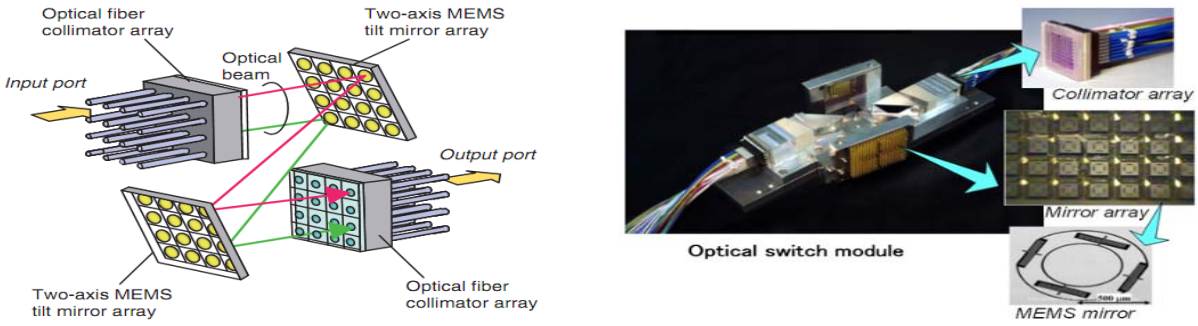


Figura 2.8 Estructura básica de un conmutador óptico MEMS 3D [50][121]

Cristal líquido (LCOS).- Los conmutadores basados en cristal líquido LC (*Liquid Crystal*), siendo el más popular el LCOS (*Liquid Crystal on Silicon*), emplean la óptica de espacio libre para conmutar. Este tipo de conmutadores se basan en el principio por el cual, la luz al reflejarse en un cristal líquido, sufre una rotación en su polarización. Esto es posible, aplicando un voltaje adecuado a través de la celda llena con cristal líquido, para cambiar la orientación de sus moléculas. La estructura de un conmutador 1x2 de cristal líquido se ilustra en la Figura 2.9, donde un splitter de haz de polarización, PBS (*Polarization Beam Splitter*) [15], divide la luz incidente en dos componentes polarizadas ortogonalmente entre sí, que son enviadas en una dirección diferente. En la trayectoria de cada una de los componentes se sitúa una celda de cristal líquido controlada mediante un voltaje, para decidir si se rota o no la polarización de la luz que se refleja. Finalmente, un combinador de haz de polarización, se encargará de incorporar las dos componentes al puerto de salida deseado, según la polarización de los haces de luz que lleguen, realizando así la conmutación.

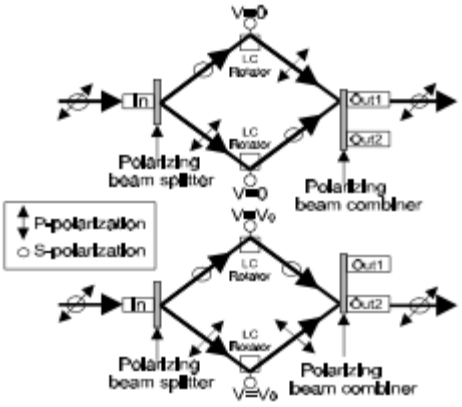


Figura 2.9 Esquema de un conmutador óptico 1x2 de cristal líquido [15]

Por tanto, para manejar un conmutador de este tipo se requiere de un nivel de voltaje que cambie cuando sea necesario conmutar. Al no tener partes movibles, como los MEMS, son más sencillos y duraderos. La tecnología LCOS está muy madura y se emplea actualmente en ROADMs (*Reconfigurable Add/Drop Multiplexers*) comerciales.

Amplificador Óptico de Semiconductor (SOA, *Semiconductor Optical Amplifier*).- Una tecnología que ofrece tiempos de conmutación más rápidos es el conmutador de compuertas basado en amplificador óptico de semiconductor (SOA). El diagrama básico de un conmutador SOA se muestra en la Figura 2.10. La luz que llega a una entrada determinada se difunde a múltiples SOAs mediante un acoplador óptico, los cuales actúan como compuertas que se pueden activar o desactivar dependiendo de la variación de corriente de polarización del dispositivo. Si la corriente de polarización se reduce, el dispositivo absorbe la señal y la bloquea (estado OFF); mientras que si la corriente de bombeo es suficientemente grande, las señales de entrada son amplificadas y transmitidas (estado ON) hacia la salida correspondiente.

Las ventajas de los conmutadores basados en SOA incluyen reducidos tiempos de conmutación en el orden de 1 ns, y la posibilidad de *multicasting* de una señal a varias salidas que se consigue estableciendo varios SOAs en estado ON. Sus pérdidas son muy bajas e incluso nulas (hasta pueden presentar ganancia) dado que los amplificadores compensan las pérdidas introducidas por divisores y combinadores de señal.

Una de sus principales limitaciones se encuentra en la adición de ruido debido al efecto de emisión espontánea, ASE (*Amplified Spontaneous Emission*), lo cual se debe tener en cuenta en el caso de que una señal deba atravesar una serie de estos dispositivos. Otra desventaja es que los acopladores reducen la potencia de la señal, limitando posiblemente las distancias que las señales pueden atravesar. Además, los dispositivos SOA son costosos y tienen alta sensibilidad a la polarización [59].

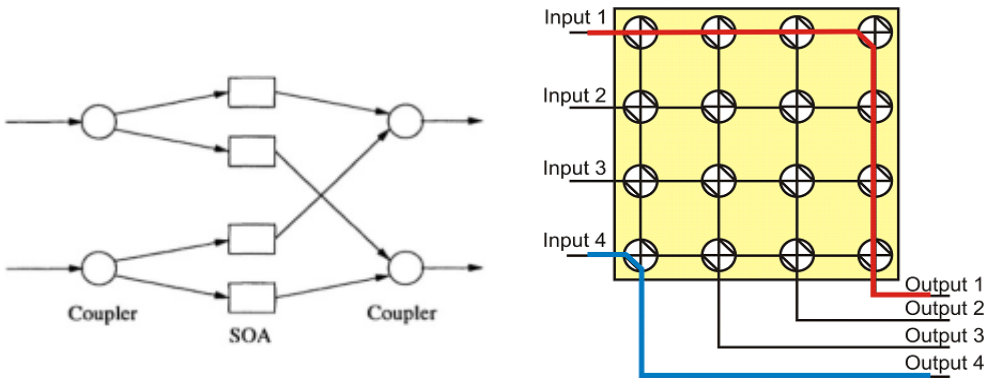


Figura 2.10 Conmutador con amplificador óptico de semiconductor (SOA) [59][18]

Electro-ópticos.- En algunos cristales, el índice de refracción es proporcional a la intensidad del campo eléctrico aplicado. Esto se conoce como el efecto lineal electro-óptico [123], que ofrece un medio para controlar la intensidad o la fase de la luz que se propaga a través del material. Existen varios tipos de elementos donde este efecto ha sido explorado, por ejemplo, materiales de estado sólido como el Niobato de Litio (LiNbO3), polímeros y PLZT (*Lead Zinconate Titane*). Dependiendo del material utilizado las condiciones generales difieren. Así, los basados en LiNbO3 se utilizan para moduladores rápidos por su tiempo de respuesta de picosegundos, mientras que los basados en PLZT alcanzan tiempos de conmutación en el orden de nanosegundos, y son de gran interés debido a que presentan bajas pérdidas y mayor efecto electro-óptico, que resultan en dispositivos de mayor eficiencia. Además, debido a su estructura, presentan una menor dependencia a la polarización. El PLZT es el material más popular en la actualidad para este tipo de conmutadores, debido a su facilidad de fabricación y costo. En cuanto a los polímeros, todavía queda un largo camino por recorrer antes de que lleguen a ser una alternativa clara [5].

Termo-ópticos.- El efecto termo-óptico es similar al electro-óptico, con la diferencia de que el cambio en el índice de refracción del material, se consigue mediante la variación de la temperatura en lugar del campo eléctrico. Debido a que la velocidad y estabilización de un flujo térmico para cambiar la temperatura de un dispositivo (que depende del tipo de material) es relativamente lenta, estos conmutadores son típicamente más lentos que sus contrapartes electro-ópticos. Los tiempos de conmutación están en el orden de decenas de milisegundos en el caso del silicio, y unos pocos milisegundos en el caso de los polímeros [5]. La velocidad de conmutación depende de cuán rápido pueden ser calentados los materiales, por ejemplo, los polímeros son más rápidos (pocos milisegundos) que el sílice (decenas de milisegundos). Además, la potencia necesaria para calentar las guías de onda con el fin de lograr suficiente cambio en el índice de refracción es altamente dependiente del tipo de material utilizado (pocos milivatios para polímeros y centenas de milivatios para el sílice) [5].

Este tipo de conmutadores pueden fabricarse utilizando tecnología de circuito planar de ondas de luz, PLC (*Planar Lightwave Circuit*)[16]. Mediante el uso de micro calentadores, se pueden inducir gradientes de temperatura dentro de estas estructuras de guías de onda, lo cual, es utilizado a su vez para realizar la conmutación. Existen dos tipos de conmutadores termo-ópticos: interferométricos y conmutadores ópticos digitales. Los primeros se basan usualmente en interferómetros de Mach-Zehnder (MZI, *Mach–Zehnder Interferometer*), donde el calentamiento de uno de sus brazos ocasiona una variación en su índice de refracción, que a su vez cambia la diferencia relativa de fase entre los dos brazos, provocando así la conmutación, dado que la interferencia puede ser constructiva o destructiva.

Por otro lado, los conmutadores ópticos digitales son más robustos debido a su respuesta escalonada, es decir, si se aplica más potencia al calentador, el conmutador permanece en el mismo estado, ya sea "activado" o "desactivado".

TWCs + AWG.- El convertidor de longitud de onda sintonizable, TWC (*Tunable Wavelength Converter*) en conjunto con el arreglo de rejillas de guías de onda, AWG (*Arrayed Waveguide Grating*) forman el equivalente de una funcionalidad de conmutación espacial, donde el camino a través del AWG se determina por la longitud de onda en su puerto de entrada.

La combinación de estas tecnologías permite alcanzar tiempos de conmutación en el orden de nanosegundos, y podría ser utilizada en redes OPS, donde la unidad control y procesamiento de la cabecera establece la señal eléctrica para que el TWC cambie la longitud de onda de la carga útil del paquete óptico a aquella requerida para asegurar que salga del AWG por el puerto especificado por la cabecera del paquete [61].

En [131] se presenta un estudio realizado sobre los elementos de conmutación ópticos más adecuados para un enrutador OBS/OPS comparando los tiempos de conmutación y su escalabilidad. La Tabla 2.2 muestra un resumen de estos resultados, completados con datos de la recopilación de tecnologías de conmutación realizada en [61].

[16] Tecnología utilizada para dispositivos ópticos integrados que son completamente pasivos (excepto los termo-ópticos).

Tabla 2.2 Comparativa de tecnologías de conmutación fotónica [61]

Tecnología	Tiempo de conmutación	Escalabilidad	Diafonía	PDL (*Polarization-dependent loss*)
Conmutador opto-mecánico	~ 4 ms	16x16	< -55 dB	< 0.1 dB
MEMS ópticos	3D ~ 10 ms; 2D ~ 3 ms	3D: 1000x1000; 2D: 32x32	< -45 dB	< 0.5 dB
PLC termo-ópticos	~ 3 ms	4x4	< -35 dB	< 0.5 dB
Electro-ópticos (PLZT)	~ 20 ns	4x4	< -25 dB	< 1 dB
Conmutador de enrutamiento por longitud (p. ej. AWG y TWC λ)	~ 1 ns	65536x65536	< -35 dB	Depende de la conversión de λ
Semiconductor *Optical Phase Array*[17]	~ 30 ns	64x64	< -25 dB	< 3 dB
SOA *Broadcast and select*	~ 1 ns	32x32	< -45 dB	Estricto
SOA *Cross-point*	~ 1 ns	4x4	< -45 dB	< 3 dB

Algunas de las tecnologías mencionadas se han usado en demostradores de OBS, como se verá más adelante en la sección 2.9 donde se presenta una recopilación de estos *testbeds*. Por ejemplo, en [7] se realizó un demostrador de OBS con MEMS. NTT y Fujitsu emplearon PLCs y MEMS en su *testbed* de OBS [63]. La Universidad de Tokio empleó conmutadores PLCs comerciales para construir una matriz de conmutación 16×16 con un tiempo de conmutación de 3 ms [104]. Por último, se debe destacar el *testbed* de OBS del proyecto OITDA en el que se empleó la tecnología PLZT alcanzando un tiempo de conmutación de 3.5 μs [2].

[17] Tecnología basada en dos fenómenos físicos. El primero es el patrón de interferencia de una rejilla óptica. Cuando una serie de líneas se iluminan con luz coherente (por ejemplo, láser) se forma un patrón de franjas claras y oscuras a cierta distancia de la rejilla. Cuando el número de rendijas es grande el patrón tiene una banda principal denominada de orden cero, donde está la mayor parte de la energía, y varias bandas laterales que contienen menos energía. El segundo fenómeno se basa en el efecto electro-óptico.

2.2.2 Láseres sintonizables ultra-rápidos

Los láseres sintonizables convencionales empleados en la transmisión WDM tienen unos tiempos de sintonización en el orden de segundos, siendo adecuados para la conmutación de circuitos, ya que una vez sintonizado el mismo, su configuración no cambia en el tiempo. Sin embargo, para poder compartir una longitud de onda entre varias fuentes, si se requiere un alto aprovechamiento del canal, la sintonización del láser debe realizarse de una manera mucho más rápida. Los láseres sintonizables ultra-rápidos que emplean la banda C de la ITU-T pueden cambiar de frecuencia en el orden de nanosegundos [41].

Existen principalmente dos métodos para los láseres ultra-rápidos, unos se basan en la selección entre múltiples cavidades, como por ejemplo el láser multi-frecuencia, MFL (*Multi-Frequency Laser*) y otros en sintonizar elementos de una cavidad, como los grating o resonadores de anillo, como los SG-DBR (*Sampled Grating Distributed Bragg Reflector*), que se resumen en la Tabla 2.3. Con respecto a los láseres MFL, se han reportado tiempos por debajo del nanosegundo con dispositivos capaces de sintonizar entre 32 y 56 canales. En cuanto a los láseres SG-DBR, se han reportado tiempos de sintonización de 45 nanosegundos entre 32 longitudes de onda en una rejilla de 100 GHz, reduciéndose a 5 nanosegundos con técnicas de pre-énfasis[18] [41].

Tabla 2.3 Láseres sintonizables ultra-rápidos [41]

Tipo de láser	Tiempo de sintonización
MFL	< 1ns
SG-DBR	45 ns
SG-DBR con preénfasis	5 ns

[18] Dado que la longitud de onda del láser está relacionada con la densidad de la portadora, se puede alcanzar un tiempo menor de conmutación hacia delante utilizando una técnica de pre-énfasis que modifica la corriente de excitación de una onda cuadrada a una con un sobreimpulso y subimpulso a la subida y bajada, respectivamente [99].

La Figura 2.11 muestra un ejemplo de la técnica de pre-énfasis, donde con un pulso de corriente de mayor amplitud (7.8 mA a 13.8 mA), la densidad de la portadora puede alcanzar el mismo nivel ($9x10^{23}/m^3$) aproximadamente 2.6 ns más rápido que el caso con un pulso de menor amplitud (7.8 a 12.5 mA).

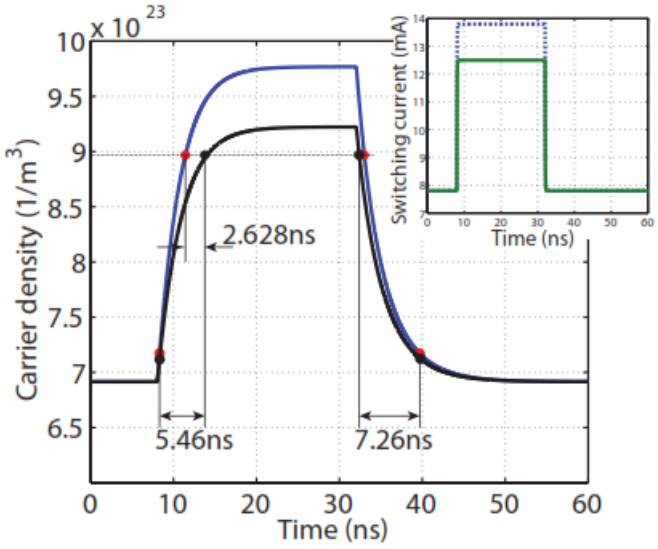


Figura 2.11 Respuesta en tiempo de la densidad de SG-DBR mediante una corriente de excitación con dos niveles diferentes (recuadro) [60]

2.2.3 Receptores en modo ráfaga

Los receptores tradicionales utilizados en los sistemas de transmisión óptica, tales como SONET/SDH, no son adecuados para la conmutación óptica de ráfagas, porque consideran que las componentes de fase y potencia de la señal entrante se mantienen constantes, y que dicha señal está siempre presente; lo cual no ocurre en una red OBS puesto que la fase y potencia de las ráfagas que llegan a un determinado receptor pueden presentar variaciones debido a que llegan de distintas fuentes y pueden haber atravesado diferentes rutas a través de la red. Adicionalmente, debido a la naturaleza propia de las ráfagas, una señal no está presente todo el tiempo sino únicamente durante el período de duración de una ráfaga.

Por lo antes mencionado, los receptores en modo ráfaga deben ser diseñados para adaptarse a las fluctuaciones de fase y potencia de las ráfagas entrantes. Otra característica de estos dispositivos es su alta velocidad de adquisición de reloj, que ha sido probada en escenarios de laboratorio con receptores capaces de recuperar el sincronismo de una señal entrante de 2.5 Gbps y 10 Gbps en decenas de ns [59].

2.2.4 Conversión de longitud de onda

En redes ópticas conmutadas por ráfagas que utilizan WDM, es deseable disponer de capacidades de conversión de longitud de onda en cada nodo para reducir la contención. Existen algunos métodos que se han desarrollado para realizar la conversión de longitud de onda, entre los cuales se tiene: conversión O/E/O, modulación de ganancia cruzada y mezcla de cuatro ondas, descritas a continuación:

Conversión electro-óptica.- La forma más sencilla de convertir una señal óptica de una longitud de onda a otra, es transformar la señal del dominio óptico al formato electrónico y utilizarla para modular una señal óptica en la longitud de onda de salida deseada. Este método es bastante simple y puede convertir señales que estén operando a velocidades de hasta10 Gbps; sin embargo, el método no es transparente y requiere que la señal óptica tenga un formato de modulación determinado y una velocidad de transmisión específica [59].

Conversión mediante Optical Gating[19].- La técnica principal emplea la modulación de ganancia cruzada XGM (*Cross Gain Modulation*), en la cual, la señal de entrada (λ_{pump}) se envía a través de un amplificador óptico de semiconductor (SOA) junto con una señal sonda (λ_{probe}) de onda continua en una longitud de onda diferente. Si la potencia de señal de entrada es lo suficientemente alta, el número de transiciones estimuladas en la zona activa del SOA aumenta, y por tanto la densidad de portadores disminuye, resultando en una reducción de su ganancia. Este fenómeno es conocido como saturación de ganancia del amplificador (ver

[19] Técnica que emplea un dispositivo óptico cuya característica cambia con la intensidad de la señal de entrada. Este cambio se transfiere a una señal sonda no modulada, con una longitud de onda diferente. Este método sirve únicamente par señales moduladas en intensidad.

Figura 2.12a) que causa una reducción en la amplificación de la señal sonda. Por el contrario, cuando la potencia de señal de entrada es baja, la señal sonda recibe amplificación completa; efecto que conduce a una modulación en la ganancia cruzada respecto a la señal modulada inicialmente. Por lo tanto, una copia invertida de los datos será transferida a la salida, con la misma longitud de onda de la señal sonda (ver Figura 2.12b).

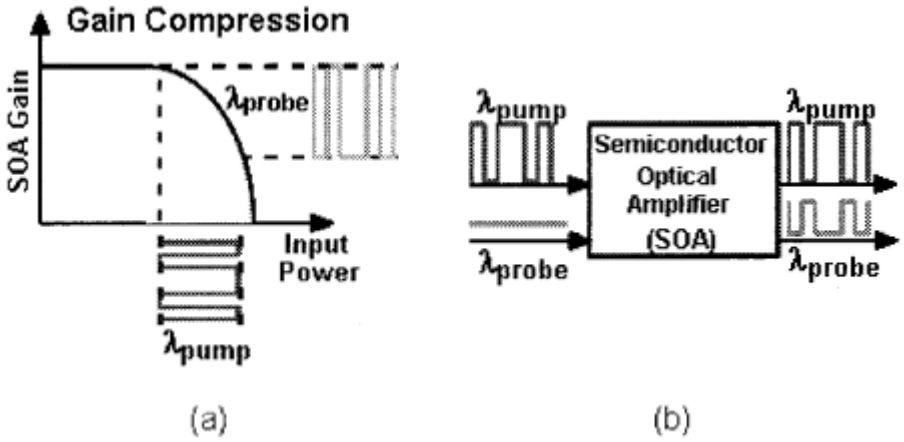


Figura 2.12 a) Saturación de ganancia en el SOA mediante una señal de bombeo de alta potencia, b) Una señal sonda de onda continua es modulada inversamente por la señal de entrada [29]

Además, estos dispositivos requieren un filtro a la salida del SOA para eliminar la señal a la longitud inicialmente modulada, dejando pasar únicamente la señal de datos a la nueva longitud de onda. Esta técnica puede ser utilizada únicamente en formatos de datos con modulación OOK (*On-off Keying*), ya que la información de fase se pierde en el proceso, y es capaz de convertir señales que estén operando a velocidades de hasta 10 Gbps, sin embargo sus principales limitaciones son que requiere alta potencia de entrada para la señal de datos, y que la señal de salida presenta una baja tasa de extinción (< 10) (relación de potencia de un símbolo‘1’ respecto a la potencia de un símbolo ‘0’), que resulta del hecho de que, aunque el SOA se encuentre en estado de saturación, la señal sonda sigue recibiendo cierta cantidad de amplificación. Otra desventaja es la degradación del OSNR debido al ruido de emisión espontánea, ASE (*Amplified Spontaneous Emission*) introducido por el SOA.

Conversión mediante efectos interferométricos.- Esta técnica explota la dependencia del índice de refracción sobre la densidad de portadores en los semiconductores. Como el índice de refracción cambia, la fase de la señal óptica que se propaga a través del medio también cambia. Este efecto se conoce como modulación de fase cruzada XPM (*Cross-phase Molulation*), que se puede convertir en modulación de amplitud haciendo que la señal se interfiera con una réplica de sí misma, después de que se ha sometido a un cambio de fase específico. La Figura 2.13 muestra una simple configuración interferométrica para conversión de longitud de onda. En ausencia de la señal de entrada (λ_{pump}) los brazos del interferómetro tendrán una diferencia de fase de π (p.ej. ajustando el *bias* de los SOAs), dando como resultado una interferencia destructiva a la salida del interferómetro. Por el contrario, cuando la señal de entrada está presente, la concentración de portadores en el SOA superior cambia, provocando que la señal sonda (λ_{probe}) experimente un cambio de fase en el brazo superior del interferómetro, resultando en una interferencia constructiva.

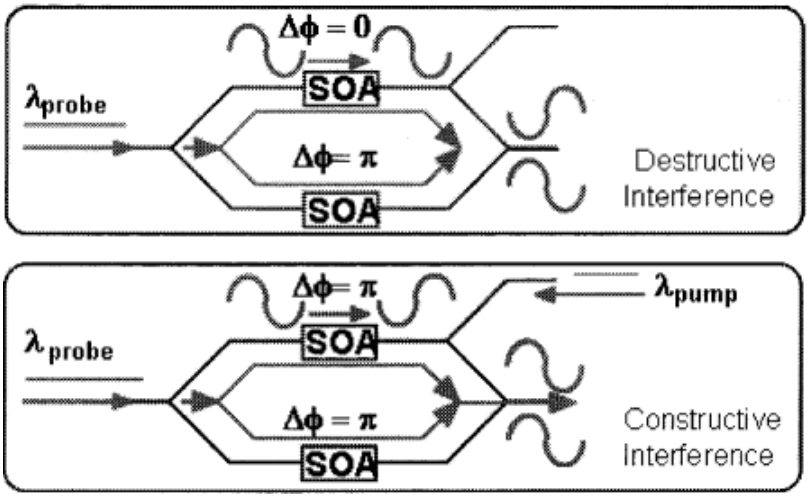


Figura 2.13 Conversión de longitud de onda interferométrica. La señal de bombeo que transporta los datos interrumpe el equilibrio entre los dos brazos [29]

En general, XPM proporciona mejor eficiencia de conversión que XGM, puesto que requiere menos potencia de entrada, obteniendo además una mejor relación de extinción.

Conversión mediante mezclado de ondas.- Otro método para proporcionar las capacidades de conversión óptica de longitud de onda, es el uso de la mezcla de cuatro ondas (FWM, *Four Wave Mixing*) que es el único que produce una copia exacta (desplazada en longitud de onda) de la señal; y se basa en un efecto no lineal, en el cual, las señales en las frecuencias f_1 y f_2 interactúan para crear nuevas componentes en las frecuencias $2f_1 - f_2$ y $2f_2 - f_1$, como se puede observar en la Figura 2.14. Si la señal de entrada (λ_{pump}) opera en la frecuencia f_1 y una señal sonda (λ_{probe}) en la frecuencia f2, los datos se generarán en nuevas señales ópticas con frecuencias $2f_1 - f_2$ y $2f_2 - f_1$, donde posteriormente se utiliza un filtro óptico para seleccionar la longitud de onda de salida. Este efecto se puede conseguir tanto dentro de una fibra como también en los SOAs, donde la no linealidad de tercer orden es mucho mayor que la de fibra.

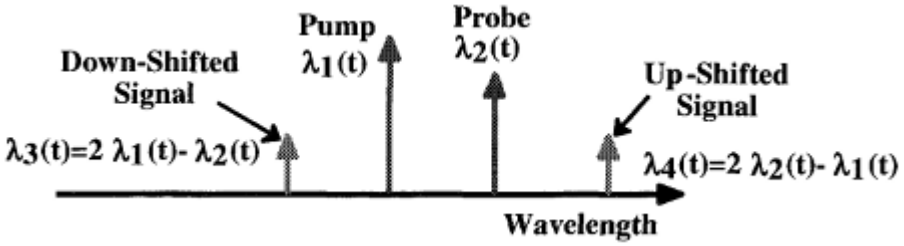


Figura 2.14 Conversión de longitud mediante el mezclado de cuatro ondas [29]

Su principal ventaja es la total transparencia en cuanto al formato de modulación y tasa de transmisión, mientras que sus desventajas son la necesidad de filtrar la onda resultante a la salida del SOA, la sensibilidad a la polarización, y una eficiencia de conversión no muy alta, puesto que disminuye a medida que incrementa la diferencia espectral entre las longitudes de onda de la señal de entrada y la señal sonda.

2.3 Inconvenientes de la capa física [1][16][59][82][94][100][118][135]

Al diseñar una red óptica conmutada por ráfagas, se deben tomar en cuenta una serie de restricciones propias de la capa física que incluyen atenuación, dispersión y no linealidades de la fibra óptica. Mientras muchos de estos inconvenientes aplican para redes ópticas en general,

ciertos asuntos pueden manifestar particulares preocupaciones en las redes ópticas conmutadas por ráfagas. En la Figura 2.15 se presenta un resumen de las limitaciones que se pueden experimentar en una fibra óptica del tipo monomodo utilizada en sistemas de transmisión basados en DWDM, que incluyen a las redes OBS.

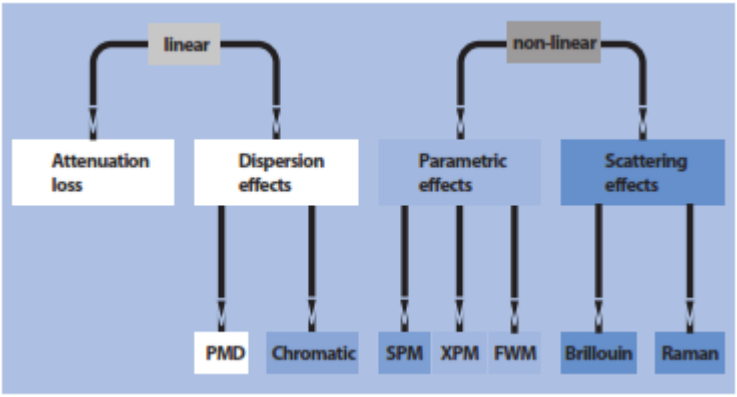


Figura 2.15 Limitaciones en fibras monomodo [16]

2.3.1 Atenuación

A medida que una señal de luz se propaga a lo largo de la fibra óptica, experimenta una disminución en su potencia, que es una función de la longitud de onda de la señal y es causada por factores intrínsecos como la absorción y dispersión de Rayleigh, y factores extrínsecos, que pueden ser producto de la tensión del proceso industrial, el ambiente, y el torcimiento físico. La absorción se produce cuando la luz incidente sobre las moléculas de silicio o las impurezas en la fibra es absorbida. Para la mayoría de las fibras, la cantidad de absorción es reducida para el rango de longitudes de onda útiles (entre 0.8 μm y 1.6 μm) siendo prácticamente despreciable. La dispersión de Rayleigh se produce cuando pequeñas variaciones en el índice de refracción de la fibra, producto de la falta de homogeneidad de su material de fabricación, dispersan la luz.

En una red óptica conmutada por ráfagas, puede limitar la distancia máxima que una ráfaga puede atravesar ópticamente. En la mayoría de los casos, se pueden utilizar amplificadores

ópticos para superar la atenuación; sin embargo, estos dispositivos normalmente introducen ruido, degradando finalmente la relación señal a ruido óptica (OSNR, *Optical Signal to Noise Ratio*) que es fundamental para la óptima operación de la red.

2.3.2 Dispersión

La dispersión es el ensanchamiento de los pulsos de luz en el dominio del tiempo cuando éstos viajan a través de la fibra óptica, lo cual resulta en una distorsión de la señal. Existen dos tipos de dispersión que pueden afectar a sistemas basados en DWDM, y por ende a redes conmutadas por ráfagas; uno de ellos denominado dispersión cromática, y el otro conocido como dispersión por modo de polarización (PMD).

La dispersión cromática se produce como consecuencia de la velocidad de la luz en una fibra que es una función de la longitud de onda. De esta manera, si la señal transmitida está compuesta de más de una longitud de onda, algunos componentes se propagarán más rápido que otros, provocando que la señal se disperse en el dominio del tiempo (ver Figura 2.16).

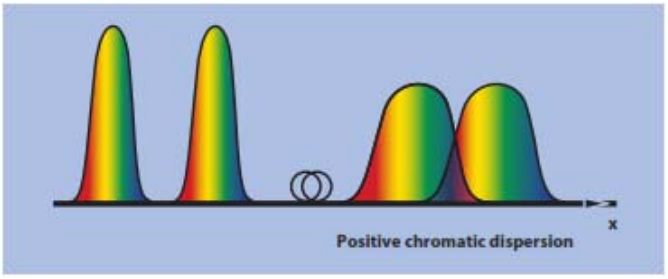


Figura 2.16 Dispersión Cromática [16]

Los tipos de dispersión cromática incluyen la dispersión del material, en la cual el índice de refracción de la fibra varía como una función de la longitud de onda, y la dispersión de guía de onda, en la que el índice de refracción para una longitud de onda particular depende de la fracción de potencia que viaja dentro del núcleo y del revestimiento de una fibra. Dado que el ensanchamiento de un pulso se puede ampliar lo suficiente como para interferir con los pulsos

adyacentes (bits o símbolos) en la fibra, conduciendo a la interferencia entre símbolos, este efecto limita el espaciamiento entre bits y por ende la tasa de transmisión máxima en un canal de fibra óptica.

Existen diferentes alternativas para contrarrestar la dispersión cromática, que incluyen la fabricación de fibras con dispersión nula para ciertas longitudes de onda como 1310 nm y 1550 nm, así como también módulos de compensación de dispersión definidos para distancias específicas, e incluso se han propuesto compensadores sintonizables.

Por otro lado, el vector unitario de polarización que representa el estado de polarización del vector del campo eléctrico, no permanece constante en las fibras ópticas prácticas, sino que cambia de forma aleatoria a lo largo de la fibra debido a su birrefringencia[20] fluctuante, que es la principal causa de la dispersión por modo de polarización (PMD) y se define como el efecto ocasionado por los dos modos de polarización ortogonales de una señal, que viajan estadísticamente a diferentes velocidades de grupo y por ende llegan al final de la fibra a instantes distintos de tiempo, dando como resultado un ensanchamiento o distorsión de la señal óptica. La diferencia temporal entre ambos modos de polarización se denomina retardo diferencial de grupo (DGD, *Differential Group Delay*), denotado usualmente como $\Delta\tau$, como se ilustra en la Figura 2.17, que se utiliza para determinar la magnitud de PMD.

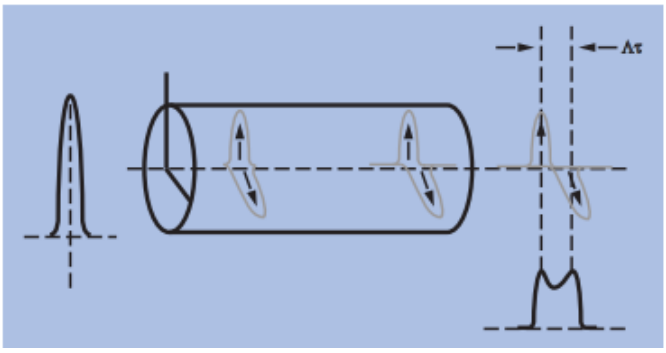


Figura 2.17 Dispersión por modo de polarización (PMD) [16]

[20] El grado de birrefringencia se describe por la diferencia de los índices de refracción de los modos polarizados ortogonalmente $B_n = |n_x - n_y| = \Delta_n$, donde n_x y n_y son el índice de refracción de polarizaciones x e y respectivamente. En una fibra monomodo típica, el coeficiente de birrefringencia es de aproximadamente 10^{-7} [100].

Se tiene dos fuentes de birrefringencia comunes que son la birrefringencia geométrica (relacionada con pequeñas desviaciones de la simetría cilíndrica perfecta) y el estrés anisotrópico (producido en el núcleo de la fibra durante la fabricación o instalación de la fibra) [100]. El coeficiente de PMD (dado en ps/√km) varía a través de distintas partes de la fibra y está sujeto a la tensión mecánica y temperatura. Si los límites del PMD para una cierta fibra son excedidos para altas velocidades de transmisión, se debe reducir su tasa de envío.

Además de los problemas con la interferencia entre símbolos, la dispersión también puede introducir problemas de sincronización en las redes ópticas conmutadas por ráfagas, donde la cabecera de la ráfaga se envía normalmente en una longitud de onda diferente de la que se utiliza para la ráfaga de datos. Cada una de estas longitudes de onda experimentará distintos grados de dispersión, causando que la cabecera y la ráfaga tiendan a separarse o acercarse más en el dominio del tiempo, como se ilustra en la Figura 2.18. Si se conocen las distancias físicas de cada enlace y el perfil de dispersión de la fibra, puede ser posible compensar este efecto ajustando apropiadamente el tiempo de *offset* en el nodo de origen.

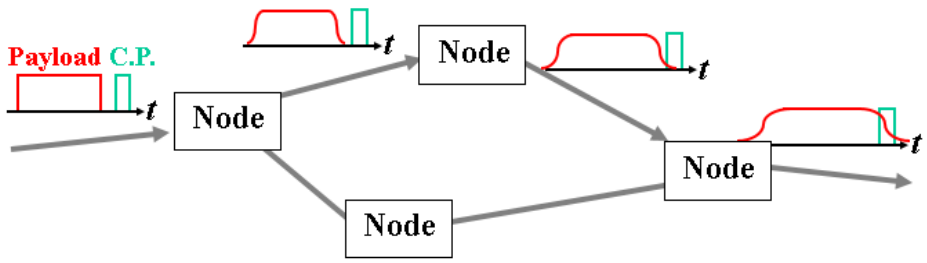


Figura 2.18 Efectos de la dispersión en la ráfaga de datos y paquete de control [118]

Los valores típicos para el parámetro de PMD en la mayoría de fibras varían en el rango de 0.1 – 1 ps/√Km, mientras que en las fibras modernas se puede obtener un valor de PMD tan bajo como 0.05 ps/√Km [1], que en sí es una limitación más restrictiva para altas velocidades de transmisión (10 Gbps o más). Sin embargo, con los avances en el área de la polarización del campo óptico, particularmente la comunicación óptica coherente y la inclusión de OFDM en las comunicaciones ópticas, PMD ya no es una barrera fundamental para la transmisión de alta velocidad y puede ser estimada y compensada eficazmente. No obstante conviene señalar que

la aplicación de las tecnologías coherente y multi-portadora no implican el final de la investigación en PMD en las comunicaciones ópticas. De hecho, PMD ha resurgido como una característica de canal sumamente importante que requiere ser revisada [100].

2.3.3 No linealidades de la fibra

Las no linealidades en la fibra normalmente tendrán un efecto sobre los parámetros de operación, como la tasa de transmisión, número de canales WDM, espaciamiento de canales, y potencia de la señal. En la subsección 2.2.4 se mencionaron algunos efectos no lineales que se emplean en las técnicas de conversión de longitud de onda, como la mezcla de cuatro ondas (FWM, *Four Wave Mixing*), la auto modulación de fase (SPM, *Self-Phase Modulation*), la modulación de fase cruzada (XPM, *Cross Phase Modulation*), que se producen debido a la dependencia del índice de refracción con la intensidad de la luz; existiendo además dos fenómenos adicionales que se producen por efectos de *scattering* en la fibra conocidos como dispersión estimulada de Raman (SRS, *Stimulated Raman Scattering*) y dispersión estimulada de Brillouin (SBS, *Stimulated Brillouin Scattering*). Dichos efectos se describen brevemente a continuación [59].

La mezcla de cuatro ondas (FWM).- Se produce cuando dos longitudes de onda, que operan en las frecuencias f_1 y f_2, respectivamente, se mezclan para producir señales de frecuencias $2f_1 - f_2$ y $2f_2 - f_1$. Estas señales adicionales, llamadas bandas laterales, pueden causar interferencia si se sobreponen con frecuencias utilizadas para la transmisión de datos. Asimismo, la mezcla puede ocurrir entre combinaciones de tres o más longitudes de onda. El efecto de FWM en sistemas WDM se puede reducir mediante el uso de canales espaciados desigualmente [59].

La automodulación de fase (SPM).- Se produce cuando los cambios en la intensidad de una señal resultan en variaciones en la fase de una señal. Las variaciones instantáneas en la fase de una señal pueden introducir componentes en frecuencia adicionales en la señal, los cuales combinados con los efectos de dispersión, conducirán al esparcimiento o compresión de los pulsos ópticos en el dominio del tiempo.

La modulación de fase cruzada (XPM).- Es un desplazamiento en la fase de una señal causado por el cambio en la intensidad de una señal que se propaga en una longitud de onda diferente. Similar a la automodulación de fase, los desplazamientos en fase pueden introducir componentes en frecuencias adicionales, dando lugar a mayor dispersión. Aunque la fase cruzada puede limitar el desempeño de los sistemas de comunicación óptica, también puede tener aplicaciones ventajosas. Mediante la modulación de fase cruzada, una señal a una longitud de onda dada se puede utilizar para modular una señal de bombeo a una longitud de onda diferente. Tales técnicas se pueden utilizar en dispositivos de conversión de longitud de onda.

La dispersión estimulada de Raman (SRS).- Se produce por la interacción de la luz con las vibraciones moleculares del material (*fonones*). Así, cuando la luz incidente colisiona con las moléculas de sílice, experimenta un desplazamiento de frecuencia debido a la transferencia de energía de algunos fotones de la señal óptica de entrada en fonones ópticos y la creación nuevos fotones de menor energía, y por ende con una longitud de onda mayor (frecuencia menor) que la señal incidente, debido a la relación inversamente proporcional entre estos dos parámetros[21].

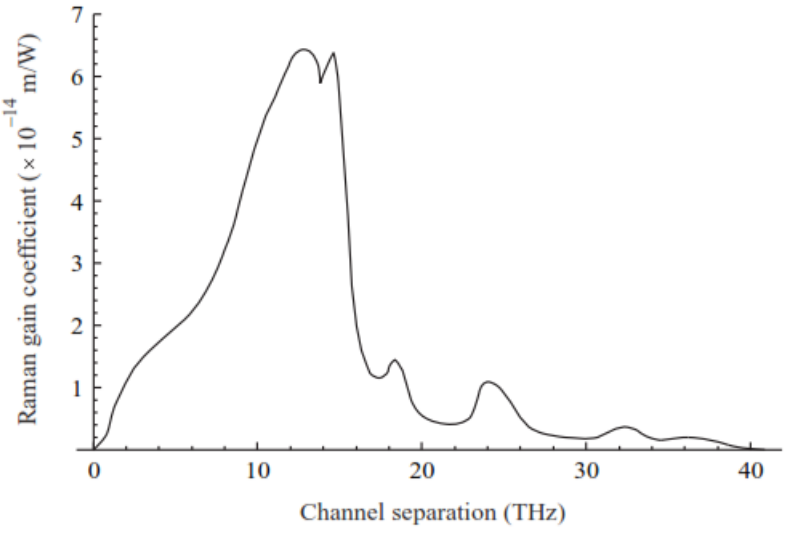


Figura 2.19 Coeficiente de ganancia SRS como una función de la separación del canal [94]

[21] La energía de un fotón a una longitud de onda λ está dada por *h*c/ λ, donde *h* es la constante de Planck (6.63x10-34 J.s).

De esta manera, una porción de la luz que viaja en cada frecuencia dentro de una fibra óptica es reducida a lo largo de una región de frecuencias más bajas, donde la luz dispersada que se genera se denomina onda de Stokes, cuyo rango de frecuencias es determinado por el espectro de ganancia Raman[22] que abarca un rango de alrededor de 40 THz, por debajo de la frecuencia de la luz de entrada. En la fibra de sílice, la onda de Stokes tiene una ganancia máxima a una frecuencia de alrededor de 13.2 THz menor que la señal de entrada, como se puede apreciar en la Figura 2.19.

La fracción de potencia transferida a la onda de Stokes crece rápidamente a medida que se incrementa la potencia de la señal de entrada. Bajo una potencia muy alta de entrada, SRS ocasionará que casi toda la potencia de la señal de entrada se transfiera a la onda de Stokes [82].

En sistemas con múltiples longitudes de onda, SRS causa una transferencia de potencia de los canales de menor longitud de onda hacia canales de mayor longitud de onda dentro del espectro de ganancia Raman. Para reducir la cantidad de pérdida, la potencia de cada canal debe estar por debajo de un cierto nivel. En [59] se hace referencia a un estudio realizado, cuyos resultados muestran que en un sistema de 10 canales con espaciamiento de 10 nm entre sí, la potencia de cada canal debería mantenerse por debajo de 3 mW para minimizar los efectos de SRS.

La dispersión estimulada de Brillouin (SBS).- Es similar a SRS, excepto que el desplazamiento en frecuencia se produce por fonones acústicos. En SBS, la luz desplazada se propaga a lo largo de la fibra en la dirección opuesta a la señal de entrada. La intensidad de la luz dispersada es mucho mayor en SBS que en SRS, pero el rango de frecuencias de SBS, del orden de 10 GHz, es mucho menor que el de SRS.

Además, el ancho de banda de ganancia de SBS es sólo del orden de 100 MHz. Para contrarrestar los efectos de SBS, se debe asegurar que la potencia de entrada esté por debajo de un umbral. Además, en sistemas con múltiples longitudes de onda, SBS puede inducir

[22] El espectro de ganancia Raman describe típicamente el coeficiente de ganancia Raman medido para fibras de silicio como una función del desplazamiento en frecuencia a una longitud de onda de bombeo de 1.0 μm.

diafonía (*crosstalk*) entre los canales, que ocurrirá cuando dos canales que se propagan en direcciones opuestas difieren en frecuencia por el desplazamiento de Brillouin, que está alrededor de 11 GHz para longitudes de onda a 1550 nm. Sin embargo, el estrecho ancho de banda de ganancia de SBS hace a la diafonía de SBS bastante fácil de evitar [59].

La interacción entre el ruido ASE y la no linealidad de la fibra, y la interacción entre el efecto estocástico de PMD y la no linealidad de la fibra no se pueden compensar completamente en el receptor. Por lo tanto, en la práctica sólo es factible una compensación parcial de la no linealidad de la fibra [135].

2.4 Ensamblado de ráfagas [18][27][39][55,59][111][125,126,128]

El ensamblado de ráfagas es el proceso mediante el cual un nodo de ingreso agrupa una determinada cantidad de paquetes, provenientes de las capas superiores, en unidades de mayor tamaño denominadas ráfagas, en función de una política de ensamblado predefinida y las envía hacia el núcleo de la red OBS en el dominio completamente óptico. Las ráfagas de datos pueden alcanzar normalmente longitudes de algunas decenas o centenas de kilobytes.

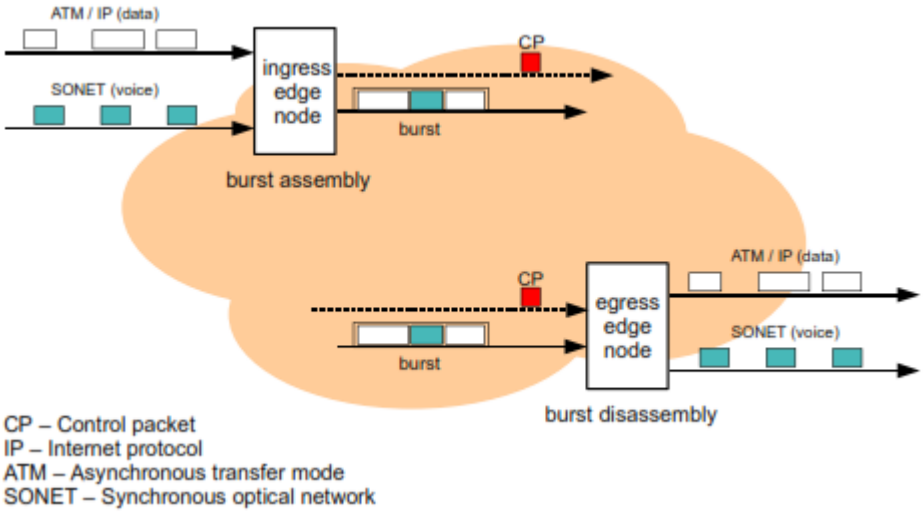


Figura 2.20 Ensamblado y desensamblado de ráfagas en el borde de una red OBS [27]

A medida que los paquetes arriban a la interfaz de un nodo de ingreso, éstos son pre-clasificados y almacenados temporalmente en *buffers* electrónicos, en función de una determinada clase de tráfico que representa datos de clientes con características similares, relacionadas principalmente con su destino y clase de servicio, esta última utilizada opcionalmente si se requiere soporte para la provisión de calidad de servicio QoS (*Quality Of Service*). Por otro lado, cuando las ráfagas llegan al nodo egreso de destino son desensambladas en sus paquetes originales. En la Figura 2.20 se ilustran gráficamente los procesos de ensamblado y desensamblado de ráfagas.

Las políticas de ensamblado establecen su funcionamiento en al menos un criterio basado en umbral, que define la creación y envío de una ráfaga de datos, determinando así la característica de llega de ráfagas al núcleo de la red OBS. El nivel de umbral puede ser establecido en función de uno de los siguientes criterios: vencimiento de un temporizador, tamaño máximo de ráfaga o una combinación de ambos (esquema híbrido). Los parámetros involucrados en estas políticas, que se derivan en algoritmos de ensamblado incluyen un tiempo de umbral T, una longitud mínima de ráfaga B_{min}, y una longitud máxima de ráfaga B. B_{min} se puede calcular en base al tiempo de procesamiento de la cabecera de la ráfaga en cada nodo y a la relación de los canales de control con respecto al número de canales de datos en la fibra [59].

Por otro lado, los umbrales T y B pueden ser establecidos de forma predefinida o dinámica, dando lugar a la siguiente clasificación de algoritmos de ensamblado de ráfagas.

2.4.1 Algoritmos de ensamblado basados en tiempo

Este tipo de esquemas utilizan un temporizador que arranca al inicio de cada nuevo ciclo de ensamblado y una vez transcurrido un tiempo fijo T (expiración del temporizador), todos los paquetes que han arribado durante ese período son ensamblados en una ráfaga. Por lo tanto, este umbral de tiempo T es empleado como límite máximo para la formación de ráfagas, que se transmiten dentro de la red OBS a intervalos periódicos de T unidades de tiempo, proporcionando vacíos uniformes entre ráfagas sucesivas del mismo nodo de ingreso dentro

las redes de *core*; sin embargo el tamaño de las ráfagas puede ser variable y depende de la tasa de llegada de los paquetes.

Adicionalmente, si la longitud de la ráfaga es menor que B_{min} en el el instante en que se debe realizar la transmisión, el algoritmo añade bits de relleno hasta cumplir con la longitud mínima requerida, antes de enviar dicha ráfaga.

2.4.2 Algoritmos de ensamblado basados en la longitud de ráfaga

Este tipo de mecanismos emplean un umbral de longitud de ráfaga *B*, como parámetro limitante en cuanto al número máximo de paquetes contenidos en cada ráfaga, dando como resultado unidades de datos de tamaño fijo que se transmiten dentro de la red OBS a intervalos no periódicos, una vez que se alcanza o supera el valor de *B*, lo cual origina retardos variables en las colas de ensamblado dependiendo de la tasa de llegada de los paquetes. En este caso, todas las ráfagas tendrán el mismo número de paquetes cuando ingresan dentro de la red; sin embargo, a medida que una ráfaga atraviesa el *core* OBS, su longitud puede cambiar en función de las políticas de resolución de contenciones configuradas en la red, tales como la segmentación de ráfagas.

Tanto los métodos de ensamblado basados en tiempo como los métodos basados en tamaño de ráfaga pueden ser considerados similares, desde el punto de vista en que a una determinada tasa constante de llegada, un valor de umbral se puede mapear a un valor de *timeout* y viceversa, dando lugar a ráfagas de longitud similar para cada caso [59]. A pesar de esto, ambos algoritmos tienen diferencias y experimentan inconvenientes bajo ciertas circunstancias que dependen principalmente de la carga de tráfico, debido a que consideran un solo criterio (tiempo o longitud) para la formación de ráfagas. Cuando la carga de tráfico es baja, un algoritmo de ensamblado basado en longitud de ráfaga no es capaz de garantizar un límite en cuanto al tiempo de espera antes de liberar la ráfaga para su transmisión; en tales condiciones, si el umbral de tamaño *B* es grande, el tiempo medio de espera de los paquetes en la cola de ensamblado llega a ser elevado. De igual manera, cuando la carga de tráfico es alta, un algoritmo de ensamblado basado en tiempo conduciría a ráfagas de gran tamaño y el tiempo

medio de espera llega a ser mayor comparado con el caso del ensamblado basado en longitud de ráfaga [111].

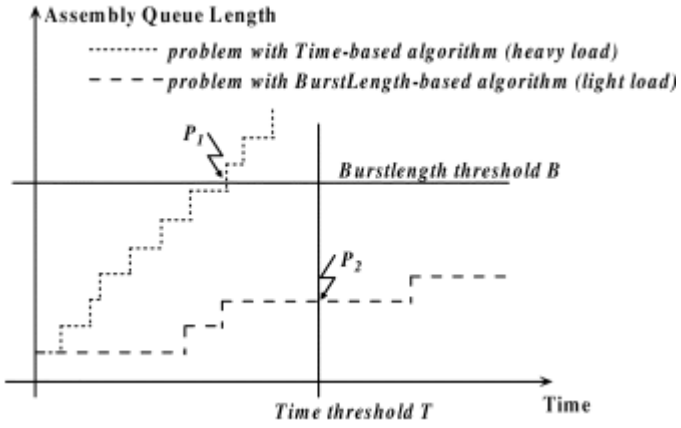


Figura 2.21 Criterios para los algoritmos de ensamblado de ráfagas [128]

En la Figura 2.21 se puede apreciar claramente que en condiciones de baja carga, es preferible utilizar un algoritmo basado en tiempo, mientras que un algoritmo basado en tamaño de ráfaga es más apropiado en escenarios de alta carga. Por esta razón, se hace necesario contar con mecanismos que se adapten a todas las condiciones de carga de tráfico y es así que surgen los algoritmos de ensamblado mixto o híbridos en los que se utilizan ambos criterios de umbral, es decir, tiempo y longitud de ráfaga.

2.4.3 Algoritmos de ensamblado mixto o híbridos

Este tipo de esquemas reúne las características de los dos algoritmos mencionados anteriormente, es decir utiliza los tres niveles de umbral: tiempo T, longitud de ráfaga B, y longitud mínima de ráfaga B_{min} [125], para generar y transmitir una ráfaga. Dependiendo de las condiciones de tráfico y de los umbrales configurados, normalmente el valor de T o B es superado primero, liberando así la ráfaga para su transmisión.

En general, el umbral T será alcanzado antes que B para bajas cargas de tráfico, mientras que lo contrario ocurrirá para altas cargas de tráfico. Sin embargo, se debería notar que, en este método híbrido, si el umbral de longitud de ráfaga B es mucho mayor que la cantidad promedio de datos que arriban en el tiempo T, es decir $T\ x\ \lambda_p << B$ (donde λ_p es la tasa de llegada promedio de paquetes a la cola de ensamblado), la longitud de la ráfaga ensamblada nunca alcanzará B antes de que sea enviada en el punto P2, como se ilustra en la Figura 2.21. En tal caso, el algoritmo de ensamblado mixto funcionará de la misma forma que el algoritmo de ensamblado basado en tiempo con un umbral de tiempo fijo T. Por otro lado, si el umbral de longitud de ráfaga B es mucho menor que el promedio de datos que arriban en el tiempo T, es decir $T\ x\ \lambda_p >> B$, el umbral de tiempo T nunca será alcanzado antes de que la longitud de una ráfaga alcance B en el punto P1, como se observa en la Figura 2.21; en este caso, el algoritmo de ensamblado mixto opera igual que el algoritmo de ensamblado basado en tamaño de ráfaga [128].

De todas maneras, este esquema proporciona mejor rendimiento y mayor flexibilidad puesto que considera los dos criterios de umbral (tiempo y longitud), evitando de esta forma el envío de ráfagas demasiado pequeñas y retardos de encolamiento de larga duración. En vez de utilizar umbrales fijos de tiempo y tamaño, uno de ellos o los dos se pueden ajustar dinámicamente de acuerdo con las condiciones del tráfico, dando lugar a la siguiente clase de algoritmos de ensamblado.

2.4.4 Algoritmos de ensamblado dinámicos

Este tipo de mecanismos se basan principalmente en las condiciones de tráfico utilizando mediciones en tiempo real y técnicas predicción, para ajustar dinámicamente los umbrales de tiempo o longitud de ráfaga, o incluso ambos. Estos algoritmos ofrecen un mejor desempeño que los esquemas fijos, pero a costa de una mayor complejidad computacional en comparación con los tres mecanismos de ensamblado mencionados anteriormente, que utilizan umbrales fijos.

Existe un cierto número de esquemas de ensamblado dinámico de ráfagas propuestos para adaptarse a diferentes tipos de tráfico. Algunos han sido diseñados especialmente para ser utilizados con el tráfico TCP de manera que su *throughput* sobre redes OBS se mejora mediante la reducción del tamaño de ráfaga en caso de contenciones. Otros utilizan metodologías de predicción para determinar la tasa de llegada del tráfico en el nodo de borde y transmisión anticipada de los paquetes de control, con el fin de reducir el retardo de ensamblado de ráfagas incurrido en el bode de las redes OBS; como es el caso del algoritmo de ensamblado denominado FRR (*Forward Resource Reservation*).

Un factor clave y a la vez complejo en los esquemas de ensamblado de ráfagas, es la elección apropiada de los umbrales del temporizador y longitud de ráfaga, que permita minimizar la probabilidad de pérdida de paquetes al atravesar una red OBS. El criterio de selección del valor óptimo es una cuestión abierta; sin embargo, se debe tener en cuenta que si el umbral es demasiado bajo, se generará un mayor número de ráfagas pequeñas en la red y por tanto aumentará la probabilidad de contenciones, pero el número medio de paquetes perdidos por contención es menor; aunque existirá una mayor carga sobre el plano de control que debe procesar las cabeceras de cada ráfaga de datos de una manera rápida y eficiente, conduciendo incluso a una baja utilización de la red si el tiempo de reconfiguración del conmutador no es despreciable. Por otro lado, si el umbral es demasiado alto, se inyectará un menor número de ráfagas de gran tamaño en la red y por ende se reducirá la probabilidad de contenciones en comparación con el caso anterior, pero el número medio de paquetes perdidos por contención aumentará. De esta manera, se puede ver que existe un compromiso entre el número de contenciones y el número promedio de paquetes perdidos por contención [59].

Ha habido mucho debate en cuanto al impacto del proceso de ensamblado, dada la naturaleza a ráfagas del tráfico entrante, que se cree reduce su grado de *auto-similaridad* (efecto suavizante). Un estudio reportado en [39] mediante el uso de métodos basados en temporizador, demostró una reducción de las características de *auto-similaridad* del tráfico ensamblado en el *backbone* óptico; característica que generalmente se considera deseable puesto que conduce a una reducción en el retardo de encolamiento y a una menor pérdida de paquetes en las redes OBS. Sin embargo, este resultado contradice el presentado más tarde en [55][125][126] donde se afirma que el ensamblado de ráfagas sólo cambia la dependencia a corto plazo del tráfico

entrante, mientras que la característica a largo plazo permanece igual. Por otro lado, el estudio en [55] mostró que la influencia de la *auto-similaridad* en la probabilidad de bloqueo es despreciable, ya que el proceso de llegada se puede asumir que es de Poisson en la escala de tiempo de interés para el bloqueo de ráfagas.

Adicionalmente, en [55][125][126], los autores sostienen que para un esquema de ensamblado basado en temporizador con una distribución entre llegadas de ráfagas fija (T), la distribución de longitud de las ráfagas es Gaussiana. De igual manera, para un esquema de ensamblado basado en longitud de ráfaga con una distribución fija (B_{max}), la distribución entre llegadas de ráfagas es Gaussiana. Sin embargo, se menciona también que, aunque la dependencia a corto plazo tiene un efecto suavizante, las técnicas de agregación de ráfagas basadas en temporizador y en longitud de ráfaga no pueden reducir la dependencia a largo plazo. Finalmente, a través de simulaciones los autores muestran en [125][126] que la estructura de correlación en general a escalas de tiempo infinito no cambia.

2.5 Señalización [22,27][41][51,59][95,96][116][124,128]

Para que una ráfaga de datos pueda atravesar los diferentes nodos de una red óptica conmutada por ráfagas, es necesario implementar un esquema de señalización que permita asignar los recursos en cada uno de los nodos (puerto y longitud de onda) y pre-configurar los conmutadores ópticos de manera que toda la red esté preparada para permitir el paso de la ráfaga de datos, esto debido a la naturaleza sin almacenamiento propia de las redes OBS. Los esquemas de señalización en una red óptica conmutada por ráfagas, se implementan normalmente utilizando paquetes de control que se envían fuera de banda, es decir, por un canal óptico diferente del que se emplea para la ráfaga de datos, la cual se transmite un tiempo después (*offset*) de su correspondiente BHP. Esta separación en longitud de onda y tiempo entre la cabecera y los datos de cada ráfaga, permite a este tipo de redes transportar grandes capacidades de información en el dominio completamente óptico con un reducido procesamiento electrónico.

En general, un esquema de señalización se puede clasificar de acuerdo a las siguientes características [59]:

- Dirección en una sola vía, en dos vías o híbrida
- Iniciación por origen, por destino o por nodo intermedio
- Recurso persistente o no persistente
- Reservación de recursos inmediata o retardada
- Liberación de recursos explícita o implícita
- Control centralizado o distribuido

Dirección en una vía, dos vías o híbrida.- En un esquema de señalización con dirección en una sola vía, el nodo origen envía un paquete de control hacia el nodo destino solicitando la reserva de recursos y la configuración de los conmutadores ópticos en cada nodo a lo largo de la red. Posteriormente, el nodo origen envía la ráfaga de datos sin necesidad de esperar por un acuse de recibo enviado de vuelta desde el destino, que notifique el éxito o fracaso del intento de reserva. En el primer caso, se experimentará una reducción del retardo extremo a extremo en la transferencia de datos, mientras que en el segundo caso, la ráfaga será descartada dando lugar a pérdida de paquetes.

En la técnica de señalización con dirección en dos vías, el nodo origen envía una solicitud de reserva de recursos y espera recibir un mensaje de reconocimiento antes de enviar la ráfaga de datos, lo cual puede eliminar la pérdida de ráfagas en la red OBS, pero incrementa el retardo extremo a extremo de cada ráfaga. En caso de que un determinado nodo no disponga de los recursos suficientes para la reserva solicitada, deberá enviar un mensaje de reconocimiento negativo hacia el origen y además tomar las medidas necesarias para la liberación de recursos de los nodos previamente configurados.

Por otro lado, un esquema de señalización híbrido reúne las dos técnicas de señalización antes mencionadas, proporcionando una confirmación parcial de recursos, donde las reservas desde el origen hacia un nodo intermedio de la red son confirmadas, mientras que las reservas desde el nodo intermedio hacia el destino no se confirman, de esta manera la ubicación del nodo intermedio determinará las características de retardo extremo a extremo y pérdidas de paquetes para cada ráfaga de datos.

Iniciación por origen, por destino, o por nodo intermedio.- Una técnica de señalización puede iniciar la reserva de los recursos solicitados en el origen, en el destino o en un nodo intermedio. En la técnica de iniciación por origen, los recursos son reservados en el sentido hacia adelante, a medida que el paquete de control atraviesa la red desde el origen hasta el destino, pudiendo utilizar o no mensajes de confirmación de recursos antes de transmitir una ráfaga. La principal causa de pérdida de datos en esta técnica es indudablemente la falta de recursos.

En la técnica de iniciación por destino, el nodo origen transmite una petición de recursos hacia el destino con el fin de recolectar información de la disponibilidad de longitudes de onda de cada enlace a lo largo del camino, en base a la cual el nodo destino escogerá una longitud de onda disponible (si hay) y enviará una solicitud de reserva de retorno hacia el origen, que se encargará de reservar las longitudes de onda elegidas a medida que atraviesa los diferentes nodos de la red. Esta técnica sufre de pérdidas, debido a la información de disponibilidad desactualizada, producto de que una longitud de onda que fue seleccionada puede ser tomada por otra petición, durante el tiempo conocido como período vulnerable[23].

En un esquema de iniciación por nodo intermedio, normalmente los recursos son reservados desde la fuente a algún nodo intermedio, de forma análoga a la iniciación por destino, y desde el nodo intermedio al destino, de forma similar a la iniciación por origen.

Recurso persistente o no persistente.- Un esquema de señalización debe tomar la decisión de esperar por un recurso que se encuentra ocupado hasta que llega a estar disponible, o activar inmediatamente la política de resolución de contenciones que esté implementada en la red como retransmisión, deflexión, *buffering*, etc. En el método persistente, el nodo debe esperar hasta que el recurso bloqueado se encuentre libre para asignar la longitud de onda correspondiente, lo cual conduce a una mínima pérdida si los nodos de la red cuentan con los *buffers* necesarios para el efecto. En el método no persistente, si el recurso no está disponible inmediatamente, el nodo declara fallida la petición de reserva y procede a efectuar las técnicas de resolución de contenciones apropiadas.

[23] Período entre el instante de recolección del estado de recursos de un nodo intermedio y el momento en el que arriba el mensaje de reserva a ese nodo.

Reserva de recursos inmediata o retardada.- De acuerdo a la duración de la reserva del canal óptico, los esquemas de señalización pueden soportar reservaciones inmediatas o retardadas. En el primer caso, el nodo reserva el canal para permitir el paso de la ráfaga de datos, desde el instante en que recibe su paquete de control. En el segundo caso, la reserva del canal inicia a partir del arribo de la ráfaga de datos al nodo, el cual determina su llegada en base al tiempo de *offset* que debe ser especificado en su paquete de control.

En general, la reservación inmediata es más sencilla de implementar que la reservación retardada, pero la primera incurre en una mayor probabilidad de bloqueo debido a la asignación ineficiente del ancho de banda, mientras que la segunda conduce a una mayor utilización de ancho de banda.

Liberación de recursos explícita o implícita.- Una reserva existente se puede liberar de forma explícita o implícita. En el primer caso, el nodo origen envía hacia el destino un mensaje de control adicional después de la ráfaga de datos, solicitando la liberación de recursos. En el segundo caso, la terminación de la reserva se realiza implícitamente en cada uno de los nodos a medida que termina la duración de la ráfaga de datos, particular que se precisa de acuerdo al tiempo de *offset* y longitud de la ráfaga, que deben ser transportados en su paquete de control.

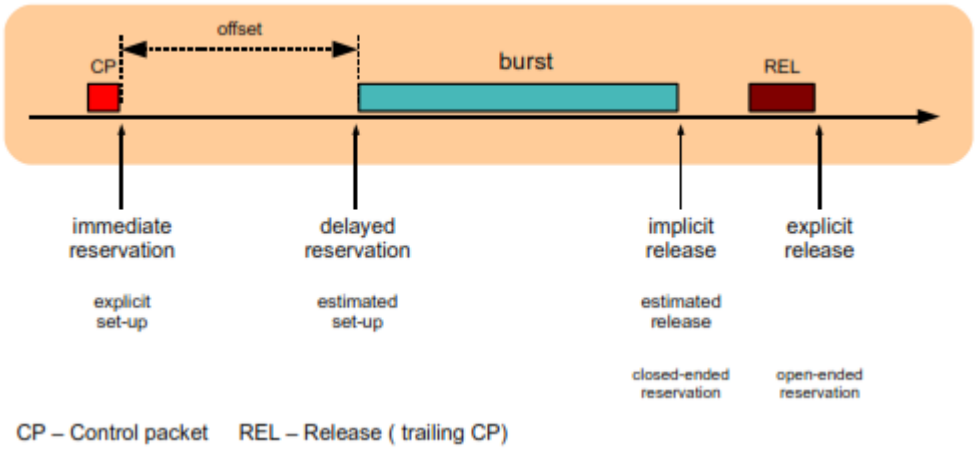


Figura 2.22 Mecanismos de reserva y liberación en OBS [27]

En función de las técnicas de reserva y liberación (ver Figura 2.22), los esquemas de señalización se pueden clasificar en cuatro categorías: *reserva inmediata con liberación explícita, reserva inmediata con liberación implícita, reserva retardada con liberación explícita y reserva retardada con liberación implícita* [59], que difieren en la cantidad de tiempo durante el cual una misma ráfaga utilizará los recursos de la red asignados, conduciendo de esta manera a un compromiso entre el desempeño global del sistema y la complejidad radicada principalmente en el planificador del nodo central.

Es claro que los esquemas implícitos proporcionan un mejor desempeño que sus contrapartes explícitas, debido a que reducen la probabilidad de bloqueo en el dominio OBS y permiten por ende una mayor utilización del ancho de banda producto de una reserva eficiente de los recursos.

Control centralizado o distribuido.- Un esquema de señalización centralizado con reserva en dos vías utiliza un servidor central de peticiones que tiene como funciones principales el procesamiento de los paquetes de control, la determinación de la mejor ruta entre los diferentes pares origen y destino, y la reserva de recursos sobre cada enlace a lo largo de todo el camino. Este servidor mantiene información global del estado de los conmutadores OBS y la disponibilidad de longitudes de onda, en base a la cual, procesa las solicitudes de conexión y envía confirmaciones a los nodos de frontera, para que luego de recibirlas puedan iniciar su transmisión. Este método, conocido también como WR-OBS (*conmutación óptica de ráfagas enrutadas por longitud de onda*), combina las funciones de OBS con la conmutación rápida de circuitos mediante la asignación y liberación dinámica de caminos de luz enrutados por longitud de onda a través de un núcleo óptico sin almacenamiento. Las ventajas potenciales de esta arquitectura en comparación con la OBS convencional y las WRONs estáticas, está en la provisión explícita de QoS para el primer caso y la rápida adaptación a cambios dinámicos de tráfico en redes ópticas y una utilización más eficiente de cada canal óptico para el segundo caso [59]. Por otro lado, un esquema de señalización distribuida incluye un planificador de ráfagas en cada nodo de la red, que es responsable de la reserva y asignación de recursos de una manera distribuida, en función de la información contenida en cada paquete de control.

El método distribuido es más flexible y escalable que el método centralizado, por lo tanto puede presentar una mejor adaptación en redes ópticas de gran tamaño y tráfico de datos a ráfagas.

2.5.1 Protocolos de reservación para OBS

Aunque el concepto de conmutación de ráfagas se remonta a principios de 1990, introducido inicialmente para sistemas de acceso múltiple por división de tiempo, TDMA (*Time Division Multiple Access*) y redes en modo de transferencia asincrónica, ATM (*Asynchronous Transfer Mode*), protocolos para redes WDM de alta velocidad no fueron propuestos sino hasta 1997. De hecho, los primeros protocolos formulados para redes OBS adaptaron las dos versiones del estándar referido como transferencia de bloques ATM, ABT (*ATM Block Transfer*) propuesto para la conmutación de ráfagas en ATM: con transmisión retardada, denominado TAW (*Tell-And-Wait*), y con transmisión inmediata, referida como TAG (*Tell-And-Go*) [27]. El protocolo TAW se basa en un modo de operación de dos vías con reserva de recursos extremo a extremo y ACK, mientras que el protocolo TAG opera en un modo de una sola vía, asignando recursos un tiempo antes del arribo de la ráfaga de datos que necesita ser ligeramente retrasada antes de cada nodo intermedio, para lo cual se inserta un retardo fijo en la ruta de los datos utilizando una línea de retardo de fibra FDL en cada puerto de entrada. Varias comparaciones de desempeño (p.ej. *throughput* y retardo) entre estos dos métodos conceptuales han encontrado que TAW supera a TAG cuando el retardo de propagación es despreciable con respecto a la longitud de la ráfaga (en términos de tiempo de transmisión). El resultado opuesto se mantiene cuando el retardo de propagación es significativo en comparación con la longitud de la ráfaga [128].

Posteriormente se propusieron nuevas alternativas a los protocolos antes mencionados, denominadas JIT (*Just-In-Time*) y JET (*Just-Enough-Time*), que corresponden a los esquemas de señalización más populares en redes OBS, siendo el último el más adecuado para este tipo de redes debido a todas las características que ofrece y que se presentan a continuación.

2.5.1.1 Just-In-Time (JIT)

La primera versión del protocolo *Just-In-Time* que puede considerarse una variante de TAW, empleaba un método de operación en dos vías utilizando un planificador central, encargado de controlar el inicio de la transmisión de ráfagas. La expresión *Just-In-Time* significa que para el momento de llegada de la ráfaga de datos a un nodo intermedio, la matriz de conmutación ya ha sido configurada. Debido a las limitaciones que presentan los esquemas centralizados en comparación con los esquemas distribuidos, una versión JIT denominada *Reservation with Just-in-Time* fue propuesta más tarde en [51], en la que una copia de las solicitudes de reserva se envía concurrentemente a todos los conmutadores de la red, donde cada planificador no sólo está sincronizado en tiempo con todos los demás sino que también comparte la misma información global de estado de enlace, lo que hace difícil su implementación.

Es así, que surge otra versión distribuida de JIT propuesta en [116] que utiliza un método de reserva de recursos salto por salto considerando algunas características del protocolo JET, que se basa en un esquema de reserva inmediata y liberación explícita (podría también ser implícita), donde cada nodo OBS configura su conmutador óptico para la ráfaga entrante a partir de la recepción de su correspondiente paquete de control, conduciendo a una reserva anticipada del canal antes de la llegada de la ráfaga de datos y por ende a una asignación ineficiente de los recursos, como se puede apreciar en la Figura 2.23, donde una vez que el primer nodo de *core* recibe un mensaje SETUP y lo procesa en el tiempo t_p, envía un mensaje CALL PROCEDING hacia el origen, para indicar que la configuración de la conexión está en camino hacia el destino, que incluye un parámetro relacionado con el retardo de tiempo que debería esperar el origen antes de transmitir la ráfaga de datos, el cual es estimado (p. ej. mediante un algoritmo de enrutamiento implementado adecuadamente) en base al número de saltos hacia el destino y al tiempo de reconfiguración t_c de las matrices de conmutación óptica a lo largo de la ruta.

En general, el mínimo tiempo de *offset* requerido es configurado a $h.T_p + T_c$, donde h representa el número de saltos en la ruta. Cuando el nodo de ingreso recibe el mensaje CALL_PROCEEDING, inicia la transmisión de la ráfaga de datos luego del tiempo de *offset*

estimado t_d. Además, un nodo OBS que recibe el mensaje SETUP intentará reservar una longitud de onda en el puerto de salida y reenviar el mensaje SETUP hacia el siguiente salto, mientras que la configuración de la matriz de conmutación óptica se ejecuta en paralelo con la propagación del siguiente salto. Cuando el nodo destino recibe el mensaje SETUP, responderá con un mensaje CONNECT que viaja de regreso hacia el origen para notificar del establecimiento exitoso de la conexión. Por otro lado, una vez finalizada la transmisión de la ráfaga de datos, el nodo de ingreso generará un mensaje RELEASE para liberar todas las longitudes de onda reservadas.

Debido a que no es posible utilizar el canal durante el tiempo entre el arribo del paquete de control y su correspondiente ráfaga de datos, inclusive hasta que concluya el proceso de liberación de recursos, este esquema resulta en una mayor probabilidad de pérdida de ráfagas. Su principal ventaja es su simplicidad, puesto que los nodos intermedios no requieren conocer la información del tiempo de *offset* que podría ser transportada en los paquetes de control, y que complicaría su implementación como en el caso del protocolo JET.

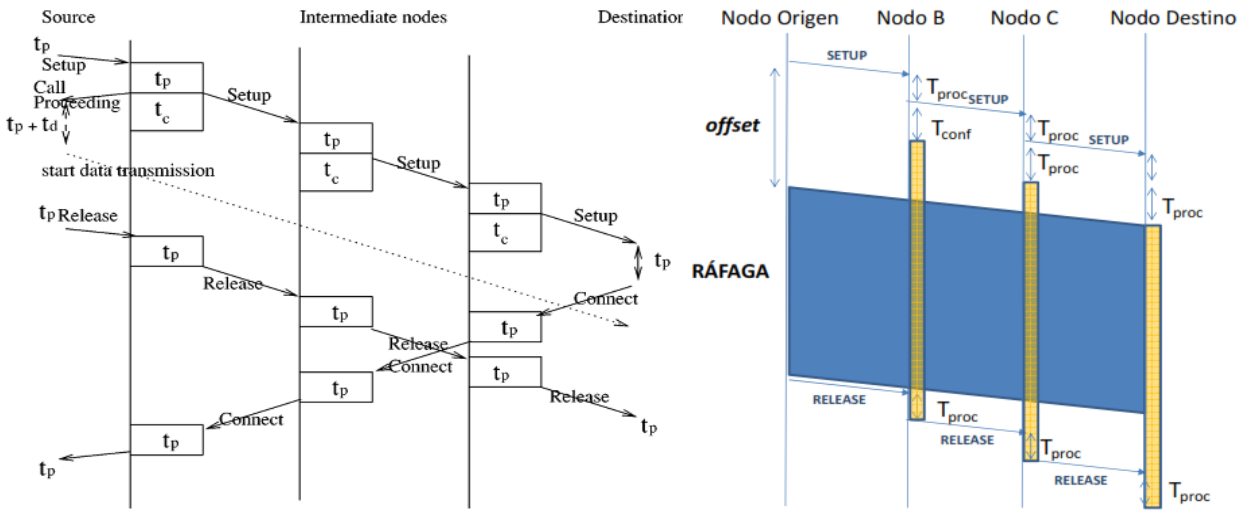


Figura 2.23 Esquema de señalización Just-In-Time (JIT) [41][116]

Adicionalmente, dado que la simplicidad es una característica relevante en cualquier esquema de señalización, una versión mejorada de JIT en términos de planificación de canales de datos

y reducción en cuanto al tiempo de reserva, denominada E-JIT (*Enhanced-JIT*), fue propuesta en [95] donde manteniendo su simplicidad de implementación se consiguió optimizar la utilización del canal y reducir la probabilidad de pérdida de ráfagas.

Incluso en un intento de mejorar aún más esta versión, se planteó en [96] una variante conocida como E-JIT plus que mostró un mejor desempeño que sus predecesoras, aunque resultó no ser muy escalable puesto que a medida que incrementaba el tráfico crecía también la probabilidad de pérdida de ráfagas, conduciendo a un menor desempeño de la red.

2.5.1.2 Just-Enough-Time (JET)

Al igual que su contraparte JIT expuesta anteriormente, el protocolo JET [124] se basa también en el principio de *Tell-And-Go* con la diferencia que emplea métodos de reserva retardada y liberación implícita, donde cada nodo OBS configura su matriz de conmutación óptica justo antes de la llegada de la ráfaga de datos y libera los recursos automáticamente al finalizar la duración de la misma, permitiendo de esta manera una asignación más eficiente del canal y por ende una menor probabilidad de bloqueo, así como también una disminución de la latencia.

Sin embargo, los nodos intermedios deben ser capaces de mantener consistentemente los tiempos de inicio y finalización de cada una de las ráfagas de datos sobre los distintos canales ópticos de cada puerto, en función de la información del tiempo de *offset* y longitud de la ráfaga que debe ser transportada necesariamente en los paquetes de control. Esto dota a los planificadores de mayor inteligencia para la programación de ráfagas, permitiendo incluso, detectar situaciones en las que no ocurre conflicto en la transmisión como se puede apreciar en la Figura 2.24, aunque el tiempo de inicio de una nueva ráfaga puede ser anterior al tiempo de finalización de una ráfaga ya aceptada, lo que se traduce en la posibilidad de planificar la nueva ráfaga entre dos reservas existentes. No obstante, todo esto contribuye a un incremento significativo en cuanto a la complejidad de su algoritmo en comparación con su contraparte JIT.

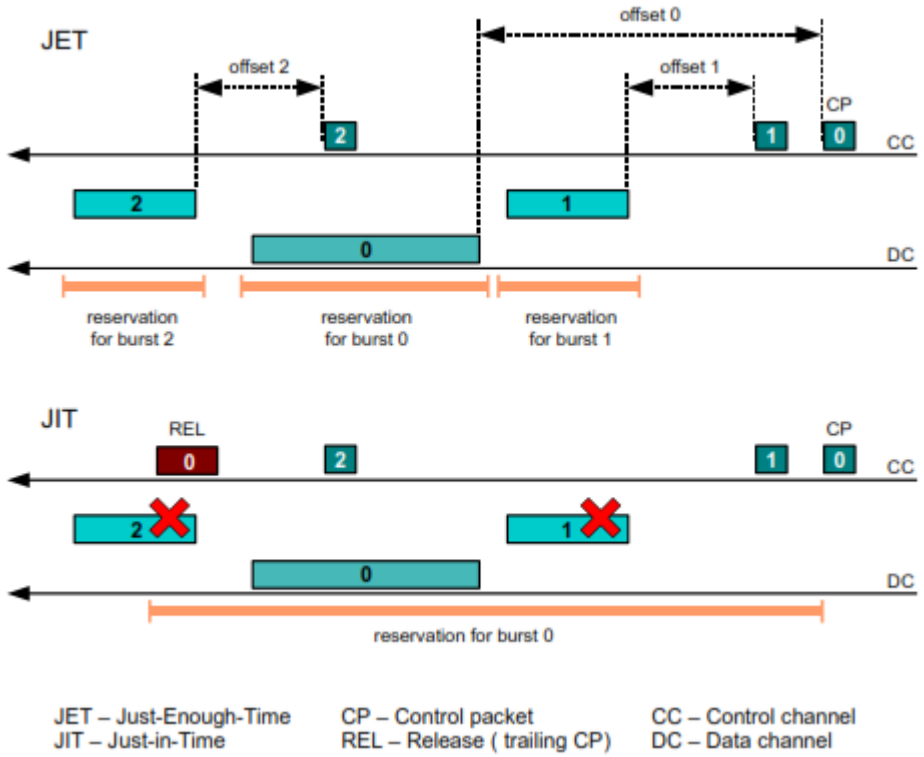


Figura 2.24 Comparación entre los esquema de señalización Just-Enough-Time (JET) y Just-In-Time (JIT) [27]

El funcionamiento general de este protocolo se ilustra en la Figura 2.25, donde el nodo origen envía un paquete de control hacia el nodo de destino, que se procesa electrónicamente en cada nodo subsiguiente, con el fin de establecer una ruta completamente óptica para la ráfaga de datos correspondiente. Si el intento de reserva es exitoso, los conmutadores son configurados a lo largo del camino antes del arribo de la ráfaga, la cual espera mientras tanto, en un *buffer* electrónico en el origen hasta que transcurra el tiempo de *offset* predeterminado, para posteriormente iniciar su transmisión en el dominio óptico a través de la longitud de onda seleccionada. El tiempo de *offset* (OT) se calcula en base al número de saltos desde el origen hasta el destino y al tiempo de reconfiguración del conmutador de un nodo central de la siguiente manera: OT=h*d+ST, donde h es el número de saltos entre el origen y el destino, d es el tiempo de procesamiento del paquete de control por nodo, y ST es el tiempo de reconfiguración de la matriz de conmutación [59]. Si en cualquier nodo intermedio la reserva no es exitosa, la ráfaga se descarta completamente, a menos que exista una política de

resolución de contenciones, que permita reducir la probabilidad de pérdidas de ráfagas que constituye un parámetro que impacta directamente en el desempeño de las redes OBS.

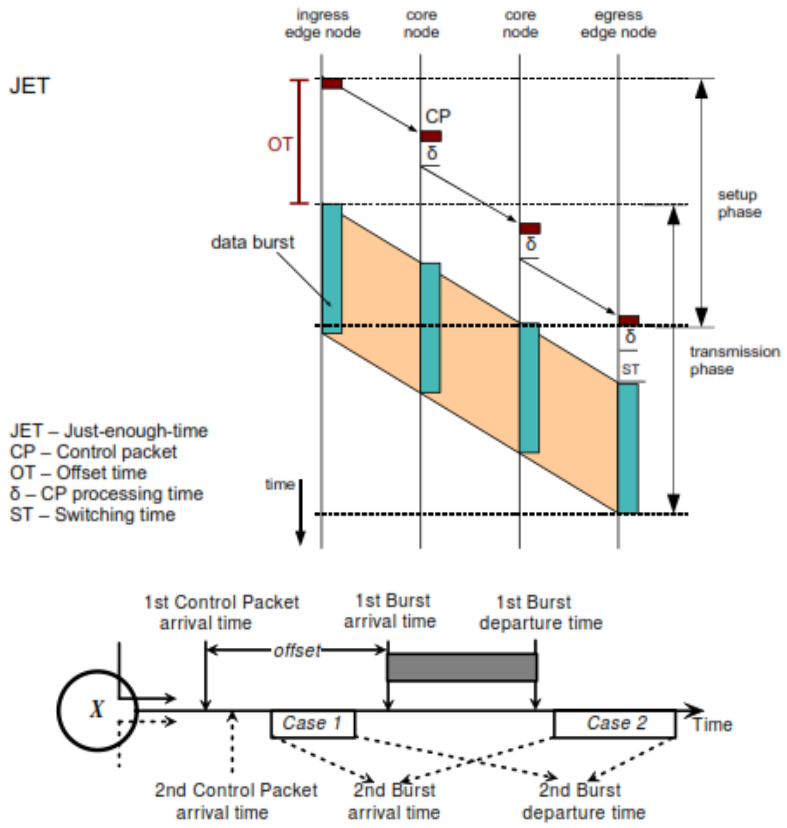


Figura 2.25 Esquema de señalización Just-Enough-Time (JET) [22][27]

Otra ventaja de JET es que puede reducir la probabilidad de pérdida de ráfagas y mejorar el performance aún más, retrasando la ráfaga de datos por la cantidad de tiempo necesaria hasta que el canal llegue a estar disponible. Como se puede apreciar en la Figura 2.26, es claro que el protocolo no podría reservar el canal para la segunda ráfaga de datos cuyo paquete de control arriba en el instante t_2' (y su correspondiente ráfaga en el instante t_2, donde $t_1 < t_2 < t_1+l_1$), a menos que pudiera retrasarla un lapso de tiempo como mínimo de $d_{min} = t_1+l_1-t_2$. Esta característica se puede conseguir con el uso de líneas de retardo ópticas (FDLs, *Fiber Delay Lines*), de manera que el nodo intermedio que detecta esta situación determina si es posible proveer un retardo suficiente d, tal que sea lo más pequeño posible pero cumpliendo la

condición $d \geq d_{min}$, evitando así el descarte inminente de la ráfaga de datos. La transmisión del paquete de control hacia el siguiente nodo, también puede retrasarse d unidades de tiempo para mantener un apropiado tiempo de *offset* para los saltos restantes a lo largo del camino. Vale la pena señalar que la reserva retardada, se aplicará a las d unidades de retardo en el nodo correspondiente, de manera similar a como se aplicaba para reservar el ancho de banda del canal de salida, es decir que la reserva para el retardo será realizada sólo desde t_2 hasta $t_2 + l_2$ (en lugar de t_2' en adelante).

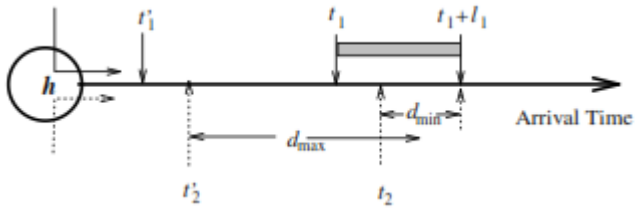


Figura 2.26 Reservación retardada de buffer en Just-Enough-Time (JET) [124]

En caso de que el retardo requerido no esté disponible, el nodo intermedio puede decidir descartar la ráfaga sin reservar el *buffer* óptico requerido para el efecto. Por otro lado, es conveniente indicar que sin el mecanismo de reserva retardada, el nodo no puede conocer qué cantidad de retardo es realmente necesaria, por consiguiente, tendrá que reservar el máximo disponible d_{max}, el momento de realizar la reserva. Tal reserva a ciegas resultará en un cierto desperdicio de la capacidad de *buffering* cuando d_{max} sea excesivo ($t_2' + d_{max} > t_1 + l_1$) y lo que es peor, resultará en un completo desperdicio de la capacidad de *buffering* cuando d_{max} sea insuficiente ($t_2' + d_{max} < t_1 + l_1$).

2.6 Planificación de canal [21,22][32][52,59][75][110,119]

En una red OBS, las ráfagas de datos generalmente pueden tener diferentes tamaños y tiempos de *offset*, pudiendo arribar no en el mismo orden que sus paquetes de control, lo cual conduce a pensar en una fragmentación del ancho de banda en espacio y tiempo, donde los canales

ópticos a través de los cuales finalmente se transmite la información incluyen huecos o vacíos[24] entre ráfagas consecutivas. Es así que cuando una ráfaga llega a un nodo, se le debe asignar una longitud de onda disponible sobre un enlace de salida durante un determinado tiempo, para su transmisión a lo largo del dominio OBS. El objetivo fundamental de la planificación es minimizar los vacíos en cada programación de canal, es así que un algoritmo de planificación eficiente, debería ser capaz de ajustar un período de reserva nuevo en un intervalo vacío existente siempre que sea posible, a fin de incrementar la utilización del ancho de banda y reducir la tasa de pérdida de datos [119].

La planificación de los canales en redes OBS difiere de la planificación tradicional de paquetes IP, ya que en este caso los enrutadores cuentan con *buffers* electrónicos que permiten almacenar temporalmente los paquetes entrantes y planificarlos adecuadamente en función de su destino y opcionalmente de sus requerimientos de calidad de servicio, a través del puerto de salida correspondiente. En OBS por el contrario, una vez que llega una ráfaga de datos a un nodo central se la debe enviar directamente al siguiente nodo sin ningún tipo de almacenamiento electrónico.

Cuando un paquete de control llega a un nodo central, se invoca un algoritmo de planificación de canal para asignar la ráfaga no programada a un canal de datos óptico de salida. El planificador de canal obtiene el tiempo estimado de llegada de la ráfaga no programada y su duración desde su correspondiente BHP. El algoritmo puede necesitar mantener la información del último instante disponible no planificado (LAUT)[25] u horizonte, huecos y vacíos sobre cada canal de datos de salida.

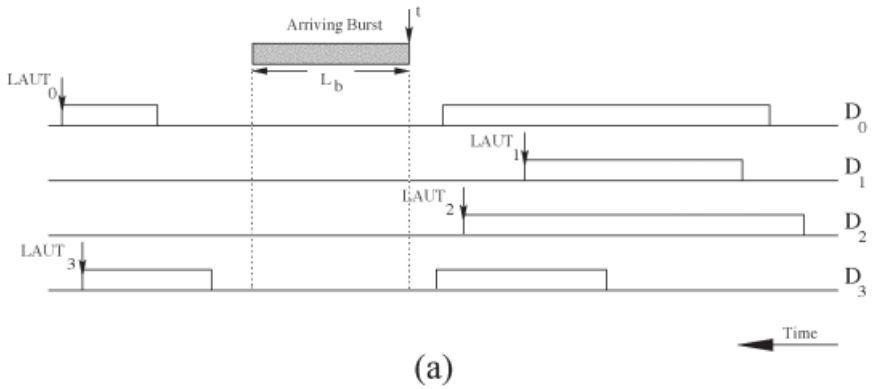
Hasta el momento se han definido una serie de algoritmos de planificación de ráfagas, que de manera general se pueden dividir en dos categorías: algoritmos de planificación sin relleno de vacíos y algoritmos de planificación con relleno de vacíos [75]. Su principal diferencia radica en el tipo y cantidad de la información de estado que deben mantener en un nodo acerca de

[24] Un hueco o vacío es la diferencia en tiempo entre la llegada de la ráfaga no planificada y el tiempo de finalización de la ráfaga planificada previarmente, o dicho de otra manera, es la duración no programada (tiempo de inactividad) entre dos ráfagas planificadas sobre un mismo canal de datos.
[25] Tradicionalmente, el LAUT de un canal es el instante más temprano en el que el canal de datos está disponible para que una ráfaga de datos no programada sea planificada.

cada canal, dando como resultado ineficiencia en ancho de banda pero simplicidad de implementación para el caso de los algoritmos sin relleno de vacíos y una mejor utilización del ancho de banda pero mayor complejidad computacional para el caso de los algoritmos con relleno de vacíos.

La siguiente información es utilizada por el planificador para la mayor parte de los algoritmos de planificación [59]:

- L_b: Duración de la longitud de la ráfaga no planificada.
- tub: Tiempo de llegada de la ráfaga no planificada.
- W: Máximo número de canales de datos de salida.
- Ni,j: Máximo número de ráfagas de datos planificadas sobre un canal de datos.
- Di: i-ésimo canal de datos de salida.
- LAUTi: LAUT del i-ésimo canal de datos, i=1, 2,…., W, para los algoritmos de planificación sin relleno de vacíos.
- Si,j y Ei,j: Tiempos de inicio y fin de cada ráfaga planificada j, en cada canal de datos i, para los algoritmos de planificación con relleno de vacíos.
- Gapi: Si el canal está disponible, Gap es la diferencia entre tub y LAUTi para los algoritmos de planificación sin relleno de vacíos, y es la diferencia entre tub y Ei,j de la ráfaga planificada anteriormente j, para los algoritmos de planificación con relleno de vacíos. Si el canal está ocupado, Gapi es fijado a 0. La información de Gap es útil para seleccionar un canal para el caso en el cual más de un canal esté disponible.



(a)

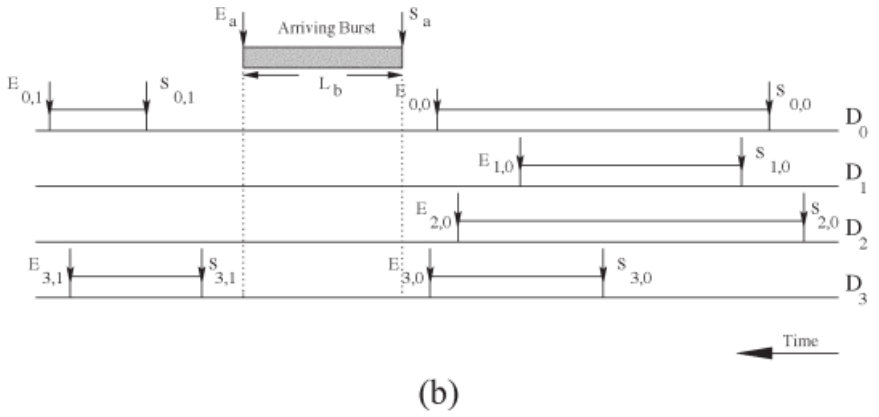


Figura 2.27 Planificación de canal de datos (a) sin relleno de vacíos (b) con relleno de vacíos [59]

En los algoritmos de planificación de canal sin relleno de vacíos, el $LAUT_i$ de cada canal de datos D_i, i=0, 1,…., W, es mantenido por el planificador de canal, mientras que en algoritmos con relleno de vacíos, el tiempo de inicio, $S_{i,j}$ y el tiempo final, $E_{i,j}$ son mantenidos para cada ráfaga en cada canal de datos, donde, i=0, 1, …., W, es el i-ésimo canal de datos y j=0,1, …, $N_{i,j}$ corresponde a la j-ésima ráfaga en el canal i; tal y como se puede apreciar en la Figura 2.27.

En base a la descripción general presentada, a continuación se exponen los algoritmos de planificación más habituales propuestos para redes OBS.

2.6.1 First Fit Unscheduled Channel (FFUC)

El algoritmo de planificación FFUC mantiene un registro del LAUT (u horizonte) sobre cada canal de datos. Una longitud de onda es considerada para cada ráfaga que llega cuando el instante no planificado (LAUT) del canal de datos es menor que el tiempo de llegada de la ráfaga. El algoritmo FFUC busca todos los canales en un orden fijo y asigna el primer canal disponible a la nueva ráfaga entrante. Su principal ventaja es la facilidad en cuanto a su implementación, dado que necesita mantener un solo valor ($LAUT_i$) por cada canal.

Sin embargo, esta característica resulta negativa en el sentido de que conduce a una elevada probabilidad de pérdidas de ráfagas.

El algoritmo FFUC se ilustra en la Figura 2.28 (a), donde en base al valor de $LAUT_i$, se puede observar que los canales de datos D_1 y D_2 están disponibles para la duración de la ráfaga no planificada. Si los canales se encuentran ordenados según el índice de sus longitudes de onda (D_0, D_1,…, D_W), la ráfaga entrante es planificada sobre el canal de salida D_1. La complejidad en tiempo del algoritmo FFUC es O(log W).

2.6.2 Horizon o Latest Available Unscheduled Channel (LAUC)

El algoritmo de planificación LAUC u horizonte [110] es similar al caso anterior en el sentido que mantiene únicamente el registro del LAUT (u horizonte) de cada canal de datos. Su diferencia radica en que asigna la nueva ráfaga al último canal de datos disponible no planificado, que lo define en base al mínimo Gap de entre todos los canales candidatos a ser utilizables, es decir, aquellos cuyo LAUT precede al tiempo de arribo de la ráfaga entrante. La idea básica de este algoritmo es minimizar los Gaps de ancho de banda creados como resultado de nuevas reservaciones.

El algoritmo LAUC se ilustra en la Figura 2.28 (a), donde en base al valor de $LAUT_i$, se puede determinar que los canales de datos D_1 y D_2 están disponibles para la duración de la ráfaga no planificada, pero en vista de que $Gap_2 < Gap_1$, la ráfaga entrante es planificada sobre el canal de salida D_2. La complejidad en tiempo del algoritmo LAUC es O(W), que presenta mejores resultados en comparación con FFUC, aunque sigue sufriendo de baja utilización de ancho de banda y alta probabilidad de pérdida de ráfagas debido a los huecos existentes entre ráfagas sucesivas que al igual que en el caso anterior no se pueden utilizar, malgastando recursos de la red que se podrían aprovechar para optimizar su desempeño.

2.6.3 First Fit Unscheduled Channel with Void Filling (FFUC-VF)

El algoritmo de planificación FFUC-VF mantiene un registro de los tiempos de inicio y fin para cada ráfaga planificada sobre cada canal de datos. El objetivo principal de este algoritmo es utilizar los espacios inactivos (huecos) entre asignaciones de dos ráfagas consecutivas, para lo cual, el primer canal con un vacío adecuado es seleccionado para transportar la nueva ráfaga.

El algoritmo FFUC-VF se ilustra en la Figura 2.28 (b), donde en base a los valores de $S_{i,j}$ y $E_{i,j}$, se puede observar que todos los canales de datos D_0, D_1, D_2 y D_3 están disponibles para la duración de la ráfaga no planificada. De igual manera que en el caso de FFUC, si los canales están ordenados según el índice de sus longitudes de onda (D_0, D_1,…, D_W), la ráfaga entrante es planificada sobre el primer canal de salida, es decir, D_0. Cabe mencionar además, que si N_b corresponde al número de ráfagas planificadas actualmente sobre cada canal de datos, se puede utilizar un algoritmo de búsqueda binaria para encontrar el canal de datos apropiado. Por lo tanto, la complejidad en tiempo del algoritmo FFUC-VF es $O(W \log N_b)$, que es superior en comparación con los dos casos anteriores, sin embargo resulta en un mejor desempeño de la red.

2.6.4 Latest Available Unscheduled Channel with Void Filling (LAUC-VF)

El algoritmo de planificación LAUC-VF, similar al caso de FFUC-VF, mantiene un registro de los tiempos de inicio y fin para cada ráfaga planificada sobre cada canal de datos, con el fin de utilizar los vacíos entre asignaciones de dos ráfagas sucesivas. El canal con un vacío adecuado que tenga el mínimo Gap es elegido para transmitir la nueva ráfaga. El algoritmo LAUC-VF se ilustra en la Figura 2.28 (b), donde en base a los valores de $S_{i,j}$ y $E_{i,j}$, se puede determinar que todos los canales de datos D_0, D_1, D_2 y D_3 están disponibles para la duración de la ráfaga no planificada, sin embargo, como Gap_3 corresponde al mínimo de todos

$Gap_3 < Gap_0 < Gap_2 < Gap_1$, la ráfaga entrante es planificada en el canal D_3. De la misma manera que en el caso de FFUC-VF, si N_b es el número de ráfagas planificadas actualmente sobre cada canal de datos, se puede utilizar un algoritmo de búsqueda binaria para seleccionar el canal de datos más adecuado. Por lo tanto, la complejidad en tiempo del algoritmo LAUC-VF es $O(W \log N_b)$, que aunque es la misma que la de su contraparte FFUC-VF, permite optimizar de mejora manera el uso del ancho de banda reduciendo a la vez la probabilidad de pérdidas de ráfagas.

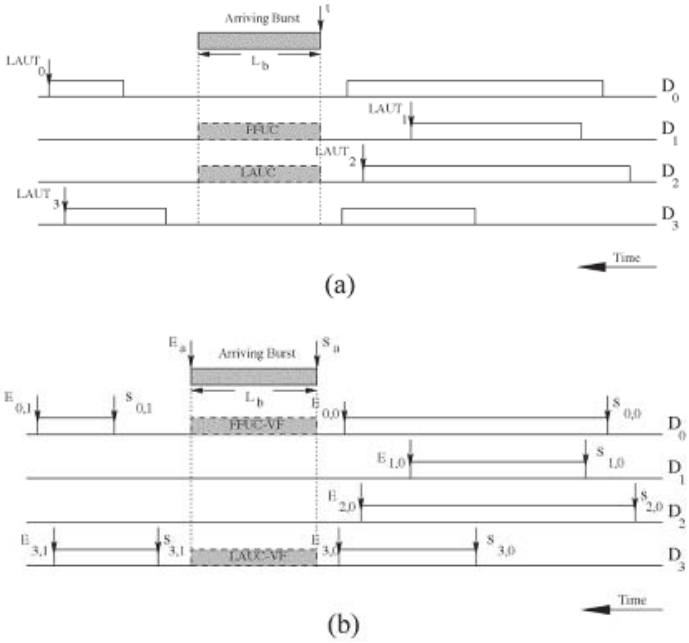


Figura 2.28 Algoritmos de planificación de canal (a) sin relleno de vacíos (FFUC y LAUC), y (b) con relleno de vacíos (FFUC-VF y LAUC-VF) [59]

A continuación se mencionan algunas optimizaciones efectuadas sobre los algoritmos de planificación descritos anteriormente. En [52], se propone un algoritmo denominado MVUC (*Minimizing Voids Unscheduled Channel*) con el fin de minimizar los huecos generados por las ráfagas entrantes en cada nodo central, seleccionando el canal de datos en el que un hueco generado recientemente después de la transmisión de una ráfaga llega a ser mínimo. Los autores concluyen mediante simulaciones que este algoritmo opera mejor que LAUC-VF en términos de pérdida de datos. La Figura 2.29 muestra una comparativa entre el algoritmo MVUC y el convencional LAUC-VF.

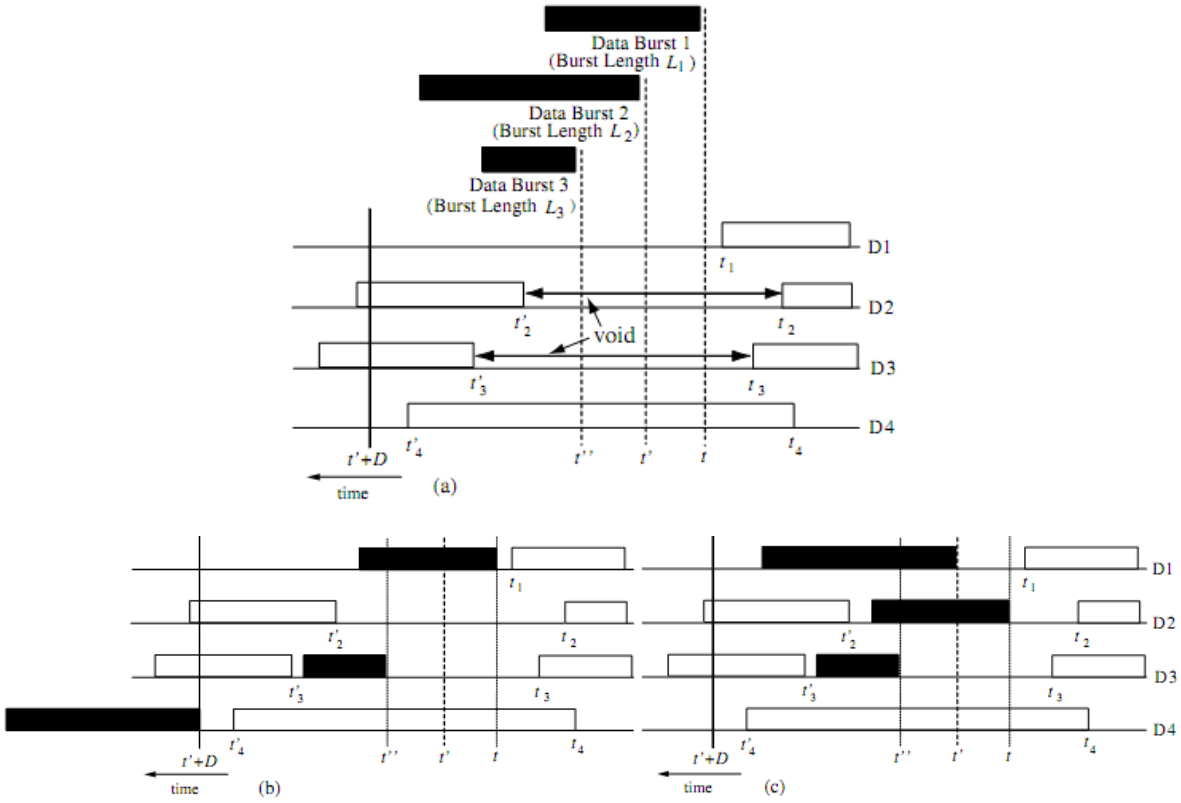


Figura 2.29 a) Arribo de ráfagas de datos b) LAUC-VF y c) MVUC [52]

Posteriormente, algunas variantes del algoritmo LAUC-VF denominadas Min-SV (*Minimum Starting Void*), Min-EV (*Minimum Ending Void*), y Best Fit fueron propuestas en [119]. Min-SV, utiliza el mismo criterio de planificación que LAUC-VF, pero su factor diferenciador es que la estructura de datos la construye utilizando un árbol de búsqueda binario balanceado, mediante el cual, se alcanza una tasa de pérdidas tan baja como en LAUC-VF pero con un tiempo de procesamiento mucho más rápido. De hecho, su velocidad puede ser casi la misma que LAUC (que tiene una tasa de pérdidas mucho mayor). Por otro lado, Min-EV intenta minimizar el hueco generado entre el final de la nueva ráfaga y la siguiente reserva existente, mientras que Best Fit trata de minimizar la longitud total de inicio y fin de vacíos generados después de la reserva.

El desempeño de estos algoritmos de planificación se compara en [119], donde se muestra que LAUC-VF, Min-SV, Min-EV y Best Fit presentan una utilización comparable en cuanto al

ancho de banda (o probabilidad de pérdida), que es mucho mayor (o menor) que los algoritmos basados en Horizon.

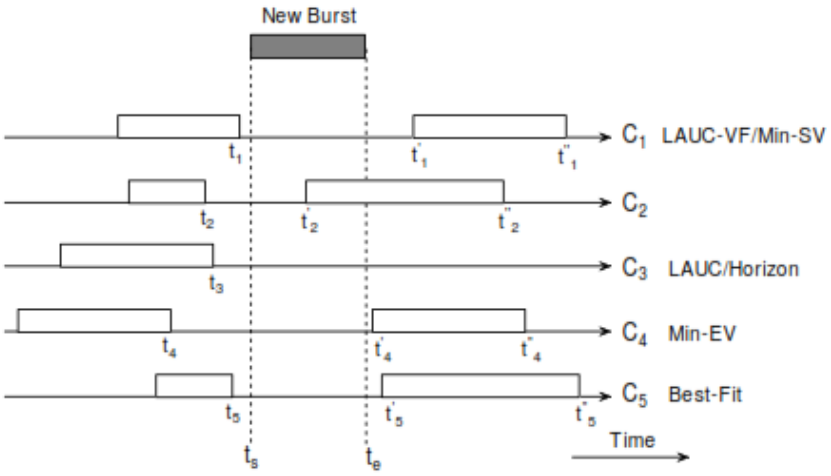


Figura 2.30 Resultados de los algoritmos de planificación [22]

La Figura 2.30 ilustra los resultados de estos algoritmos de planificación, mientras que la Tabla 2.4 presenta un resumen descriptivo que compara su desempeño en general.

Tabla 2.4 Comparación de los diferentes algoritmos de planificación [22]

Algoritmo de Planificación	**Complejidad Temporal**	**Información de estado**	**Utilización de ancho de banda**	**Probabilidad de pérdida de ráfagas**
LAUC	O(W)	$Horizon_i$	Bajo	Alta
LAUC-VF	O(W log M)	S_{ij}, E_{ij}	Alto	Baja
Min-SV/EV	O(log M)	S_{ij}, E_{ij}	Alto	Baja
Best Fit	O ($\log^2$ M)	S_{ij}, E_{ij}	Alto	Baja

Donde:

W = Número de longitudes de onda en cada puerto de salida

M = Máximo número de ráfagas de datos (o reservaciones) en todos los canales

$Horizon_i$ =El horizonte del i-ésimo canal de datos

S_{ij} y E_{ij} = Tiempos de inicio y fin de la j-ésima reservación en el canal i

Es claro que estos algoritmos consideran una planificación individual de las ráfagas, lo cual condujo a explorar nuevas alternativas bajo el concepto de una planificación agrupada de ráfagas, donde cada una es representada por un intervalo de tiempo. Por lo tanto, al tener conocimiento de un conjunto de intervalos de tiempo, la idea es poder ajustarlos eficientemente a una línea de canales en el tiempo. Los algoritmos denominados LAW (*Look-ahead Window*) [32] y planificación en grupo para OBS [21], se basan en este concepto, que se beneficia de la separación entre las ráfagas de datos y los paquetes de control, de tal manera que, al recibir los BHPs un tiempo de *offset* antes que sus correspondientes ráfagas de datos, es posible construir una ventana de agrupación con un tamaño de W unidades de tiempo, como se ilustra en el Figura 2.31.

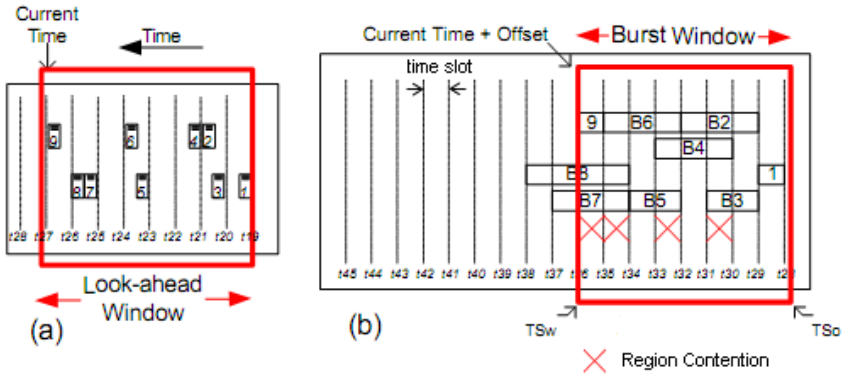


Figura 2.31 Construcción de la ventana de agrupación para todas las ráfagas que se dirigen hacia el mismo puerto de un conmutador con 2 canales de datos [32]

Se considera que esta visión colectiva, resulta en decisiones más eficientes con respecto a qué ráfagas se deberían descartar o planificar. El criterio básico de la programación de grupos es el de retrasar brevemente la programación de una ráfaga, de forma que se pueda tomar una mejor

decisión sobre una serie de ellas en conjunto; es así que, para compensar cualquier instante perdido en el tiempo de *offset*, se sugiere la utilización de FDLs, sobre las cuales existe también una importante investigación en el tema de la planificación sobre redes OBS [59].

Adicionalmente, varios algoritmos basados en la segmentación de ráfagas (tema que se abordará en la siguiente sección), con y sin FDLs, han demostrado alcanzar una pérdida significativamente menor que los algoritmos de planificación anteriores [59].

2.7 Resolución de contenciones [8][19][22,24,26,27][38][41][59][64,66,67] [69][73,75][89][114][139]

Debido a que las redes ópticas conmutadas por ráfagas se proponen como una alternativa de transporte óptico no orientado a la conexión, debido a que utilizan esquemas de señalización en una vía, que no permiten garantizar una transmisión confiable, es posible que múltiples ráfagas puedan colisionar en los nodos intermedios por el hecho de competir por el mismo recurso a la vez. Este evento se conoce como contención (o contienda) que resulta en pérdidas de ráfagas y se presenta de forma temporal con una naturaleza aleatoria, pero únicamente cuando dos o más ráfagas provenientes de distintos puertos de entrada, están destinadas hacia el mismo canal óptico de un determinado puerto de salida simultáneamente.

La contención es la principal causa de pérdidas de ráfagas en redes OBS, por lo tanto la resolución de contiendas es un objetivo de diseño de suma importancia debido a que permite reducir la probabilidad de pérdida de ráfagas, que es un parámetro crucial en el desempeño de redes OBS debido a que puede degradar las aplicaciones que utilizan este tipo de sistemas. Por ejemplo, siempre que se pierde una ráfaga, la capa TCP sufre una pérdida de múltiples segmentos de la cual todos podrían ser de la misma ventana de congestión. En tal caso, dependiendo del tamaño de la ventana, el transmisor TCP podría incluso ir al estado de *timeout*. Múltiples pérdidas de ráfagas conducirían a un *throughput* muy bajo, resultando además en un desperdicio de recursos debido a las retransmisiones propias de la capa TCP, lo cual se analizará como con mayor detalle en el capítulo III.

En contraste a lo anterior, las redes tradicionales de conmutación de paquetes electrónicos, resuelven normalmente las contenciones mediante el almacenamiento temporal (*buffering*) de los paquetes en memorias RAM; sin embargo, en el dominio óptico, no existe un equivalente a la memoria RAM electrónica. Es así que varias investigaciones han conducido a diferentes alternativas para tratar este tema sobre redes OBS, planteando que la contención entre ráfagas se puede resolver de cuatro maneras conocidas como: deflexión (o desvío), descarte, priorización o segmentación [22].

Deflexión (**o desvío**).- Es el resultado de enviar una ráfaga por un canal de salida diferente del que estaba previsto. Puesto que la contención puede suceder sólo cuando las ráfagas compiten por la misma longitud de onda sobre el mismo puerto de salida simultáneamente, la deflexión se puede aplicar en los dominios de 1) longitud de onda, 2) espacio y/o 3) tiempo.

1) La ráfaga que contiende es enviada sobre otra longitud de onda utilizando el mecanismo de **conversión de longitud de onda.**
2) La ráfaga que contiende es desviada a un puerto de salida diferente y luego sigue una ruta alterna hacia su destino. Esta técnica es referida como **enrutamiento por deflexión.**
3) La ráfaga que contiende es retrasada durante un período de tiempo determinado (normalmente fijo), mediante el uso de FDLs. Este método es conocido como ***buffering*** **óptico.**

Es posible permitir que una ráfaga entrante se anticipe a una ráfaga existente (con una reserva ya asignada), en base a una prioridad o perfil de tráfico. Existe además un método denominado segmentación de ráfagas que permite dividir la ráfaga entrante en múltiples segmentos, y cada segmento puede ser luego desviado, descartado o anticipado.

Descarte.- Si una ráfaga que contiende no puede ser desviada debido a la falta de disponibilidad de recursos, yasea de longitud de onda, puerto de salida o FDL, la pérdida de datos llega a ser inevitable, siendo el método más común descartar la ráfaga de datos entrante (opción no preferente).

Priorización.- Como una alternativa a la técnica de descarte, es posible que la ráfaga entrante tenga preferencia sobre una ráfaga existente en función de una prioridad o perfil de tráfico

definido, pudiendo de esta manera conceder los recursos de una reserva existente a una ráfaga prioritaria.

Segmentación.- Adicionalmente, es factible dividir la ráfaga entrante o la ráfaga existente en múltiples segmentos, y cada segmento luego puede ser desviado, descartado o priorizado.

Algunos de estos algoritmos que se expondrán más detalladamente en esta sección se presentan gráficamente en la Figura 2.32.

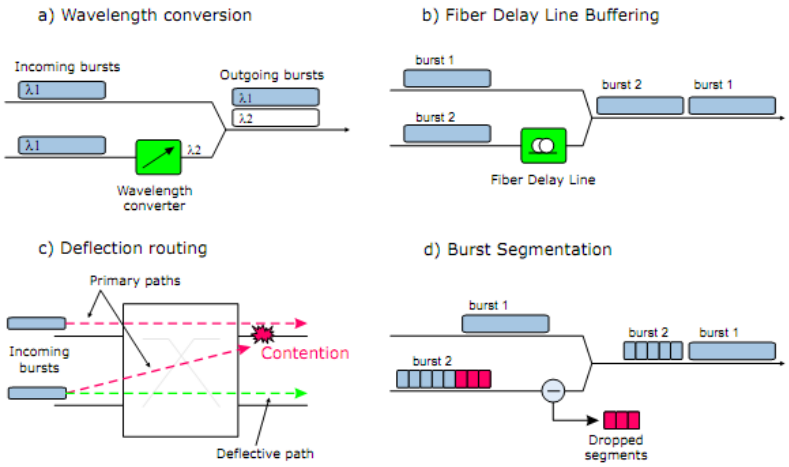


Figura 2.32 Mecanismos de resolución de contenciones [66]

Se debe mencionar también, que algunas de estas técnicas para la resolución de contenciones se pueden combinar y aplicar de manera conjunta. Por otro lado, una solución alternativa e interesante para mitigar la contención en redes OBS sin requerir ninguno de los mecanismos de resolución de contención mencionados, es la asignación cuidadosa de longitudes de onda a las ráfagas de los usuarios OBS en el borde de la red [75]. Varias heurísticas de asignación de longitud de onda adaptativas y no adaptativas se han explorado para redes OBS en ausencia de técnicas de resolución de contenciones, demostrando que las elecciones inteligentes en la asignación de longitudes de onda en el borde de la red OBS pueden tener un impacto significativo en la probabilidad de pérdidas de ráfagas. Entre las heurísticas propuestas, las de mayor desempeño tienen el potencial de mejorar este parámetro en hasta dos órdenes de

magnitud en comparación con las dos heurísticas más simples de asignación de longitud de onda aleatoria y de primer ajuste (*first fit*) [75].

2.7.1 Almacenamiento óptico (*buffering*)

En principio, las memorias RAM electrónicas han sido utilizadas tradicionalmente para el almacenamiento temporal de paquetes. Incluso, los primeros prototipos de conmutación fotónica propuestos, adoptaron este tipo de memorias para el *buffering* de paquetes ópticos [109]. Sin embargo, las memorias RAM tienen una velocidad de acceso limitada que finalmente restringen la tasa de transferencia y capacidad de los sistemas ópticos. Sin embargo, el *buffering* electrónico se puede considerar como una alternativa dentro de sistemas OBS con la desventaja de que la red perdería transparencia y cada nodo debería incluir capacidades de conmutación o enrutamiento electrónico, que requieren de conversiones O/E/O y la posibilidad de memorias electrónicas que mantengan el ritmo de las velocidades de transmisión óptica.

Es así que, como un intento de trasladar el concepto de *buffering* al dominio óptico, se han propuesto varias alternativas basadas principalmente en elementos conocidos como Líneas de Retardo de Fibra (FDL), que se pueden implementar en simples y múltiples etapas con el fin de retrasar las ráfagas durante un período de tiempo limitado, que puede ser fijo o incluso variable (múltiplo de una unidad de tiempo). Algunas investigaciones han conducido a técnicas para diseñar grandes *buffers* sin un gran número de FDLs, incluyendo esquemas con múltiples etapas en cascada y el uso de los denominados *buffers* no degenerados, en los que la longitud de las líneas de retardo puede ser mayor que el número de líneas de retardo en el *buffer*. Este último método resulta en menores probabilidades de pérdida, pero no garantiza el ordenamiento correcto de los paquetes [59].

El comportamiento de los FDLs es bastante diferente de los *buffers* electrónicos convencionales. Un *buffer* electrónico puede aceptar una ráfaga entrante si existe suficiente espacio disponible, en cuyo caso puede almacenar la ráfaga por cualquier período de tiempo

arbitrario. En contraste, un FDL permite retrasar (no almacenar) una ráfaga por un período máximo de tiempo y por lo tanto proporciona un retardo determinístico para una ráfaga entrante. Además, los FDLs presentan la denominada propiedad de *balking*, que hace referencia al hecho de que una ráfaga entrante debe ser descartada si el retardo máximo proporcionado por el FDL no es suficiente para evitar la contención con una ráfaga que está siendo transmitida actualmente sobre un puerto de salida dado [73].

Estas propiedades generan una dificultad importante en el diseño de dispositivos de almacenamiento óptico, que se traduce en cómo implementar *buffers* eficientes de longitud variable a partir de FDLs de longitud fija. Uno de los métodos más populares para la construcción de tales *buffers* ópticos es utilizar líneas de retardo conmutadas (SDL, *Switched Delay Line*), que comprenden una combinación de FDLs concatenados y conmutadores fotónicos 2x2 de barras cruzadas (*cross-bar*), que son controlados electrónicamente sobre una base de *slots* fotónicos en función de la información proporcionada por los receptores locales RX_c (ver Figura 2.33), con el objetivo de retrasar estos *slots* fotónicos hasta que las contenciones sean resueltas.

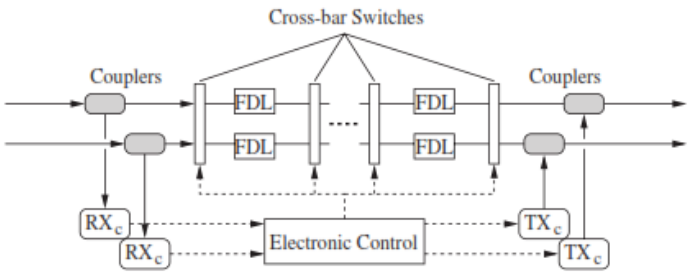


Figura 2.33 Arquitectura de un bridge SDL [75]

Los *buffers* ópticos se pueden categorizar en dos formas fundamentales en función de los siguientes criterios:

Número de etapas.- Se pueden distinguir líneas de retardo de una sola etapa, de múltiples etapas o esquemas híbridos. En el primer caso, se tiene únicamente un bloque FDL en paralelo, que permite un control más sencillo, pero tiempos de retardos fijos (ver Figura 2.34).

En el segundo caso, se incluyen varios bloques FDL multi-etapa en cascada, que pueden alcanzar retardos variables (ver Figura 2.35) y optimizar el hardware requerido para un *buffer* de mayor capacidad.

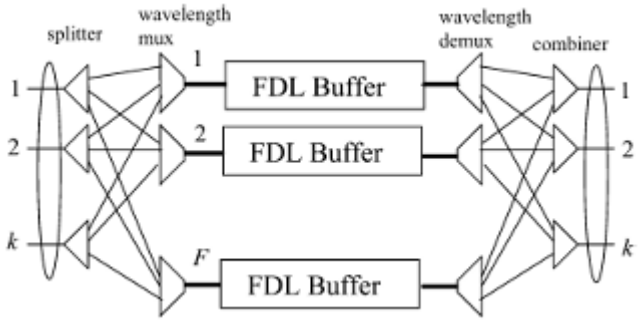


Figura 2.34 Estructura de buffers ópticos en un puerto de salida en el plano de datos de un conmutador óptico [73]

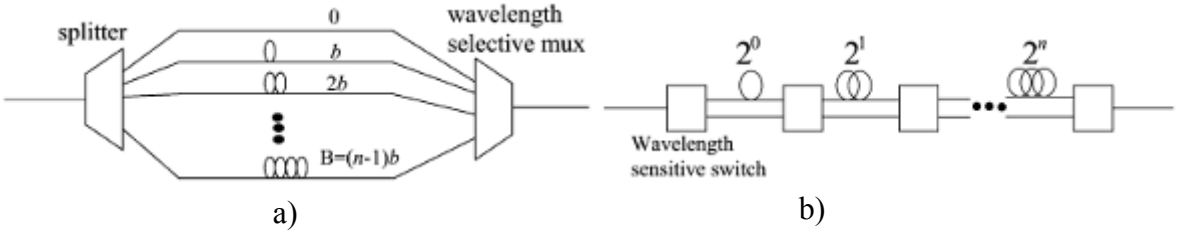


Figura 2.35 Buffer FDL a) con retardo fijo b) con retardo variable [73]

Tipo de arquitectura.- Se pueden encontrar *buffers* configurados con alimentación directa *"feed-forward"*, con retroalimentación *"feedback"* y esquemas híbridos. En una arquitectura *feed-forward* (ver Figura 2.36a) las líneas de retardo conectan la salida de una etapa de conmutación a la entrada de la siguiente etapa de conmutación. En una arquitectura *feedback* (ver Figura 2.36b), las líneas de retardo conectan la salida de la etapa de conmutación de regreso a la entrada de la misma etapa o a una etapa anterior.

En una arquitectura híbrida se combinan los *buffers feed-forward* y *feedback*. Se pueden implementar largos tiempos de espera y un cierto grado de retardos variables con los esquemas

que incluyen realimentación, variando el número de bucles que una ráfaga puede experimentar [24].

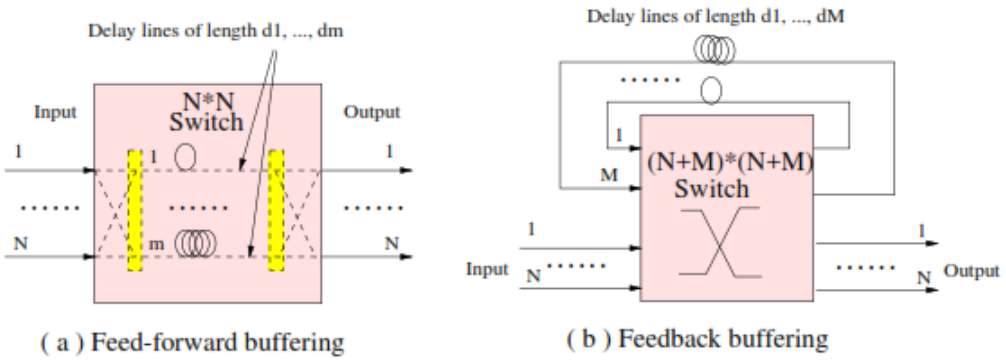


Figura 2.36 Esquemas de buffering a) feed-forward y b) feedback [69]

Al mismo tiempo, según la posición de los *buffers*, se pueden distinguir esencialmente tres tipos de configuraciones de los conmutadores ópticos: *buffering* en entrada, *buffering* en salida y *buffering* compartido. En el *buffering* en entrada (o salida), un conjunto de *buffers* está dedicado para cada puerto de entrada (o salida), mientras que en el *buffering* compartido, se puede distribuir un conjunto de *buffers* para todos los puertos del conmutador. El *buffering* de entrada tiene un desempeño deficiente debido al bloqueo en la cabecera de la línea, (HOL[26], *Head-Of-Line*), mientras que los otros dos tipos de *buffering* pueden alcanzar un buen desempeño en cualquier conmutador de paquetes. Sin embargo, el *buffering* en salida requiere un número significativo de FDLs así como también grandes tamaños de conmutadores. Por otro lado, el *buffering* compartido permite que todos los puertos de salida puedan acceder a los mismos *buffers*, reduciendo así el número total de éstos en un conmutador, mientras se alcanza un nivel deseado de pérdida de paquetes. En el dominio óptico, el *buffering* compartido se puede implementar mediante una etapa de recirculación con *feedback* o a través de múltiples etapas

[26] El bloqueo HOL se produce cuando un paquete en la cabecera de una cola bloquea el resto de los paquetes en dicha cola, incluso si solicitan acceso a puertos disponibles. Como consecuencia, la latencia promedio de los paquetes eventualmente aumenta y el *throughput* de la red disminuye.

con *feed-forward*. Adicionalmente, los *buffers* se pueden configurar como *buffer* degenerado (incremento lineal), o *buffer* no degenerado (incremento no lineal) [59].

La Figura 2.37a muestra la llegada de una solicitud de reserva en el instante t_a y de la ráfaga correspondiente, separada por un tiempo de *offset* δ. La nueva ráfaga es bloqueada debido a una reserva existente. Sin embargo, como un *buffer* está disponible durante el tiempo de transmisión de la nueva ráfaga, ésta se puede retrasar mediante el FDL.

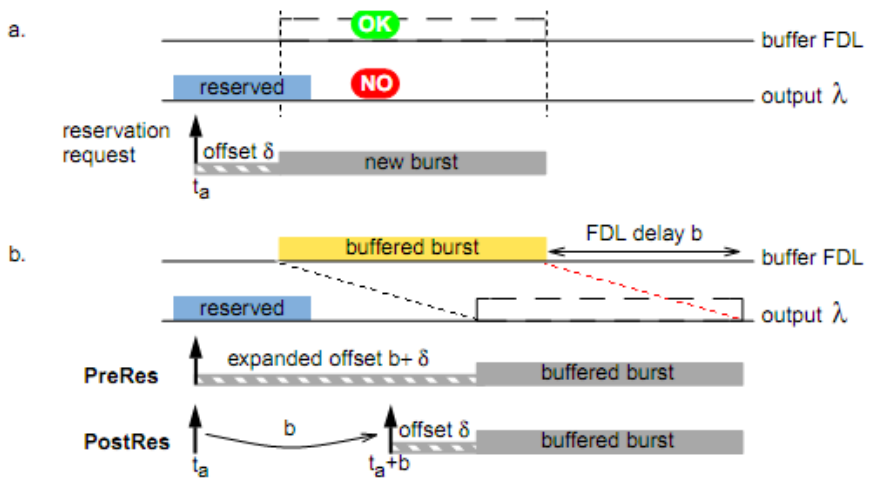


Figura 2.37 a) Evento de bloqueo de ráfagas y b) estrategias de buffer PreRes y PostRes[38]

Existen dos formas de realizar la reserva de canales de salida cuando se utilizan líneas de retardo [38]:

- **Reserva previa (PreRes).-** Es el método adoptado más comúnmente, donde la solicitud de reserva de un *buffer* FDL para una ráfaga entrante se realiza tan pronto como el paquete de control es procesado y se determina que no existe una longitud de onda disponible en el puerto de salida requerido. Por lo tanto, luego de realizar la búsqueda de un FDL libre, se comprueba si el canal de salida estará disponible en el instante de tiempo resultante de la suma del tiempo de *offset* original más el retardo FDL, en cuyo caso se realiza la reserva del FDL y del canal de salida antes de enviar la ráfaga al *buffer* correspondiente.

- **Reserva posterior** (**PostRes**).- A diferencia del caso anterior, en este método sólo se reserva el *buffer* FDL. Cuando el canal de salida no está disponible, se reserva el menor FDL libre, pero ninguna salida es reservada en ese instante. Sólo en $t_a + b$, es decir después de que la ráfaga ha ingresado en el FDL y δ antes de que la ráfaga abandone el *buffer*, se solicita una reserva del canal de salida. De esta manera, el tiempo de *offset* se mantiene en su valor original retrasando tanto el paquete de control como su ráfaga correspondiente por un período de tiempo, sin conocimiento previo de si un recurso estará disponible para la ráfaga cuando abandone el *buffer* FDL.

La principal diferencia entre ambos métodos, ilustrados en la Figura 2.37b, es que *PreRes* presenta una preferencia de las ráfagas que se envían al *buffer*, sobre otras ráfagas que pueden experimentar bloqueos. Por otro lado, en el método *PostRes*, el comportamiento es similar al caso sin FDLs, puesto que las ráfagas que se han enviado al FDL no interfieren con el resto de ráfagas, ya que sólo utilizan el canal de salida si está disponible, es decir, no aprovechan la ventaja de reservar los recursos con mayor tiempo de antelación. El mecanismo *PreRes* permite obtener probabilidades de bloqueo sensiblemente inferiores al mecanismo *PostRes* y es más adecuado en arquitecturas tipo *feed-forward*, ya que el mecanismo *PostRes* no permite aprovechar eficientemente múltiples líneas de retardo, mientras que si es más conveniente para aprovechar la flexibilidad que proporciona la arquitectura *feedback*.

Es necesario notar que, en cualquier arquitectura con *buffers* ópticos, el tamaño de los *buffers* es muy limitado, no sólo por problemas de calidad de la señal, sino también por restricciones en cuanto a espacio físico[27]; y aunque han existido muchos esfuerzos e investigación tendientes a lograr un *buffering* completamente óptico, se puede considerar que las líneas de retardo pueden ser aceptables en conmutadores prototipo, pero no son viables comercialmente [59].

Por otro lado, debido a que la solución deseable sería emplear el equivalente óptico de una memoria RAM, sin necesidad de conversiones opto-electrónicas, se hace necesaria la búsqueda de una verdadera memoria óptica de acceso aleatorio que se encuentra en sus primeros pasos. Por ejemplo, en [89] se ha desarrollado una celda RAM estática de 1 bit a

[27] Para alcanzar un retardo de 1 ms se requiere más de 200 kilómetros de fibra óptica.

partir de dos conmutadores SOAs y un *flip-flop* óptico SOAMZI (*SOA-Mach Zender Interferometer*). En Japón, desde el año 2006 al 2011 se ha desarrollado un programa para la búsqueda de una memoria RAM óptica [64]. En [65] en el marco de este programa se ha logrado desarrollar un sistema completo de memoria RAM, a partir de un arreglo de celdas de 1 bit construidas mediante una nano-cavidad, con una velocidad de acceso de 10 GHz (la velocidad de las memorias se mide actualmente en Hertzios, con lo que se obtendría un tiempo de acceso de 1 ps [139]). Para construir las nano-cavidades de las celdas de memoria, se han empleado distintos cristales fotónicos, empezándose por una prueba de concepto empleando Silicio (Si), mejorando drásticamente el consumo energético y el rendimiento utilizando *InGaAsP*[28]. Por lo tanto, el grado de avance experimentado en los últimos años permite pensar que pueda emplearse una RAM óptica en el futuro [41].

2.7.2 Conversión de longitud de onda

La conversión de longitud de onda es tal vez la técnica de resolución de contenciones más eficiente, puesto que tiene lugar únicamente en el dominio de longitud de onda y no retrasa las señales como en el caso de FDLs o el enrutamiento por deflexión [27]. Además, se puede pensar que esta técnica puede llegar a tener un importante potencial en cuanto a minimizar las contenciones, particularmente por el hecho de que se pueden transmitir cada vez más canales WDM sobre una sola fibra óptica.

Como ya se expuso anteriormente en este capítulo, esta técnica se basa en el proceso de convertir la portadora óptica de una señal de entrada en una longitud de onda diferente para su transmisión sobre un canal de salida, lo cual incrementa el reúso espacial de los canales ópticos incluso entre un 10% y 40% cuando la disponibilidad de longitudes de onda es pequeña [59]. En la conmutación óptica de ráfagas con conversión de longitud de onda, la contención se reduce mediante el uso de capacidad adicional en forma de canales ópticos disponibles por enlace, que pueden admitir la conmutación de una ráfaga en contienda en un

[28] *InGaAsP* es un sólido cristalino utilizado como semiconductor y en aplicaciones foto ópticas.

determinado momento. De hecho, la conversión de longitud de onda permite que un canal óptico de salida sea utilizado por ráfagas de entrada de distintas longitudes onda, incrementando así el grado de multiplexación estadística y el desempeño global de la red al reducir la probabilidad de pérdidas de ráfagas [27]. La conversión de longitud de onda se puede clasificar en función de su capacidad en las siguientes categorías:

Conversión fija.- Mapeo estático entre una longitud de onda de entrada λ_i y una longitud de onda de salida λ_j.

Conversión de rango limitado.- Una longitud de onda de entrada λ_i puede ser mapeada a un subconjunto de longitudes de onda de salida disponibles.

Conversión de rango completo.- Una longitud de onda de entrada λ_i puede ser mapeada a todas las longitudes de onda de salida disponibles.

Conversión dispersa.- La conversión de longitud de onda es soportada sólo por un grupo de nodos en la red.

La conversión fija puede ser considerada el tipo más simple de esta técnica, donde una longitud de onda λ_i que arriba a un nodo es convertida a otra longitud de onda de salida λ_j. Este tipo de conversores son estáticos y por tanto no proporcionan ninguna flexibilidad respecto a la longitud de onda de salida. La flexibilidad es mejorada permitiendo al conversor mapear una longitud de entrada dada λ_i a más de una longitud de onda de salida. Si las longitudes de onda de salida comprenden sólo un subconjunto de longitudes de onda que están soportadas sobre el enlace de fibra de salida, el tipo de conversión se denomina conversión de rango limitado. En contraste, la conversión de longitud de onda de rango completo no impone ninguna limitación en las longitudes de onda de salida, de manera que una determinada longitud de onda de entrada λ_i puede ser convertida a cualquier longitud de onda disponible sobre el enlace de fibra de salida, por lo que no existe restricción de continuidad de longitud de onda en las conexiones extremo a extremo.

Los conversores de longitud de onda son dispositivos bastante costosos, por tal razón, los costos de las redes completamente ópticas se pueden reducir desplegando conversores únicamente en unos cuantos nodos de la red (bien seleccionados) en lugar de equipar cada

nodo con estos dispositivos, lo cual conduce a la última clase de conversión de longitud de onda denominada dispersa.

La conversión de longitud de onda es una técnica poderosa para resolver las contenciones en redes OBS y se la puede utilizar sola o en combinación con otras métodos de resolución de contenciones como *buffering* óptico o segmentación de ráfagas. En la práctica, sin embargo, la implementación de sistemas con capacidad total de conversión de longitud de onda resulta no ser factible desde el punto de vista tecnológico-económico, y más bien, esquemas con conversión de rango limitado podrían ser más viables en escenarios reales, puesto que incluso han mostrado que apoyados en métodos heurísticos pueden determinar el número necesario de conversores y su óptima ubicación demostrando resultados aproximados al desempeño ofrecido por la capacidad de conversión completa. A pesar de esto, se estima que las contenciones podrían ocurrir más frecuentemente en redes OBS reales que se podrían contrarrestar mediante el uso de una o más de las técnicas de resolución de contenciones mencionadas anteriormente para complementar esta característica y alcanzar el mejor desempeño posible manteniendo una relación costo-beneficio acorde a las necesidades actuales y futuras.

2.7.3 Enrutamiento por deflexión

El enrutamiento por deflexión [114] es una técnica de resolución de contenciones, que se puede utilizar cuando el puerto de salida para una determinada ráfaga está ocupado, y en lugar de descartarla se desvía la ráfaga a través de un puerto de salida alternativo diferente al que estaba previsto originalmente.

En el enrutamiento por deflexión, una ráfaga desviada normalmente toma una ruta más larga hacia su destino, dando como resultado ciertos inconvenientes como un incremento en el retardo extremo a extremo, degradación de la calidad de la señal y entrega de paquetes fuera de orden que pueden causar inestabilidad en la red. Sin embargo, puede ofrecer ciertas ventajas como una mayor utilización del ancho de banda y un menor retardo extremo a

extremo, en comparación con el caso de retransmitir la ráfaga de datos desde el origen, tal y como se muestra a continuación.

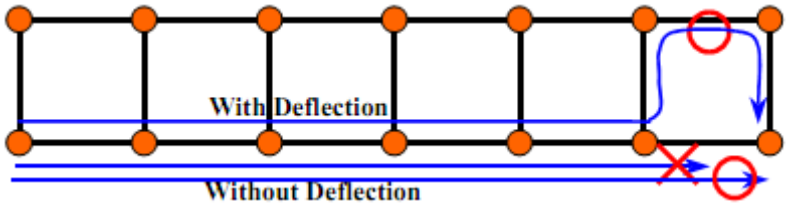


Figura 2.38 El efecto del enrutamiento por deflexión en la ganancia de ancho de banda [114]

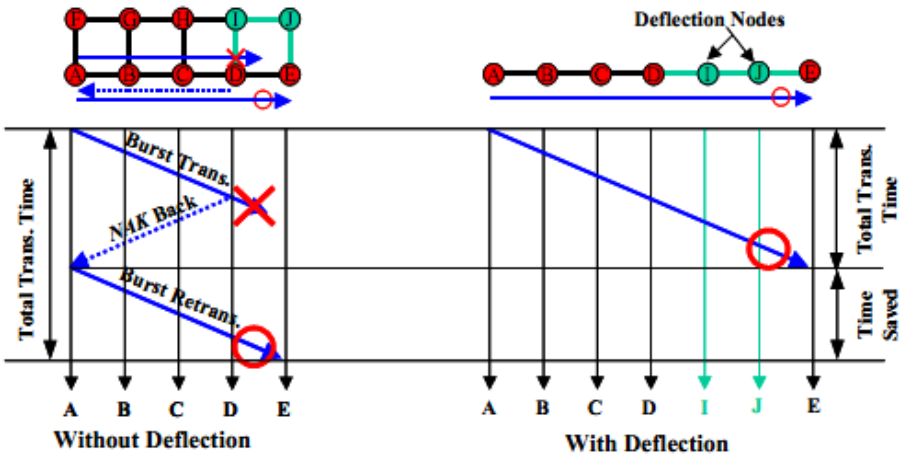


Figura 2.39 El efecto del enrutamiento por deflexión en la ganancia del retardo [114]

En la Figura 2.38, se puede observar cómo una ráfaga retransmitida malgasta el ancho de banda correspondiente a la ruta preestablecida hasta el punto donde ocurre la contención (5to. salto), y para alcanzar su destino requiere de un total de 11 (= 5 + 6) saltos, mientras que si la ráfaga fuera desviada en el 5to. salto, necesitaría únicamente 8 (= 5 + conteo saltos por deflexión) saltos. Además, en la Figura 2.39 se puede apreciar que la ráfaga retransmitida presenta un mayor retardo de transferencia desde el origen (nodo A) hasta su destino (nodo E), en contraste a que si la ráfaga fuera desviada en el punto de contención (nodo D) a través de los nodos I y J.

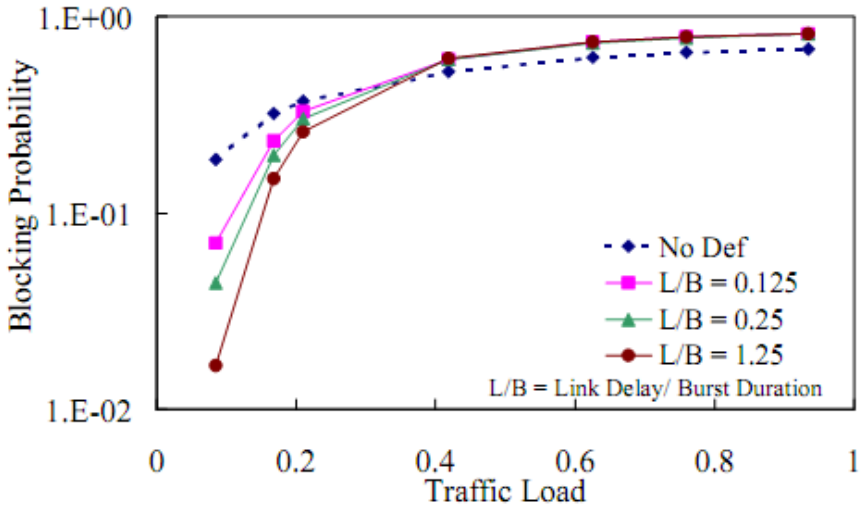


Figura 2.40 Probabilidad de bloqueo vs. carga de tráfico en el algoritmo de enrutamiento por deflexión normal [114]

El enrutamiento por deflexión también tiene sus desventajas. Por ejemplo, es eficiente sólo cuando la carga de tráfico es baja, mientras que cuando la carga es alta, el efecto de deflexión decrece y eventualmente podría conducir a probabilidades de bloqueo mayores en comparación a no utilizar deflexión, como se ilustra en la Figura 2.40.

Además en [114] se propone una mejora al algoritmo de deflexión normal, incluyendo dos funciones denominadas verificación del emisor y retransmisión desde el origen, con las que se logra disminuir en gran medida la probabilidad de bloqueo especialmente para bajas cargas de tráfico. La función de verificación del emisor incluye dos etapas: 1) si el enlace de salida predeterminado está ocupado, el nodo emisor no desvía la ráfaga a través de otro puerto sino que espera hasta que el canal predeterminado llega a estar disponible para transmitir la ráfaga como se ilustra en la Figura 2.41a y 2) si una ráfaga retorna al emisor, producto de la deflexión, éste la enviará por su enlace de salida predeterminado en lugar de desviarla por otro puerto, como se puede apreciar en la Figura 2.41b.

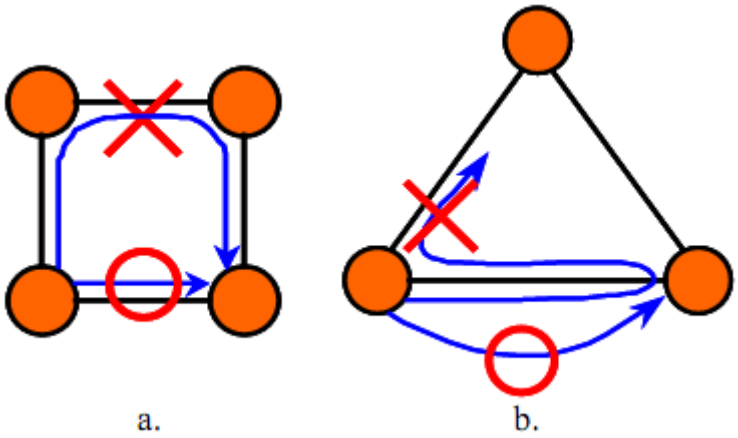


Figura 2.41 Función de verificación de emisor [114]

En la función de retransmisión del emisor, si el enlace de salida predeterminado está ocupado, el emisor intenta retransmitir la ráfaga después de un cierto período de tiempo definido en función de la longitud de la ráfaga y un valor aleatorio calculado en base a un contador de intentos y un límite máximo, evitando así una sobrecarga de tráfico en la red de enrutamiento por deflexión.

Además, es posible que la ráfaga desviada pueda entrar en un bucle indefinido dentro de la red incrementando la congestión, por lo cual, es necesario implementar mecanismos basados en contador del número máximo de saltos para prevenir longitudes de ruta excesivas, que se especifica normalmente en el campo TTL de los paquetes de control.

El enrutamiento por deflexión se puede considerar una alternativa apropiada para redes OBS con capacidades de almacenamiento limitadas o incluso sin ningún tipo de almacenamiento. En el segundo caso, el enrutamiento por deflexión se conoce como enrutamiento *hot-potato*, en el cual una ráfaga en contienda es reenviada inmediatamente (sin almacenarse) a través de otro puerto de salida disponible. En el método tradicional de enrutamiento *store-and-forward*, ampliamente utilizado en los enrutadores equipados con suficiente memoria electrónica, la ráfaga que contiende sería almacenada y luego reenviada a través del mismo puerto previsto tanto pronto como sea posible.

Con el fin de contrastar estas técnicas, se han realizado algunas evaluaciones utilizando topologías virtuales de red regulares normalmente utilizadas para análisis matemáticos, tales

como *Shu□eNet, Hypercube* y *Manhattan Street*, ilustradas a manera de referencia en la Figura 2.42, encontrando que el enrutamiento por deflexión trabaja de forma deficiente en comparación con el enrutamiento *store-and-forward*, a menos que la topología en uso esté muy bien conectada. Sin embargo, su rendimiento se puede mejorar significativamente con una pequeña cantidad de *buffers* [24].

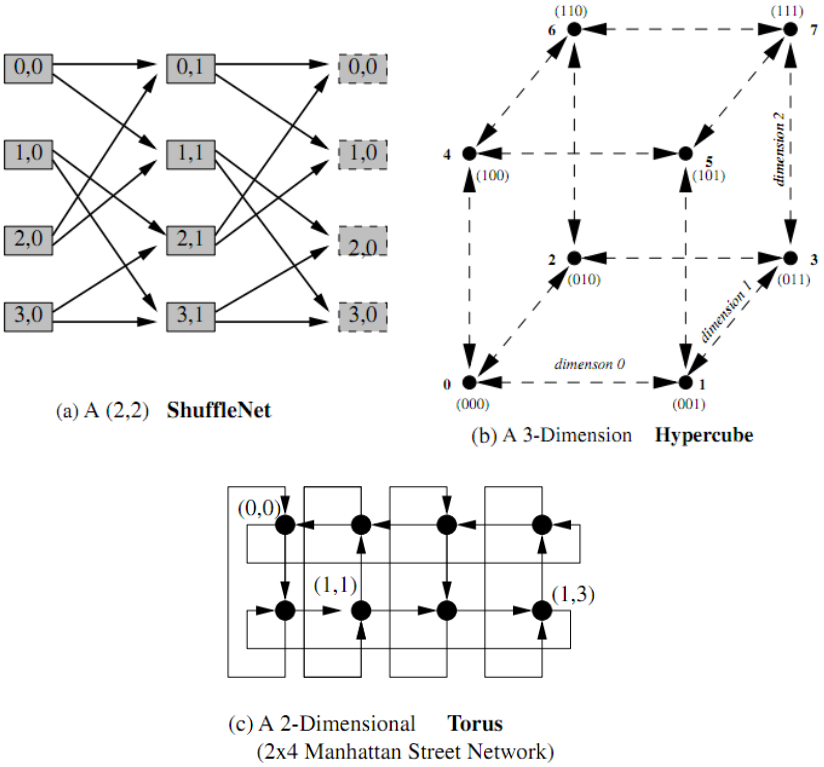


Figura 2.42 Topologías lógicas de red virtuales a) ShuffleNet, b) Hipercubo y c) Manhattan Street [8]

Para una topología arbitraria, la elección de que enlaces de salida y longitudes de onda utilizar para las ráfagas desviadas es crítica para el desempeño global de la red. Existen distintos protocolos de enrutamiento por deflexión, que de forma general se pueden dividir en tres categorías: enrutamiento alternativo fijo, enrutamiento dinámico con reconocimiento de tráfico y enrutamiento aleatorio [24].

Enrutamiento alternativo fijo.- Es el método más comúnmente utilizado para el propósito de deflexión a través de una ruta alterna fija definida sobre una base salto por salto, o mediante el almacenamiento completo de las rutas primaria y secundaria en cada nodo hacia cada posible

destino dentro de la red. En el primer caso, el principal inconveniente es la elección de una ruta alterna adecuada debido especialmente a la fuerte dependencia entre las probabilidades de pérdida de las ráfagas subsecuentes, las matrices de tráfico y la topología de la red. En el segundo caso, se puede obtener un buen desempeño en topologías pequeñas, porque es clara su limitación en términos de escalabilidad a medida que crece la red. Los algoritmos empleados para la implementación de este método varían considerablemente desde heurísticas simples como la siguiente ruta más corta de enlaces disjuntos, a los más complejos que requieren rutas alternas que retornen al nodo de "*next-hop*" original en menos de tres saltos [19].

Enrutamiento dinámico con reconocimiento de tráfico.- Este método toma en consideración la condición transitoria del tráfico en la selección de los enlaces de salida para las ráfagas desviadas [67]. Este tipo de enrutamiento presenta características similares con los mecanismos de balanceo de carga.

Enrutamiento Aleatorio.- En [19] se presenta un esquema de enrutamiento por deflexión denominado SP-PRDR (*Shortest Path Prioritized Random Deflection Routing*), que parece encontrar un buen equilibrio entre simplicidad de implementación, robustez y desempeño. En este método, las ráfagas transportan en sus cabeceras un campo de prioridad que se decrementa en uno cada vez que una ráfaga es desviada, permitiendo que ráfagas no desviadas (mayor prioridad) a través de sus rutas principales puedan preferirse a aquellas que han sido desviadas (menor prioridad). Por lo tanto, la probabilidad de pérdida de ráfagas en el peor caso de este método está limitada máximo por la probabilidad de pérdida de ráfagas de una red OBS estándar.

Un aspecto importante a considerar es que, normalmente el enrutamiento por deflexión es activado por un nodo OBS en función de la información local del estado de sus propios recursos (p. ej. longitudes de onda, congestión del enlace), lo cual puede conducir a una degradación en su desempeño debido a que no se tiene cuenta el estado global de la red. Por esta razón, con el fin de mejorar la resolución de contenciones por medio de esta técnica en [67] se propone un protocolo de enrutamiento por deflexión bajo demanda denominado CLDR (*Contention-Based Limited Deflection Routing*), en el cual todos los nodos intercambian periódicamente información de su estado local como: carga de tráfico, tasa de congestión de ráfagas, probabilidad de bloqueo, etc.

El algoritmo CLDR se ejecuta en todos los nodos OBS y secuencialmente realiza lo siguiente: 1) en base a cierto criterio de desempeño se determina dinámicamente si la ráfaga debería ser enrutada por deflexión o descartada para su posterior retransmisión desde el origen y 2) si la decisión es enrutar por deflexión, entonces la misma se realiza utilizando una ruta alterna que está basada en la minimización de una medición de desempeño que combina distancia y bloqueo debido a la contención.

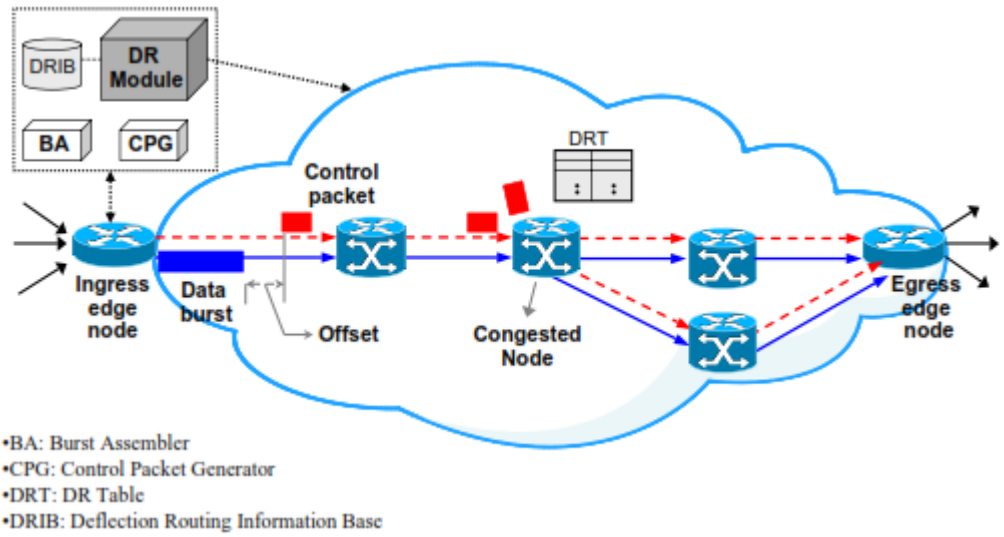


Figura 2.43 Arquitectura básica de una red OBS con enrutamiento por deflexión [67]

En la Figura 2.43 se presenta una red OBS en la que se aplica el algoritmo de enrutamiento CLDR, donde a medida que se procesan los paquetes de control, si un nodo determina que existe contención para una determinada ráfaga, genera un nuevo paquete de control a través de una ruta alterna, optimizada para el mejor desempeño (probabilidad de bloqueo y retardo), a fin de transportar la ráfaga de datos que debe ser desviada. Para esto, se utiliza un módulo de enrutamiento por deflexión (DR), encargado de mantener la base de información de enrutamiento por deflexión (DRIB) que almacena información de administración de la capa óptica (p. ej. carga de tráfico, tasa de contención de ráfagas, etc.) y de determinar las tablas de rutas por deflexión (DRT) calculadas de forma periódica o bajo demanda, mediante técnicas de programación lineal entera (ILP, *Integer Linear Programming*) que minimizan una función

objetivo definida como una suma ponderada del número de saltos desde el nodo congestionado hasta el destino y la probabilidad de pérdida de ráfagas de esa ruta. Los pesos correspondientes son normalmente establecidos por el administrador de la red y se pueden utilizar en función de los requerimientos de QoS en las decisiones de enrutamiento, para transportar las ráfagas desviadas de acuerdo con su clase de servicio.

La información necesaria para las funciones de operación, administración y mantenimiento (OAM) dentro de la red OBS, que incluyen la actualización de la DRIB, se envía en paquetes de control especiales (no asociado a cada ráfaga individual) a través del canal de supervisión óptico OSC. El número de saltos se transporta normalmente en el paquete de control, mientras que la probabilidad de bloqueo de todos los enlaces se distribuye por medio de los protocolos de GMPLS como OSFP o IS-IS con sus respectivas extensiones de Ingeniería de Tráfico (TE, *Traffic Engineering*) y el protocolo LMP (*Link Management Protocol*) de GMPLS.

La Figura 2.44 presenta la señalización utilizada por el algoritmo CLDR, para actualizar la información de la DRIB a lo largo de las rutas principales y alternas, que se transmite en forma de dos mensajes NACK_C y NACK_D, definidos para la ruta primaria y secundaria respectivamente.

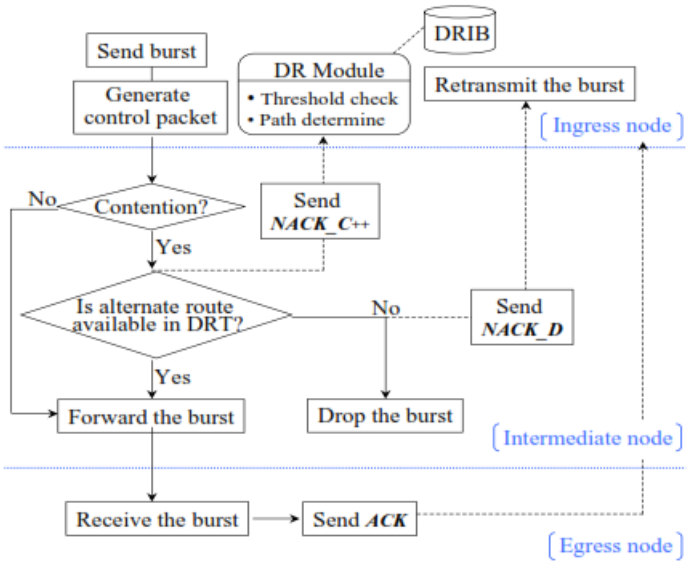


Figura 2.44 Diagrama de flujo que describe la notificación de contención de ráfagas y medición en el algoritmo CLDR [67]

Para escalar la funcionalidad del algoritmo CLDR se puede implementar el módulo de enrutamiento por deflexión (DR) de forma distribuida en múltiples enrutadores *hub*[29], que finalmente serán los encargados de calcular y actualizar de la información de las rutas alternas de un grupo de nodos vecinos, los cuales deben intercambiar información con su enrutador *hub* más cercano para obtener la información correspondiente a la DRT.

El algoritmo CLDR permite optimizar las decisiones de enrutamiento por deflexión disminuyendo los casos en los que no es acertado realizar esta acción. Para esto, el nodo que experimenta contención debe evaluar previamente su decisión de acuerdo a una función de verificación de umbral, aplicada sobre la ruta principal, que incluye dos variables ponderadas: 1) el número de saltos atravesados desde el origen hasta el nodo congestionado y 2) la probabilidad de pérdida de ráfagas extremo a extremo.

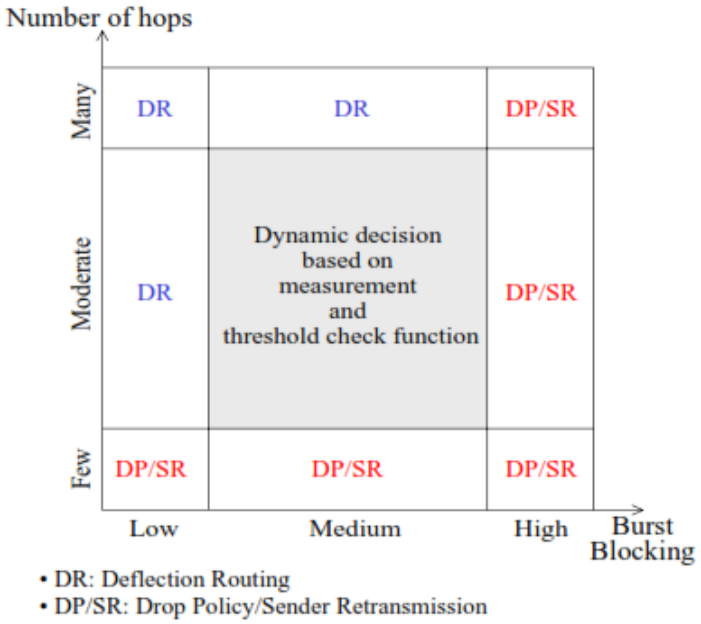


Figura 2.45 Acciones deseadas para diferentes rangos de bloqueo de ráfagas y conteo de saltos [67]

[29] Enrutadores *hub* son nodos que tienen muchos más grados nodales que otros, para servir como nodos de enrutamiento principales [122].

En la Figura 2.45 se resume la operación de la función de umbral, donde se puede apreciar que en condiciones de baja probabilidad de bloqueo (baja carga) se prefiere el enrutamiento por deflexión siempre y cuando el salto donde se detecta la contención esté más cercano al destino. Por el contrario, en condiciones de alta probabilidad de pérdidas (alta carga) se prefiere el descarte de la ráfaga seguido por la retransmisión desde el origen.

Por otra parte, para condiciones de carga moderada se puede elegir entre enrutamiento por deflexión o descarte y retransmisión desde el origen en función del resultado obtenidos de la función de umbral, que finalmente se diseña para minimizar el consumo de recursos y proveer un mayor rendimiento a la red.

Finalmente, es importante señalar que al aplicar CLDR o cualquier otro esquema de enrutamiento por deflexión en redes OBS, para que las ráfagas arriben exitosamente a su destino a través de las rutas alternas calculadas, se debe superar el problema del tiempo de *offset* insuficiente causado porque una ráfaga atraviesa más saltos que los previstos originalmente producto de ser desviada y debido a que el tiempo de *offset* entre la ráfaga de datos y su paquete de control decrementa después de cada salto, la ráfaga puede sobrepasar a su paquete de control. Por esta razón, se han propuesto algunas alternativas, tales como establecer un tiempo extra de *offset*, o un retardo a las ráfagas en algún nodo dentro de la ruta, encontrando que la opción más adecuada es retrasar una ráfaga en el siguiente salto después de la deflexión [24].

2.7.4 Segmentación de ráfagas

En los métodos de resolución de contenciones presentados anteriormente, cuando no se puede resolver la contención experimentada entre dos ráfagas, necesariamente una de ellas se debe descartar por completo a pesar de que su solapamiento sea mínimo. En ciertas aplicaciones con requerimientos estrictos de retardo pero tolerantes en cuanto a pérdidas de paquetes, puede ser preferible descartar unos cuantos paquetes en lugar de la ráfaga completa. Es así que en [59] se propuso una nueva técnica de resolución de contenciones denominada segmentación de ráfagas, que permite reducir la probabilidad de pérdida de paquetes dividiendo cada ráfaga en

segmentos, de manera que cuando se experimenta contención, únicamente los segmentos de una ráfaga que se sobreponen con los de otra ráfaga serán descartados.

Además, esta técnica permite que unas ráfagas tengan preferencia sobre otras, proporcionando una resolución de contenciones de una manera priorizada.

La segmentación de ráfagas fue considerada inicialmente dentro del contexto del paradigma OCBS (*Optical Composite Burst Switching*) [26], que controla las contenciones por medio de conversión de longitud de onda y la técnica de descarte de ráfagas (*burst-dropping*), en la cual, la parte inicial de una ráfaga se descarta hasta que una longitud de onda llega a estar disponible sobre el puerto de salida (y no por completo como en el caso de OBS). De esta manera, se puede enviar la parte restante de la ráfaga y por ende alcanzar una menor probabilidad de pérdida de paquetes, permitiendo además soportar mucho más tráfico en comparación con el esquema OBS tradicional.

En esta técnica, cada uno de los segmentos que componen una ráfaga, representa una unidad básica de transporte que consta de una cabecera y su carga útil (ver Figura 2.46), pudiendo contener uno o múltiples paquetes. La cabecera del segmento debe contener campos para bits de sincronización, información de detección o corrección de errores, información de origen y destino, y la longitud del segmento en el caso de segmentos de longitud variable.

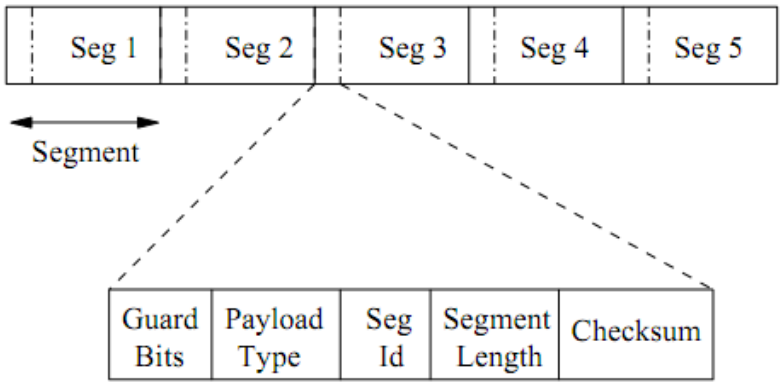


Figura 2.46 Campos de la cabecera de un segmento [59]

Con el fin de mantener la transparencia de los datos en el dominio óptico, y no sobrecargar los nodos centrales con funcionalidades para gestionar los segmentos de cada ráfaga, los límites reales del segmento son completamente transparentes en el núcleo de la red (no necesitan ser conocidos por la capa óptica), puesto que son entidades electrónicas no visibles en el dominio óptico. Consecuentemente, la responsabilidad de definir y procesar los segmentos se debe incluir en los nodos de frontera.

Es así que una posible implementación de segmentación es definir un segmento como una trama *Ethernet*, con el fin de utilizar el encabezado y cola propios de la trama *Ethernet* para la sincronización y detección de errores (o tramas incompletas) que se realiza específicamente a través de los campos preámbulo y CRC respectivamente. De esta manera, no se requeriría ninguna información de control adicional en cada segmento.

Al considerar el caso de una contención, un nodo OBS debe tomar la decisión de que segmentos debe descartar. Es así que se definen dos tipos de políticas denominadas *tail-dropping* y *head-dropping*, que se describen a continuación considerando que la ráfaga que arriba primero a un nodo OBS es referida como ráfaga original, mientras que la que llega después es referida como ráfaga que contiende, tal y como se puede apreciar en la Figura 2.47.

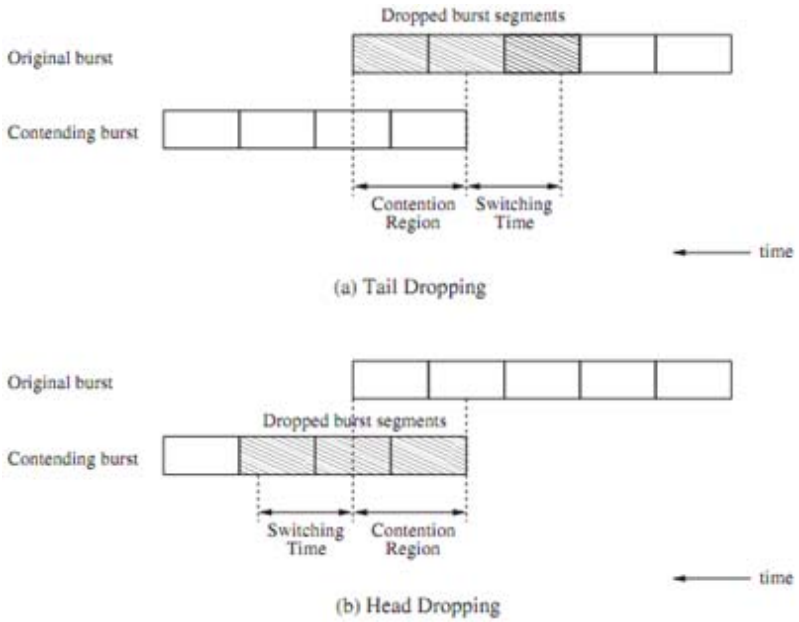


Figura 2.47 Políticas de descarte de la segmentación de ráfagas a) tail-drop y b) head-drop [59]

Tail-dropping.- En esta política, un nodo OBS descarta los últimos segmentos de la ráfaga original que se sobreponen con la ráfaga en contienda (región de contención) y adicionalmente los que se pierden durante el tiempo de conmutación del canal de salida de una ráfaga a la otra.

Head-dropping.- En esta política se descartan los primeros segmentos de la ráfaga en contienda que se encuentren entre la región de contención y el tiempo de conmutación, que es una medida directa del número de paquetes perdidos mientras se reconfigura el conmutador debido a la contención. *Tail-dropping* presenta una mejor oportunidad de entrega ordenada de paquetes, asumiendo que los que se perdieron son retransmitidos por el nodo de *edge* en un tiempo posterior; y aunque *head-dropping* es más propenso a una entrega desordenada de paquetes, su ventaja es que una vez que arriba una ráfaga a un nodo sin encontrar contención, se garantiza que ésta atravesará el nodo sin preferencia de ráfagas posteriores [59].

Con el fin de minimizar la pérdida de paquetes durante la contención y evitar que ráfagas pequeñas se prefieran sobre ráfagas de mayor tamaño, se considera una política de *tail-dropping* modificada, en la que se comparan en longitud la cola de la ráfaga original y toda la ráfaga en contienda, descartando la de menor longitud.

Cuando se descarta la cola de una ráfaga, se puede presentar el inconveniente de que su paquete de control transporte todavía la longitud de la ráfaga original, ocasionando pérdidas innecesarias derivadas de una contención virtual generada porque los nodos subsiguientes al punto de congestión, no tienen conocimiento de que la ráfaga fue truncada, a pesar de que una parte de ella haya sido descartada en un nodo previo. Por lo tanto, una alternativa sería que el nodo que realizó el truncamiento genere y envíe un paquete de control de terminación denominado *trailer* que notifique a los nodos posteriores de la finalización de la ráfaga truncada, como se puede apreciar en la Figura 2.48. Por el contrario, si se utiliza *head-dropping*, la cabecera de la ráfaga truncada puede ser actualizada inmediatamente en el nodo de contención.

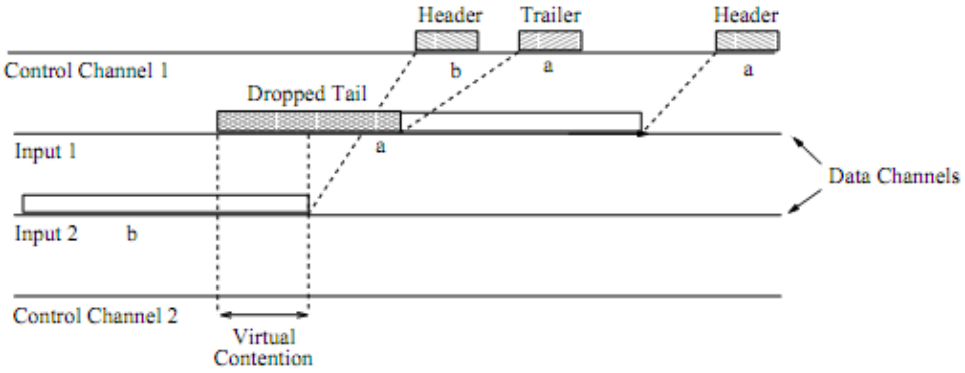


Figura 2.48 Paquete de trailer efectivo [59]

El paquete de *trailer* no elimina completamente la situación de una contención virtual, porque puede no ser completamente efectivo, como se ilustra en la Figura 2.49, sin embargo el *trailer* puede minimizar tales situaciones. Por lo tanto, es importante que el nodo en contención pueda generar y transmitir el *trailer* tan pronto como sea posible.

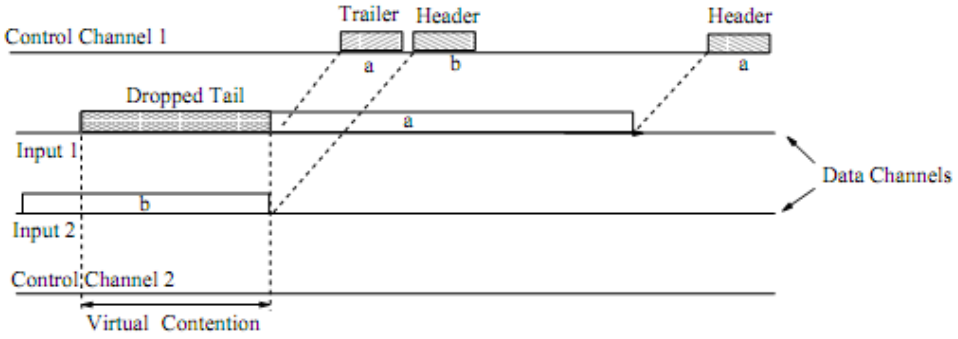


Figura 2.49 Paquete de trailer no efectivo [59]

La segmentación de ráfagas puede ser utilizada para resolver contenciones y reducir la pérdida de paquetes en redes OBS, mientras que la segmentación de ráfagas priorizada se puede utilizar para proveer diferenciación de servicios y soporte de QoS permitiendo que ráfagas de alta prioridad sean preferidas sobre las de baja prioridad, particular que fue experimentado en [59] considerando la política de *tail-dropping* y un esquema de ensamblado de ráfagas conocido como *Composite Burst Assembly*, encargado de colocar los segmentos de acuerdo a su clase de

servicio en orden descendente de preferencia desde el inicio hacia el final de la ráfaga, para luego asignar diferentes prioridades a las ráfagas (incluidas en sus paquetes de control), con el fin de permitir un tratamiento preferente en función de las clases de tráfico transportadas, de modo que cuando sea necesario, se descarten los segmentos dispuestos en la cola una ráfaga que son los menos prioritarios.

Existen también propuestas que utilizan segmentación en conjunto con otras técnicas de resolución de contenciones (p.ej. *buffering* óptico) presentadas en [59], donde se demuestra además una extensión básica de la técnica de segmentación implementada combinando segmentación con deflexión, de manera que en lugar de descartar uno de los segmentos sobrepuestos de una ráfaga en contención, es posible desviar los segmentos sobrepuestos de la ráfaga original o toda la ráfaga que contiende, hacia un puerto de salida alternativo diferente del que se tenía previsto.

La técnica de segmentación con deflexión puede incrementar la probabilidad de que las ráfagas alcancen su destino, mejorando por tanto, su desempeño en comparación con el hecho de utilizar sólo segmentación, para lo cual se define dos métodos denominados *segment-first* o *deflect-first*, que gobiernan la resolución de contenciones.

Segment-first.- En este método, si la longitud restante de la ráfaga original es menor que la ráfaga que contiende, entonces la ráfaga original es segmentada y su cola es desviada. En caso contrario, la ráfaga que contiende es desviada. Si el puerto alternativo está ocupado, la parte desviada de la ráfaga original o la ráfaga que contiende es descartada.

Deflect first:- En este esquema, la ráfaga que contiende es desviada si el puerto alternativo está libre. En caso contrario y si la longitud restante de la ráfaga original es menor que la ráfaga que contiende, entonces la ráfaga original es segmentada y su cola es descartada. Por el contrario, si se encontró que la ráfaga que contiende es menor, entonces es descartada.

En la Figura 2.50 se puede apreciar la operación del mecanismo de *segment-first*, donde la ráfaga original *a* es segmentada para ser transmitida seguida de la ráfaga en contienda *b* por el canal

de salida Output 1, mientras que la cola de la ráfaga original, *a'* es desviada utilizando el canal de salida Output 2.

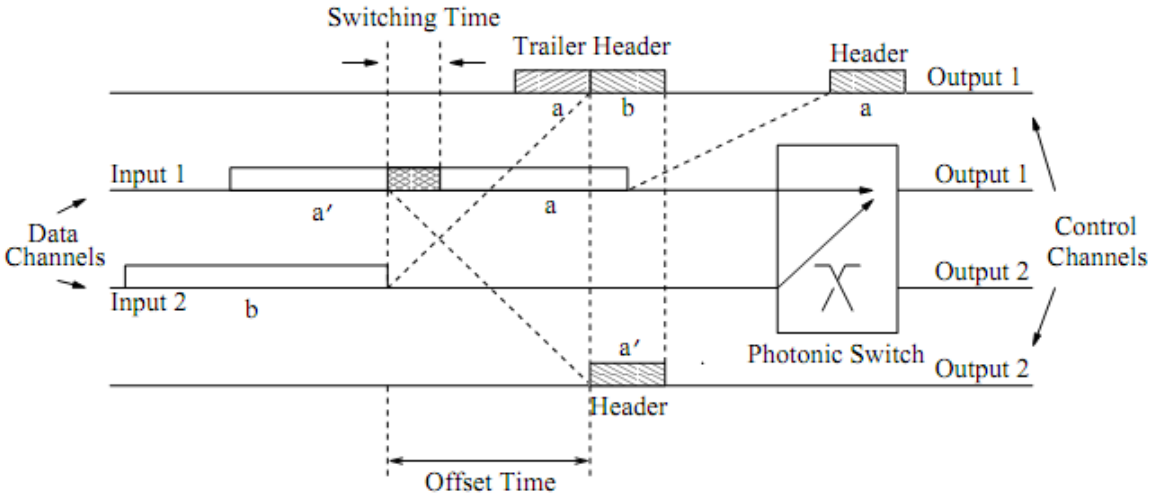


Figura 2.50 Segmentación con política de deflexión para dos ráfagas que contienden [59]

Un posible efecto secundario de la segmentación con deflexión es que, después de la contención, segmentos más cortos serán enviados como ráfagas nuevas que se propagan a través de la red, incurriendo en un excesivo overhead con respecto a los tiempos de conmutación y overhead de control por ráfaga. Más allá del mantenimiento estricto de los tiempos de *offset* para las ráfagas desviadas, otro asunto a considerar cuando se implementa segmentación con deflexión es cómo manejar ráfagas largas que pueden extenderse a lo largo de múltiples nodos simultáneamente [59].

2.8 Mecanismos para la recuperación de pérdidas de ráfagas [6][41,49] [103][132]

En la sección anterior se expuso los distintos mecanismos para solucionar el problema de contención de ráfagas, pero en el caso de que una contención de ráfagas no se haya podido resolver satisfactoriamente, inevitablemente se procederá a descartar una de las ráfagas. Ante

este evento, se tienen dos posibilidades, o bien son las capas superiores las encargadas de asegurar la fiabilidad de la transmisión (por ejemplo, el protocolo de transporte TCP se encarga de retransmitir los segmentos perdidos), actuando de manera independiente para cada flujo de tráfico, o bien es la propia red OBS la que proporciona mecanismos para asegurar que los datos no se pierdan.

Los principales mecanismos de recuperación en la red OBS de manera que no sea necesario recurrir a las capas superiores son: retransmisión, FEC y clonado de ráfagas [41]. El problema de recurrir a las capas superiores es que las pérdidas se podrían interpretar como estados de congestión, y tomar decisiones erróneas en base a ello. En otros casos, esto puede implicar elevados retardos en la aplicación hasta que se supera la falla, o bien, por ejemplo en el caso de servicios de video en tiempo real, se degradaría la señal.

2.8.1 Retransmisión de ráfagas

El objetivo fundamental de esta técnica es permitir que las ráfagas de datos que han sido descartadas debido a una contención, puedan ser retransmitidas en la capa OBS con el fin de reducir la probabilidad de pérdida de ráfagas. En este esquema, un paquete de control es enviado previamente para la reserva de recursos, mientras la ráfaga de datos es duplicada y almacenada en el nodo de ingreso antes de ser transmitida hacia el *core* de la red, para posibles retransmisiones.

Para identificar de forma individual las ráfagas que necesitan ser retransmitidas, se debe asignar a cada una un identificador único o número de secuencia. A medida que el paquete de control atraviesa la red, si la reserva del canal falla debido a una contención, el nodo intermedio enviará al nodo origen un mensaje de ARQ (*Automatic Retransmission Request*) notificando el fracaso de esta reserva. En ese momento, el nodo origen transmitirá el duplicado de la ráfaga de datos precedido de un nuevo paquete de control.

En caso de que la red esté ligeramente cargada, existe una alta probabilidad de que el duplicado llegue a su destino, mientras que si la red se encuentra altamente cargada, las

ráfagas retransmitidas tienen una menor probabilidad de ser recibidas exitosamente. El mecanismo de retransmisión puede permitir que una ráfaga sea retransmitida múltiples veces hasta que alcance su destino o hasta que el proceso de retransmisión exceda una restricción de retardo. El principal inconveniente de esta técnica es que introduce más carga sobre la red, conduciendo a una mayor probabilidad de contención de ráfagas; sin embargo, se logra reducir la probabilidad de pérdidas de ráfagas, especialmente a niveles de baja carga en la red.

En [132] se compara a través de simulaciones, el rendimiento de esta técnica en una red OBS sin este mecanismo y en una red OBS con deflexión. En el primer caso, los resultados muestran que el esquema de retransmisión mejora significativamente la probabilidad de pérdida de ráfagas en comparación con una red OBS sin este método. En el segundo caso, a una carga de tráfico moderada se obtiene una probabilidad de bloqueo cuatro veces menor comparada con el caso con deflexión, aunque el retardo extremo a extremo de las ráfagas en el esquema de retransmisión es mayor que en el esquema con deflexión, esto debido al retardo adicional requerido para notificar al origen de que la ráfaga ha sido descartada.

2.8.2 Corrección de pérdida de ráfagas con FEC (*Forward Error Correction*)

Una alternativa para recuperar las ráfagas pérdidas es el empleo de técnicas basadas en FEC (*Forward Error Correction*), que consisten en enviar información redundante junto con los datos, de tal manera que aunque falte una parte de la información, ésta se pueda recuperar en su totalidad en el destino. El codificado FEC se realiza en el origen y el decodificado en el destino.

En [6] se proponen dos técnicas que utilizan FEC para transmitir múltiples ráfagas de manera confiable y simultánea (p.ej. para aplicaciones de *grid computing*). En ambos métodos, los datos redundantes se generan a partir de múltiples ráfagas originales con codificación FEC. Luego, las ráfagas originales se transmiten a un nodo destino junto con la información redundante. En el método OBG (*Out of Burst Generation*) se genera una nueva ráfaga que consiste de datos

redundantes de las demás ráfagas. Si se llega a perder una ráfaga en un nodo intermedio y las demás ráfagas se transmiten finalmente hacia el destino, los datos faltantes se pueden recuperar en el nodo destino utilizando decodificación FEC. El principal inconveniente de este método es que genera un mayor número de ráfagas, y por tanto incrementa la probabilidad de bloqueo.

En el otro método, denominado IBG (*In-Burst Generation*), el número de ráfagas a ser transmitidas no cambia, sin embargo, su tamaño es mayor que el de las ráfagas originales debido a los datos redundantes, que se calculan a partir de la información de las otras ráfagas y se añaden al final de cada ráfaga. Se consideran dos mediciones de desempeño: la tasa de fallos[30] y el tiempo medio de procesamiento del FEC.

Un punto notable es que los esquemas propuestos pueden disminuir la tasa de fallos en un 90%, siendo IBG el método que presenta un menor valor en cuanto a este parámetro de desempeño, en comparación al método OBG, debido a que el número de ráfagas en el primer caso es menor que en el segundo, resultando en un uso más eficiente de longitudes de onda; por lo tanto, el esquema IBG es más eficaz que su contraparte OBG.

Aunque el tiempo de procesamiento del FEC de IBG es mayor que el de OBG, el método IBG requiere un menor número de procesadores FEC que el método OBG, debido a que su tasa de fallos es menor. Mediante simulaciones, los autores en [6] comprueban los beneficios de estos métodos propuestos y la capacidad de la red para recuperarse de pérdidas de ráfagas, consiguiendo mejores resultados en comparación con otras técnicas como el tiempo de *offset* extra, sin conducir a un incremento en el retardo extremo a extremo de las ráfagas transmitidas. Además, es posible reducir el retardo de transmisión extremo a extremo mejorando el procesador del FEC o incrementando el número de procesadores FEC.

[30] Múltiples ráfagas para β bloques de datos originales se consideran como una solicitud de transmisión. Por lo tanto, una solicitud de transmisión es exitosa si los β bloques de datos originales se transmiten eventualmente al destino con o sin recuperación FEC. En caso contrario, la solicitud de transmisión falla. Así se define la tasa de fallos como la relación entre el número de peticiones fallidas y el número de solicitudes de transmisión en general.

2.8.3 Clonado de ráfagas

En [49] se propone un esquema proactivo denominado clonado de ráfagas para reducir las pérdidas de datos debido a la contención en redes OBS. La idea consiste en replicar una ráfaga y enviar copias duplicadas de la ráfaga a través de la red de forma simultánea. Si la ráfaga original se pierde, la ráfaga clonada todavía puede ser capaz de llegar al destino. En esta técnica, se pueden crear una o más ráfagas clonadas por cada ráfaga original, y con el fin de que el tráfico clonado no interfiera con el tráfico original, se introduce un mecanismo de aislamiento de tráfico que emplea una técnica de planificación de ráfagas basado en prioridades, asignando alta prioridad a las ráfagas originales y baja prioridad a las ráfagas clonadas, de manera que estas últimas se descarten en caso de una contención con una ráfaga original. Los principales inconvenientes en el diseño de esta técnica en una red OBS son: el número de ráfagas clonadas por cada ráfaga original, la selección del nodo que realiza el clonado y el enrutamiento de las ráfagas originales y clonadas.

Este mecanismo se puede utilizar conjuntamente con la técnica de segmentación de ráfagas, específicamente con el método de *tail-dropping*, donde se propone invertir por completo la ráfaga clonada asegurando que los paquetes en la cola de la ráfaga original estarán en la cabecera de la ráfaga clonada, de manera que en el destino si ambas ráfagas son recibidas, incluso aunque la cola de cada ráfaga se pueda haber perdido debido a la segmentación, es posible recuperar la ráfaga completa. A través de extensivas simulaciones, los autores demostraron que el clonado de ráfagas puede mejorar considerablemente el desempeño en cuanto a pérdidas de datos sin un incremento significativo en el retardo de los paquetes. Además, los resultados mostraron que el clonado en el origen presenta el mejor desempeño entre todas las posibles configuraciones, y que la alternativa de inversión completa puede reducir aún más las pérdidas de datos en comparación con la solución sin inversión.

En [103] se evalúa el rendimiento de una red OBS con tráfico TCP empleando esta técnica, cuyos resultados mostraron un incremento del rendimiento de TCP con cargas de tráfico tanto altas como bajas.

2.9 Demostradores de OBS [2,7][41,44,48][58][63][71,72,79][87][97,98] [104,105][111][130,133,134,136,138][140-142,144]

OBS ha tenido una gran acogida en el ámbito de investigación de las tecnologías de comunicaciones ópticas, con más de 1000 artículos en congresos y revistas en los últimos 10 años [41]. No solamente se ha abordado OBS desde el punto de vista teórico, sino también se han desarrollado prototipos y demostradores (*testbeds*) experimentales en centros de investigación de universidades y empresas de varios países, confirmado la viabilidad tecnológica de este paradigma, de los cuales se menciona a continuación los más relevantes.

Aunque el concepto de OBS surgió originalmente alrededor de 1999, el primer demostrador no llegó sino hasta el año 2002, con el diseño e implementación de un prototipo de enrutador OBS que empleó una matriz de conmutación de alta velocidad basada en SOA integrada con interfaces ATM, utilizado para emular las funciones de ensamblado y desensamblado de ráfagas [111].

Posteriormente en el año 2003, se desarrolló el *testbed* de ATDnet (*Advanced Technology Demonstration Network*) [7] en el área de Washington DC, destinado a ser representativo para las posibles futuras redes metropolitanas. En este primer *testbed* se desarrollaron tres prototipos de nodo OBS interconectados mediante enlaces ópticos DWDM de 2.5 Gbps con distancias en el orden de varios kilómetros, donde se demostró experimentalmente el funcionamiento del protocolo de JIT, constituyendo una prueba de concepto de los protocolos de reserva en un sola vía sin confirmación, muy característicos de OBS. El protocolo JIT fue implementado en *hardware* en un prototipo de nodo OBS, mediante el uso de controladores de red OBS JIT, referidos como JITPACs (*Just-in-Time Protocol Acceleration Circuit*), para las funciones de señalización y control del protocolo, empleando señalización fuera de banda sobre una longitud de onda dedicada, llegándose a transmitir cerca de 80.000 mensajes de señalización por segundo (~ 12.5 μs por mensaje de señalización).

Adicionalmente, en este *testbed* se implementó el plano de datos, con un conmutador óptico basado en MEMS comerciales de *Firstwave*, con tiempos de conmutación de unos cuantos milisegundos. Sobre este prototipo se logró conmutar exitosamente una señal de televisión de

alta definición de 1.5 Gbps como información *óptica*. El protocolo JIT también fue utilizado ampliamente como parte del proyecto *JumpStart* que dio lugar a la implementación de varios *testbeds* de OBS a lo largo de la costa este de los EEUU. *JumpStart* soportaba el aprovisionamiento ultra-rápido de *lightpaths* mediante la implementación en hardware de la señalización JIT en los nodos intermedios, permitiendo a los usuarios finales establecer y liberar los *lightpaths* [111].

En el año 2004, en el *testbed* JGN II (*Japan Gigabit Network II*), en el que participaron el fabricante Fujitsu, la operadora NTT y las universidades de Osaka y Tokio, se realizó una prueba de concepto de OBS con un protocolo de señalización en dos vías basado en GMPLS para el control de la red [63][97]. En este demostrador se implementaron seis prototipos de nodo OBS con conmutadores MEMS en dos nodos y tecnología PLC en los demás prototipos, con tiempos de conmutación en el orden de milisegundos. Durante las pruebas se demostró la transmisión de ráfagas con tamaños de 100 ms, 200 ms y 300 ms, a una velocidad de 10 Gbps, logrando alcanzar un tiempo de conmutación menor a 30 ms. Posteriormente, en el año 2005, la empresa japonesa NTT llevó a cabo una demostración [98] con seis prototipos de nodo OBS implementados con tecnología PLC y señalización en dos vías, con un mecanismo de control de congestión basado en enrutamiento alternativo. Las pruebas se realizaron con cargas de tráfico entre 0.1 y 0.4, con probabilidades de congestión del 6% y 18% respectivamente, consiguiendo transmitir para cada caso el 77% y 45% de las ráfagas congestionadas a través de las rutas alternativas. En comparación al *testbed* anterior, en este demostrador se registró una mejora en cuanto al tiempo de conmutación, con un valor reportado menor a 20 ms, cubriendo además una distancia de 126 km, típica de la una red de alcance metropolitano. De esta manera, entre el *testbed* de ATDnet y el JGN II se evaluaron las dos enfoques principales de OBS, es decir, la señalización en una y dos vías.

Estos trabajos previos se enfocaron principalmente en algunas tecnologías clave, como los conmutadores ópticos y los protocolos de señalización, quedando aún un largo camino por recorrer para demostrar OBS con todas las funcionalidades que debe tener una tecnología de red en un entorno real. Es así que, el *testbed* de la Universidad de Tokio realizado en el año 2005, intentó dar un paso más allá que los demostradores anteriores, centrándose en una plataforma de red flexible de propósito general que consideró los demás aspectos de OBS,

como el proceso de ensamblado/desensamblado, protocolos de enrutamiento, algoritmos de planificación, esquemas de resolución de contiendas y la interacción con redes de tecnologías existentes [104]. En este sentido, uno de los principales aportes de este *testbed* fue el desarrollo de un prototipo de nodo de borde, con un ensamblador asincrónico de ráfagas de longitud variable, compuesto de múltiples colas y soporte para varias clases de servicio (CoS).

Además del nodo de borde, se mejoraron los nodos de transporte implementando en *hardware* el conocido protocolo de señalización JET y un mecanismo de planificación de ráfagas con técnicas de resolución de contenciones combinadas tales como el enrutamiento por deflexión y el algoritmo de asignación de longitudes de onda inteligente PWA (*Priority-based Wavelength Assignment*)[31]. Sin embargo, en los tres nodos de transporte OBS implementados, la tecnología de conmutación basada en PLC era la misma que en el *testbed* JGN II. En este demostrador se conectó uno de los nodos a Internet y se efectuaron pruebas de transmisión de video en tiempo real, basados tanto en TCP como en UDP para servicios de video bajo demanda y video en vivo respectivamente, entre los clientes conectados a los nodos OBS. Por tanto, este *testbed* cumplió el hito de ser el primer demostrador con completas funcionalidades de OBS e interconectado a Internet.

En el mismo período de tiempo que los *testbeds* japoneses mencionados anteriormente, el gobierno chino lanzó el programa TBOBS (*Testbed for Optical Burst Switching Network*), con el objetivo de experimentar distintos aspectos de OBS, en el que participaron BUPT (*Beijing University of Posts and Telecommunications*) y la Universidad de Shaghai Jiaotong [44][58]. Este demostrador consistió en una red en estrella de cuatro nodos, sobre la que se implementó el protocolo de señalización JET y el algoritmo de planificación de ráfagas LAUC-VF, del que hasta esa fecha sólo se disponían evaluaciones mediante simulación. Posteriormente, en el año 2007 en base a este *testbed* se evaluaron la transmisión de tráfico con soporte multi-QoS y el desempeño de TCP sobre OBS, cuyos resultados mostraron que el *throughput* de TCP es muy

[31] Con PWA, un nodo de borde incrementa la prioridad de una longitud de onda dada si las ráfagas han sido enviadas con éxito sobre esa portadora óptica; de lo contrario, se disminuye su prioridad. El nodo de borde siempre asigna una longitud de onda con el valor de prioridad más alto con el fin de reducir la BLP (*Burst Loss Probability*). Los resultados experimentales mostraron que tanto el enrutamiento por deflexión como el algoritmo PWA son eficaces en la reducción de la BLP, siendo el primero en mayor grado que el segundo [111].

sensible a la probabilidad de pérdida de ráfagas (se puede garantizar para un valor de pérdidas del 0.1%), y que el tiempo de ida y vuelta *"round-trip"* extra causado por la capa OBS tiene gran influencia sobre el ancho de banda disponible de TCP. Desde entonces, este demostrador chino se ha utilizado como base para estudiar de manera experimental el rendimiento de TCP sobre OBS, y se ha empleado además en varios estudios [133][134]. Otro de los cambios de este prototipo con respecto a los demás demostradores de OBS, fue la introducción de un módulo de conmutación rápida basado en SOA, desarrollado por la propia universidad. Además, este *testbed* fue utilizado posteriormente en el año 2009 para la primera demostración de interoperabilidad entre una red basada en GMPLS y una red OBS [48].

A partir de los resultados y experiencias del *testbed* JNG II mencionado anteriormente, en el año 2009 el consorcio japonés OITDA (*Optoelectronic Industry and Technology Development Association*), al que pertenecen la Universidad de Tokio y KDDI, en su proyecto NEDO continuó avanzando en la tecnología OBS [2][105]. Un primer paso fue crear un prototipo de nodo OBS empleando dispositivos de conmutación más modernos, en concreto los basados en la tecnología PLZT, que reduce en varios órdenes de magnitud los tiempos de conmutación ópticos. Un esquema de reconocimiento de etiqueta óptica implementado en una tarjeta FPGA, se integró en un prototipo de nodo OBS que podía escalar potencialmente hasta 64x64 puertos. En este *testbed* se demostró la transmisión de ráfagas de datos con tasas de bit mixtas, específicamente de 10 Gbps y 40 Gbps, empleando un esquema de resolución de contenciones basado en conversión de longitud de onda. De esta manera se logró un envío de ráfagas libre de errores a lo largo de más de diez saltos en una red OBS con arribo asincrónico de ráfagas de longitud variable. Debido a la mejora en cuanto a los tiempos de conmutación, la longitud de las ráfagas era mucho menor que en los *testbeds* anteriores, reportándose ráfagas con tamaño de 30 microsegundos, frente a los cientos de milisegundos de los prototipos anteriores. Al igual que en el *testbed* JGN II, las distancias entre nodos en este demostrador fueron representativas de un alcance habitual en una red metropolitana con recorridos de hasta 92 km.

Mientras que los demostradores anteriores no fueron diseñados para una aplicación específica, la Universidad de Essex, dentro del proyecto IST PHOSPHORUS [136], realizó en el año 2007 un *testbed* de OBS con una característica muy particular, el estar orientado o centrado a las aplicaciones y servicios de red (*application-aware*), en concreto con tecnologías de computación

distribuida [130]. Este demostrador utilizó tres nodos OBS operando a 2.5 Gbps tanto para el plano de datos como para el plano de control, en el que cada prototipo, además de sus funcionalidades típicas, como el protocolo de reserva JIT, incluyó capacidades para procesar los requerimientos de la capa de aplicación del usuario sobre la capa física de OBS, basadas en esquemas de administración y de reserva de recursos de red (p. ej. ancho de banda) y no de red (p. ej. recursos de computación), así como también en mecanismos de planificación y control de trabajos, utilizando procesadores de red de alta velocidad con FPGAs. En este demostrador se probó la transmisión de ráfagas ópticas de longitud variable, concretamente desde 60 μs hasta 400 μs con sus respectivos paquetes de control, que se transmitieron sobre tres longitudes de onda (λ5=1538.94 nm, λ6=1542.17 nm, λ7=1552.54 nm para las ráfagas de datos). En el nodo de *core*, se empleó una matriz OXS 4x4 compuesta de 16 celdas de conmutación óptica interconectadas, cada una conectada con dos guías de onda pasivas intersecadas perpendicularmente entre sí y dos AVC (*Active Vertical Couplers*) formados en cada celda mediante la colocación de una capa de guía de onda activa en la región de cruce de las guías de onda pasivas, permitiendo la función de conmutación, en base al efecto electro-óptico que induce cambios en el índice de refracción y ganancia-absorción en los AVCs, alcanzando un tiempo de conmutación de 10 μs.

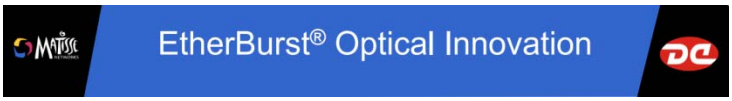
Tabla 2.5 Testbeds de OBS [41]

Testbed	Hitos	Tecnología de conmutación
ATDnet	Primer *testbed* de OBS, prototipo de JIT	MEMS
JGN II	Prototipo de reserva en 2 vías, 6 nodos	MEMS, PLC
BUPT	Pruebas de TCP sobre OBS, interacción con GMPLS	SOA
Universidad de Tokio	Primer *testbed* de OBS completo	PLC
OITDA	Reserva salto por salto, conmutador muy rápido	PLZT
Universidad de Essex	OBS centrado en la aplicación	AVC

Todos los intentos por comprobar las diversas arquitecturas y protocolos propuestos para redes OBS, tanto en simulaciones como en entornos controlados de laboratorio alrededor de todo el mundo, no se ha quedado únicamente en artículos o experimentos de prueba, sino que además han conducido a implementar algunos equipos comerciales, que se consideran un hito tecnológico que ha evidenciado los primeros pasos en cuanto a la viabilidad real de este paradigma.

La empresa pionera que apostó por OBS fue la *startup* norteamericana *Matisse Networks* (2003-2009), que desarrolló su solución denominada "EtherBurst" (ver Figura 2.51) enfocada al mercado de las redes ópticas metropolitanas con topologías en anillo (ver Figuras 2.52 y 2.53), llegando el año 2007 a realizar pruebas en laboratorio con proveedores de servicio [141]. El producto se basaba en ráfagas de tamaño constante, que se planificaban en el tiempo de una manera centralizada, utilizando *slots* de tiempo fijos.

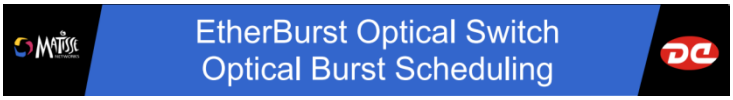
La otra base del producto eran los láseres sintonizables de 10 Gbps para transmitir ráfagas ópticas, capaces de sintonizarse en cualquier longitud de onda de la banda ITU-C[32] en el orden de nanosegundos. A continuación se presenta una breve descripción de los equipos desarrollados por esta compañía.



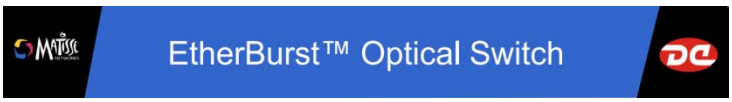
- **eBURST – 10** Gbps **Optical Burst Transponder**
  - Láser sintonizable que re-sintoniza cualquier longitud de onda ITU a ráfagas de paquetes hacia el destino
  - Procesador de paquetes que mapea inteligentemente tramas *Ethernet* al dominio de ráfagas ópticas
- **eWAVE – Optical Burst Amplifier**
  - EDFA con supresión de transiente de potencia

[32] Banda C para señales DWDM con longitudes de onda entre 1.530 nm and 1. 565 nm.

- Aproximadamente 18 dB de alcance



- **Packet processor controls when eBURST can Transmit**
  - El mapeo de ráfagas ópticas asegura que no existan colisiones
- **Packet processors communicate to form a common understanding of metro-wide bandwidth demands**
  - El mapeo de ráfagas ópticas se ajusta dinámicamente para manejar cambios en los patrones de tráfico



- **PX-1000 Photonic Node**
  - eWAVE EDFA con supresión de transiente
  - Administración de potencia óptica
  - Adición/extracción óptica para nodos SX
- **SX-1000 Ethernet Service Node**
  - Módulos Gigabit y 10 Gigabit Ethernet
  - Módulos eBURST
    - Optical Burst transponder
    - Procesador de paquetes
- **MatisseView**
  - Acceso unificado y seguro
  - GUI, SNMP, CLI

Figura 2.51 Conmutador óptico EtherBurst y sistema de gestión unificado [141]

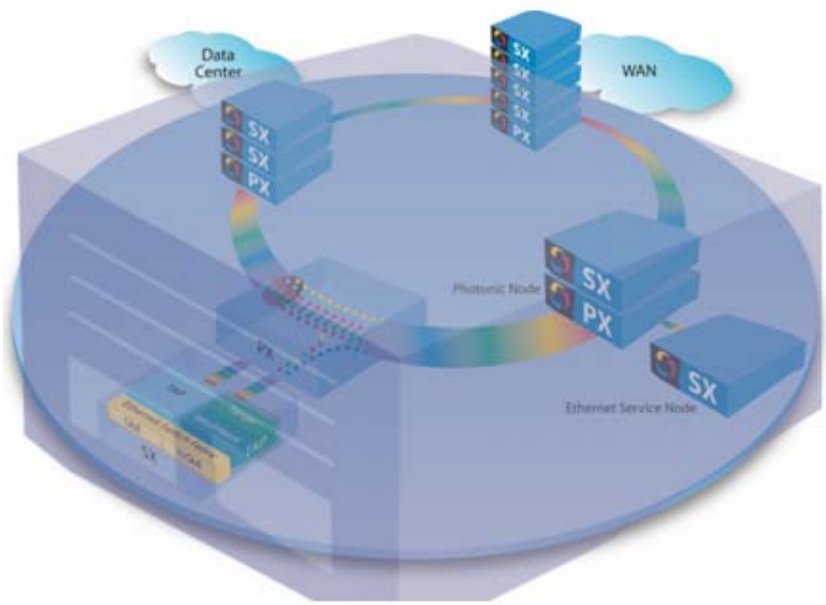


Figura 2.52 EtherBurst como un switch de capa 2 distribuido [141]

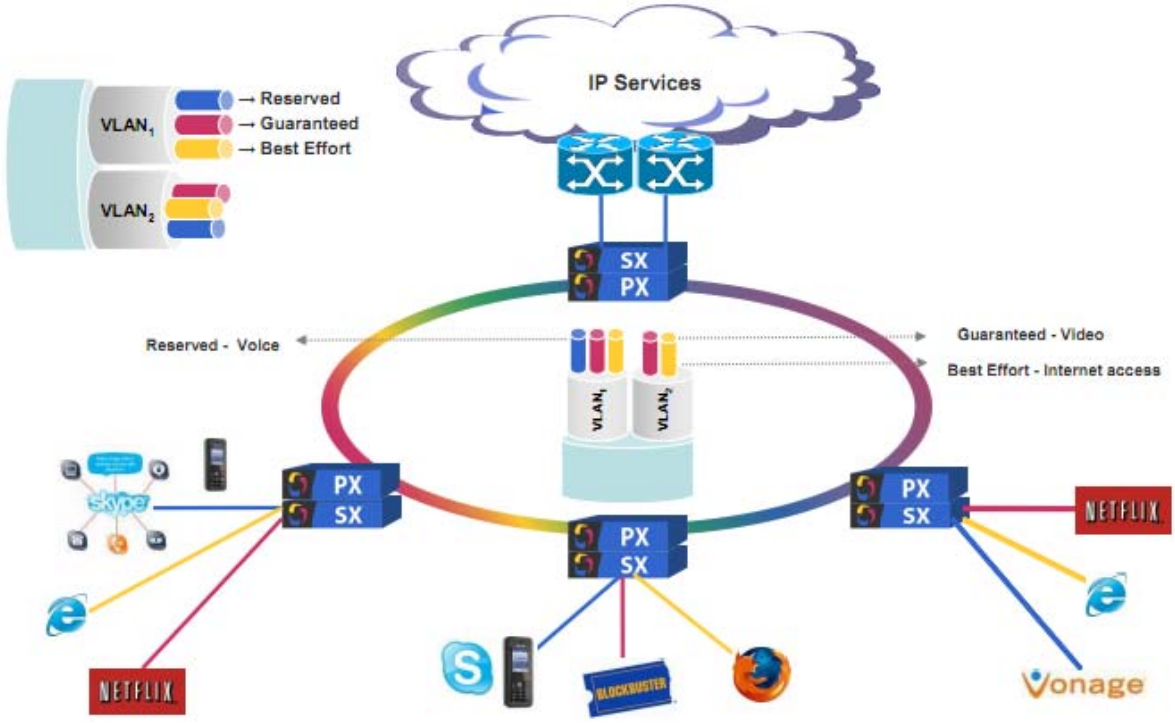


Figura 2.53 EtherBurst para servicios Triple-play [141]

Con este primer producto comercial, se podía transmitir en total desde 10 hasta 640 Gbps en un entorno metropolitano. A pesar de las cualidades de la tecnología y de informes

económicos favorables, como el llevado a cabo por *Network Strategy Partners* [141], lamentablemente Matisse Networks cerró sus operaciones en diciembre de 2009, al no poder conseguir más fondos para seguir adelante [142].

La siguiente empresa en apostar por OBS fue la compañía Irlandesa *Intune Networks* [138], que anunció desde 2009 en varias notas de prensa su disponibilidad y continuo desarrollo de la tecnología OBS para el entorno metropolitano [71][87]. Sin embargo, no fue sino hasta mayo de 2011 cuando oficialmente se lanzó el producto comercial denominado "*Optical Packet Switch & Transport*" (OPST), nombre comercial "*Verisma IVX8000*" [72][140]. Al igual que Matisse, la solución de Intune se centró en anillos metropolitanos y empleó láseres sintonizables rápidos que permiten conectividad en malla a nivel óptico, como se ilustra en la Figura 2.54.

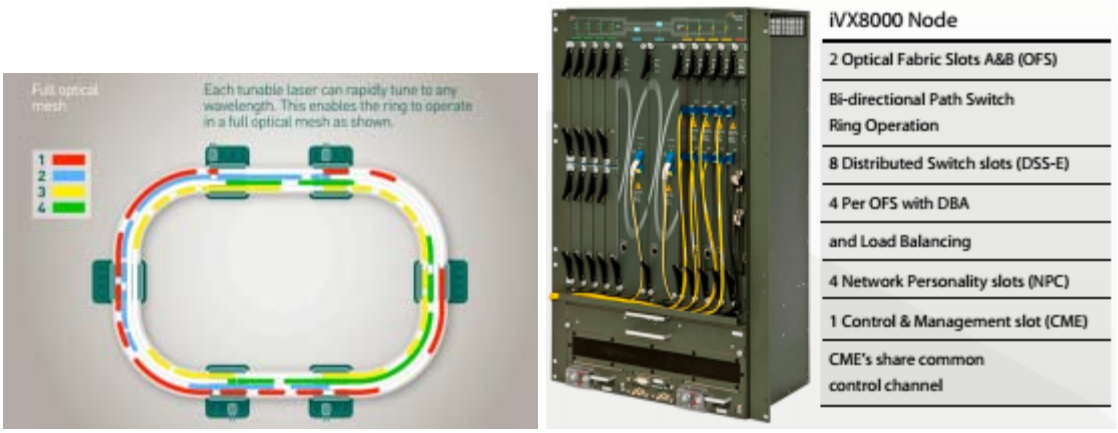


Figura 2.54 OPST con láseres sintonizables ultra rápidos que proveen conectividad en malla a nivel óptico [140]

La principal novedad de Intune es que se trata de una tecnología totalmente asíncrona (al contrario que Matisse, en la que los slots de tiempo eran fijos), que apunta a satisfacer los requerimientos de las redes de nueva generación, sustituyendo las capas de conmutación y transporte separadas, por una red de conmutación óptica de paquetes unificados, como se ilustra en la Figura 2.55.

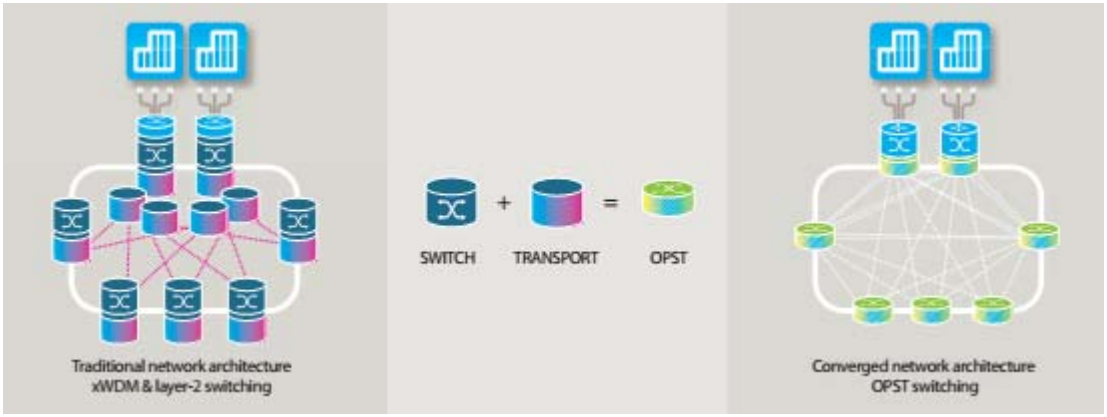


Figura 2.55 OPST integra las capas de conmutación y transporte para consolidar una arquitectura de red convergente [140]

Finalmente, la última empresa en anunciar un producto de OBS ya no se trata de una startup como las dos anteriores, sino de Huawei una empresa madura con productos comerciales en comunicaciones ópticas, que anunció en la OFC (*Optical Fiber Conference*) del año 2011 la demostración de una versión pre-comercial de su solución OBTN (*Optical Burst Transport Network*)[33] que utiliza la tecnología de ráfagas para llevar a cabo la conmutación de paquetes en la capa óptica y constituye un hito importante en el avance hacia una red todo óptica [144]. En la siguiente edición de la OFC en el 2012, Huawei continuó mostrando su interés por la tecnología OBS con un nuevo prototipo de nodo OBS, denominado PPXC (*Petabit per-second Photonic CrossConnect*) [79], que consta de una matriz de 80×80. De esta manera, se puede apreciar claramente que OBS va tomando cada vez más fuerza como alternativa para las redes de próxima generación y sin duda ha captado el interés de la industria. Sin embargo aún no existe ningún despliegue comercial en producción en una operadora [41].

[33] OBTN es una tecnología de conmutación completamente óptica dirigida por la empresa Huawei, que permite la convergencia de la capa 0 (capa óptica), capa 1 (capa física eléctrica) y capa 2 (capa de enlace), reduciendo la OAM total de una red.

CAPÍTULO III TCP SOBRE REDES OBS

TCP (*Transmission Control Protocol*) se ha convertido en el estándar de facto de los protocolos de transporte por su uso generalizado en la mayor parte de las aplicaciones en Internet de hoy en día, y se prevé que seguirá siendo, en el mediano y largo plazo, el protocolo de transporte predominante. Es así que surge la necesidad de estudiar TCP en redes OBS, puesto que su combinación puede ser una alternativa viable para la próxima generación de un Internet Óptico de alta velocidad.

Al respecto se han desarrollado varios estudios de TCP sobre OBS basados en modelos estocásticos, que proveen expresiones matemáticas para describir el comportamiento del *throughput* de TCP y que han sido validadas mediante simulaciones, cuyos resultados han demostrado que varían principalmente en función del ancho de banda de acceso, versión de TCP, tiempo de ensamblado de ráfagas, y probabilidad de pérdidas de ráfagas; dando lugar además a una propuesta de nuevas implementaciones de TCP que se adaptan de mejor manera al escenario sobre redes OBS.

En este capítulo se presenta una breve introducción al protocolo de transporte TCP y sus variantes más comunes basadas en pérdidas como Reno, New Reno y SACK. Además, se describen los factores que afectan el desempeño de TCP sobre redes OBS, y posteriormente se examinan los modelos matemáticos para el *throughput* de TCP Reno, basados en el modelo de Padhye [85], específicos para el caso de fuentes 'lentas' y 'rápidas; finalizando con una presentación analítica de los modelos matemáticos basados en la teoría de procesos regenerativos de Markov o teoría de renovación para las distintas variantes de TCP antes mencionadas, que modelan el *throughput* de TCP en función de la probabilidad de pérdidas de ráfagas y el retardo de ida y vuelta, RTT (*Round-Trip Time*), extendiendo los modelos iniciales para fuentes de velocidad genérica.

3.1 Introducción [35][42][57][101][111][120]

TCP es uno de los protocolos más importantes del stack TCP/IP, que se utiliza ampliamente en Internet para el transporte de información extremo a extremo, mediante el establecimiento de una comunicación orientada a la conexión que proporciona una entrega ordenada y confiable de datos, específicamente flujos de bytes de una aplicación en un host a una aplicación en otro host, que operan normalmente bajo una arquitectura cliente/servidor. Es así que muchos estudios realizados, han mostrado que aproximadamente el 90% de las aplicaciones en Internet se soportan sobre este protocolo, e incluyen un sin número de servicios tales como: la navegación web, el correo electrónico, la transferencia de datos como el caso de *mega* o *rapidshare* que ofrecen funciones de carga y descarga de archivos, transmisiones P2P que emplean programas como *emule* o *bittorrent*, e incluso el *streaming* de audio y video bajo demanda como es el caso de los populares servicios de *youtube* y *netflix*.

Por otro lado, con el crecimiento cada vez mayor en cuanto al número de usuarios en Internet y aplicaciones que demandan gran ancho de banda, la tecnología utilizada en las redes de *core* ha ido evolucionado hacia redes ópticas de alta velocidad donde el diseño de un protocolo de transporte eficiente y confiable se ha convertido en un tema desafiante debido a los problemas asociados con el diseño fundamental de TCP [111]. Las redes de transporte óptico de alta velocidad como es el caso de OBS, sufren de un problema conocido como el elevado producto ancho de banda por retardo, DBP (*Bandwidth Delay Product*), que corresponde a la capacidad del enlace, es decir al número de bits que como máximo pueden estar en tránsito.

Dado que TCP es un protocolo de ventana deslizante, la cantidad de datos que pueden ser enviados a la vez sin acuse de recibo, puede ser como máximo el tamaño de la ventana de transmisión, y si su valor es menor que el producto ancho de banda por retardo, existe un desperdicio de la capacidad del enlace, en cuyo caso, el emisor TCP estaría inactivo la mayor parte del tiempo. El límite de la ventana TCP estándar es 2^{16} - 1 bytes (64 Kbytes), que es muy baja para redes ópticas de alta velocidad, y aunque las extensiones definidas en la RFC 1323 [57] donde una opción de escalamiento de ventana permite ampliarla hasta 2^{32} - 1

bytes (4 Gbytes), el incremento lento de la ventana de congestión y el decremento abrupto cuando se pierden segmentos dificulta alcanzar tal ventana de transmisión [42].

Otro problema que debe enfrentar TCP en una red con un elevado producto ancho de banda por retardo, es el tiempo que tarda en crecer la ventana hasta llegar al tamaño ideal en ese medio, debido a su mecanismo de congestion avoidance AIMD (*Additive-Increase/Multiplicative-Decrease*). Por ejemplo, si la capacidad del enlace es de 10 Gbps y el RTT en la ruta es de alrededor de 50 ms, una conexión TCP con un tamaño promedio de paquete de 1.5 KB tardaría aproximadamente media hora en alcanzar el tamaño máximo de la ventana, incluso con una probabilidad de pérdida de paquetes de 10^{-8} [111]. Para reducir este tiempo, las versiones de TCP incorporan incrementos cada vez más agresivos en el período de congestion avoidance, como el caso de BIC TCP [120], High SpeedTCP [35], etc. Inclusive se han propuesto ciertas alternativas y técnicas denominadas variantes de TCP sobre redes OBS para mejorar su desempeño, que se pueden clasificar en las siguientes tres categorías: 1) soluciones en la capa de enlace, 2) detección de congestión con notificaciones explícitas, y 3) detección de congestión sin notificaciones explícitas [101, 111], descritas brevemente a continuación de manera general, sin explorar las variantes de cada clase, debido a que su diversidad y análisis implicaría extensiones adicionales fuera del alcance de esta tesis, que se podrían abordar en futuras líneas de estudio como resultado de esta investigación.

En la primera categoría, el dominio OBS se trata simplemente como la capa de enlace debajo de los enrutadores IP de *edge* (de manera que la transmisión de ráfagas es completamente transparente para la capa de transporte), donde se puede mejorar el *throughput* mediante la implementación de funciones de soporte adicional en los nodos de *edge* y *core*, que incluyen entre otras algoritmos de ensamblado adaptativo, FEC (*Forward Error Correction*), ARQ (*Automatic Retransmission Request*) y técnicas de resolución de contenciones como la retransmisión de ráfagas, el enrutamiento por deflexión, y la adopción de FDLs (*Fiber Delay Lines*) que son las más comunes.

La segunda categoría corresponde básicamente a soluciones *cross-layer* que utilizan el intercambio de señalización extra con los nodos OBS, de manera que se puede notificar a los agentes TCP sobre el estado de la red OBS, específicamente las condiciones del canal y la

causa real de la pérdida de paquetes como: congestión de la red, corrupción de paquetes, o cualquier posible fallo en los componentes de hardware.

Los emisores TCP retransmitirán los segmentos perdidos sin afectar su ventana de congestión para las pérdidas de paquetes debido a cualquier otra razón distinta a la congestión de la red; se incluyen en esta categoría implementaciones TCP con ECN (*Explicit Congestion Notifications*) o ELN (*Loss Notifications*) tales como BTCP (*Burst TCP Back/BNack*), TCP-BCL (*TCP with Explicit Burst Contention Loss*), y TCP-ENG (*TCP Explicit Notification GAIMD*). La tercera categoría, incluye protocolos de transporte completamente nuevos que se adaptan al comportamiento del transporte de ráfagas subyacente sin el intercambio de información explícita, por lo tanto los emisores TCP tratan de evaluar los estados en el dominio OBS a través de ciertas ecuaciones, medidas de RTT, y/o herramientas de estadística, con lo cual se puede mantener y analizar un número de RTT anteriores en los emisores TCP a fin de identificar si un evento de pérdida de paquetes es debido a congestión persistente o a contención aleatoria de ráfagas; se incluyen en esta categoría esquemas de control de congestión del lado del emisor como SAIMD (*Statistical AIMD*), Threshold-Based Vegas, Burst TCP with BLE (*Burst Length Estimation*), y *BAIMD* (*Burst AIMD*).

En una red OBS donde los paquetes son ensamblados en ráfagas en el nodo de ingreso, los segmentos TCP correspondientes a una sola sesión pueden ser agregados en una o varias ráfagas dependiendo de la tasa de llegada del tráfico y del tiempo de formación de las mismas. En este sentido, la pérdida de una sola ráfaga podría conducir a una pérdida de varios segmentos TCP al mismo tiempo, lo cual depende de muchos factores que no es simple de evaluar. De acuerdo al número de segmentos TCP perdidos, el emisor TCP puede percibir la pérdida de una sola ráfaga como una severa congestión y reducir drásticamente la ventana de congestión.

Además, el uso de técnicas de resolución de contenciones para OBS tales como el enrutamiento por deflexión o la retransmisión de ráfagas, podría dar lugar a la entrega de paquetes fuera de orden que para TCP es equivalente a pérdida de paquetes. Por lo tanto, se podría decir que el problema fundamental de TCP en la entrega de datos sobre una red OBS se debe al mecanismo AIMD usado por el emisor TCP, a la agregación de ráfagas y a las

pérdidas por contención que ocurren aleatoriamente[34] en los nodos intermedios, cuando varias ráfagas compiten por el mismo recurso de salida al mismo tiempo.

En base a lo antes mencionado, es necesario indicar que cualquier estudio adicional o diseño de nuevas variantes de TCP para redes OBS, requiere una comprensión más profunda del comportamiento de este protocolo sobre este tipo de redes. El impacto de la agregación de paquetes y las pérdidas de ráfagas (debido a la contención) sobre el *throughput* de TCP, debería ser analizado para diseñar nuevos métodos para mejorar su desempeño. Por lo tanto, evaluar analíticamente el desempeño de TCP sobre redes OBS, ayuda a entender el efecto del incremento en el retardo extremo a extremo y de las pérdidas de ráfagas aleatorias sobre el *throughput*. Es así que, en este capítulo se presentan varios modelos matemáticos que analizan el comportamiento de las versiones más comunes de TCP sobre redes OBS que permiten no sólo evaluar la dependencia del *throughput* frente a varios parámetros de la red OBS, sino también determinar el enfoque para diseñar mejores modelos para el protocolo TCP sobre redes de alta velocidad, en las cuales las pérdidas son correlacionadas [111].

3.2 Una breve introducción a TCP [3][11,13,14][23,28][30,33,34][40-42,46] [54,56][76,78][90][107][111,117]

Como se expuso en la sección anterior, TCP es un protocolo confiable orientado a la conexión que permite la transferencia de flujos de datos entre diferentes aplicaciones, para lo cual, el emisor recolecta una cierta cantidad de bytes (datos) de una aplicación y adiciona una cabecera TCP para formar una unidad de información denominada segmento, que lo encapsula en una datagrama IP y lo envía a través de la red hacia el receptor. La transmisión de diferentes aplicaciones de un host a otro se realiza multiplexando los distintos flujos con un

[34] En la conmutación sin almacenamiento dentro de una red OBS, las pérdidas de datos (ráfagas) ocurren aleatoriamente en su mayor parte debido a la variabilidad "*burtiness*" de corto alcance del tráfico ensamblado en ráfagas, a diferencia de las pérdidas de paquetes correlacionadas debido al *burtiness* de largo alcance en el tráfico de paquetes y desbordamiento de *buffer* en las redes electrónicas de conmutación de paquetes [107].

número de puerto diferente. En el receptor se procesa la información recibida, que incluye el desensamblado de las unidades de información, reordenamiento de los segmentos y entrega de los datos a cada aplicación demultiplexando los flujos de acuerdo al número de puerto correspondiente. Durante el establecimiento de una sesión TCP, se negocia un tamaño máximo de segmento MSS (*Maximum Segment Size*) que representa la cantidad máxima de datos, en bytes, que el emisor está dispuesto a enviar en un solo segmento. A lo largo de este capítulo, los términos paquete y segmento son utilizados indistintamente para referirse a un datagrama IP con un segmento TCP.

En la Figura 3.1 se presenta una visión general de la senda de estándares RFC de TCP que incluyen las características más importantes en el desarrollo de este protocolo, desde sus inicios hasta las definiciones de las diferentes implementaciones objeto de este proyecto de tesis, de las cuales se expondrán en esta sección las de mayor interés.

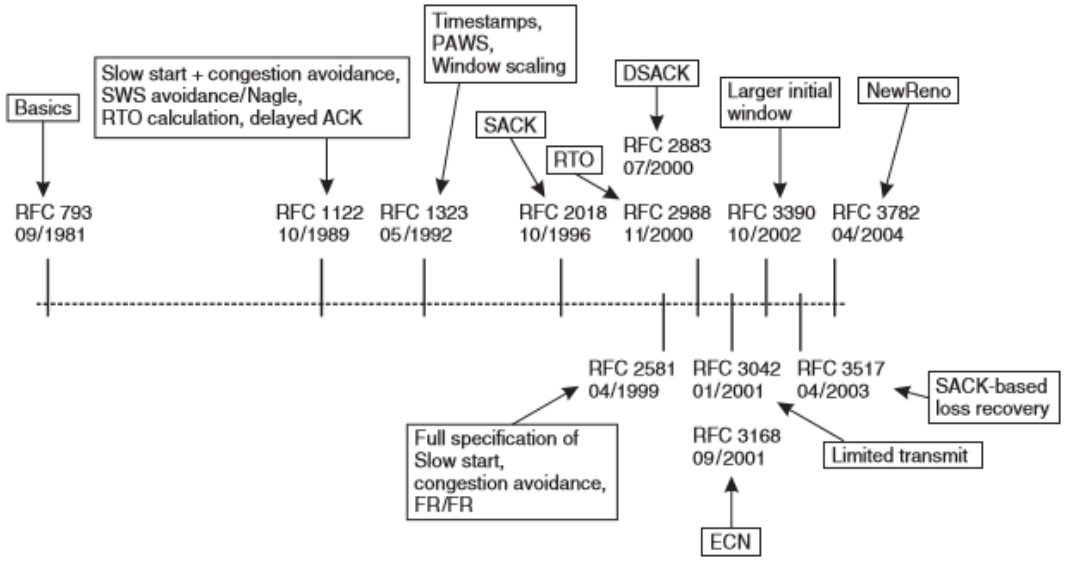


Figura 3.1 Especificaciones de la senda de estándares TCP que influencian cuando se envía un paquete [117]

La función principal del protocolo TCP es el control de flujo y control de congestión, realizado por el emisor ajustando el número de segmentos enviados en cada ronda para prevenir el desbordamiento de *buffer* del receptor y una sobrecarga en la red respectivamente.

Para lo cual, TCP utiliza un mecanismo de ventana deslizante que varía en función de la ventana de congestión *cwnd* que se define en el emisor (no anunciada en la cabecera TCP) y la ventana del receptor *rwnd* anunciada al emisor en la cabecera TCP. El valor de *cwnd* limita la cantidad de segmentos que se pueden transmitir en un determinado momento antes de recibir una confirmación de su recepción, en función del estado de la red; mientras que el valor de *rwnd* está relacionado con la capacidad disponible en el *buffer* de recepción. Por lo tanto, la ventana de transmisión efectiva del emisor para el envío de segmentos, es el mínimo entre la ventana de congestión y la ventana de recepción. La Figura 3.2 ilustra la evolución de las ventanas de congestión y transmisión de TCP.

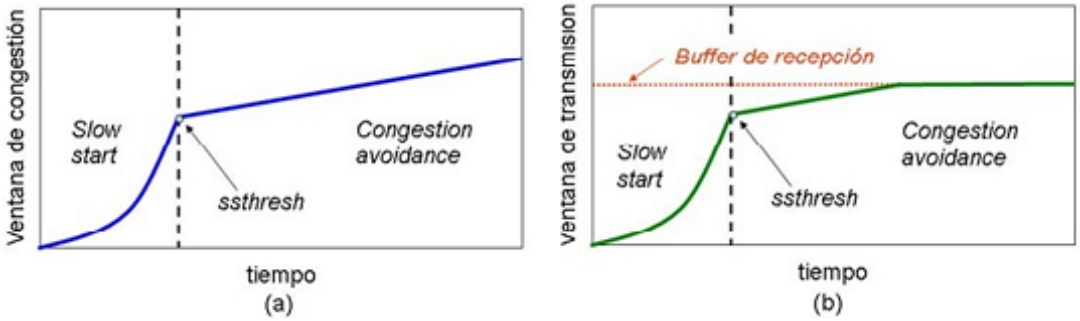


Figura 3.2 (a) Evolución de la ventana de congestión de TCP. (b) Evolución de la ventana de transmisión de TCP [41]

En la Figura 3.3 se han enumerado los bytes desde el 2 hasta el 11 con el fin de ilustrar la estructura de la ventana deslizante de TCP. La ventana anunciada por el receptor se denomina "ventana ofrecida" que cubre los bytes desde el 4 hasta el 9, lo que significa que el receptor ha reconocido todos los bytes hasta e incluido el número 3, y ha anunciado un tamaño de ventana de 6.

El emisor calcula su ventana utilizable (cuantos datos puede enviar inmediatamente), como la ventana ofrecida menos la cantidad de datos ya enviados pero no reconocidos todavía. Las variables *SND.UNA*, *SND.WND y SND.NXT* se utilizan para mantener los valores correspondientes al extremo izquierdo de la ventana, a la ventana ofrecida y al siguiente

número de secuencia a ser enviado, respectivamente. Por lo tanto, la ventana utilizable es igual a *SND.UNA+SND.WND-SND.NXT*.

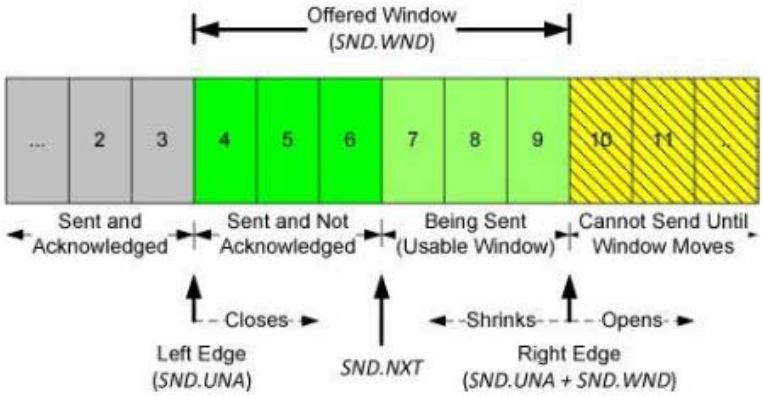


Figura 3.3 Estructura de la ventana deslizante del lado del emisor TCP [30]

Con el tiempo, la ventana deslizante se mueve hacia la derecha conforme el receptor reconoce los datos. El movimiento relativo de los dos extremos de la ventana incrementa o decrementa su tamaño. A continuación se mencionan tres términos que describen dicho movimiento [30]:

1. La ventana se cierra "*closes*" conforme el extremo izquierdo avanza hacia la derecha, lo cual sucede cuando los datos que han sido enviados son reconocidos, reduciendo así el tamaño de la ventana.
2. La ventana se abre "*opens*" cuando el extremo derecho se mueve hacia la derecha, permitiendo que más datos sean enviados. Esto sucede cuando el proceso receptor en el otro extremo lee los datos reconocidos, liberando espacio en el buffer de su receptor TCP.
3. La ventana se encoge "*shrinks*" cuando el extremo derecho se mueve hacia la izquierda. Los Requerimientos de Host RFC 1122 [13] rechazan fuertemente esta condición, pero TCP debe ser capaz de sobrellevar dicha situación.

El receptor mantiene también una estructura de ventana, pero algo más sencilla que la del emisor como se puede apreciar en la Figura 3.4, y que básicamente le permite controlar los datos que ya han sido recibidos y reconocidos, así como también el número máximo de

secuencia que está dispuesto a recibir. Además, el receptor TCP utiliza esta estructura para evitar el almacenamiento tanto de segmentos duplicados como de paquetes que no se deberían haber recibido (cualquier byte más allá del extremo derecho de la ventana del emisor), con el fin de asegurar la integridad de los datos que recibe.

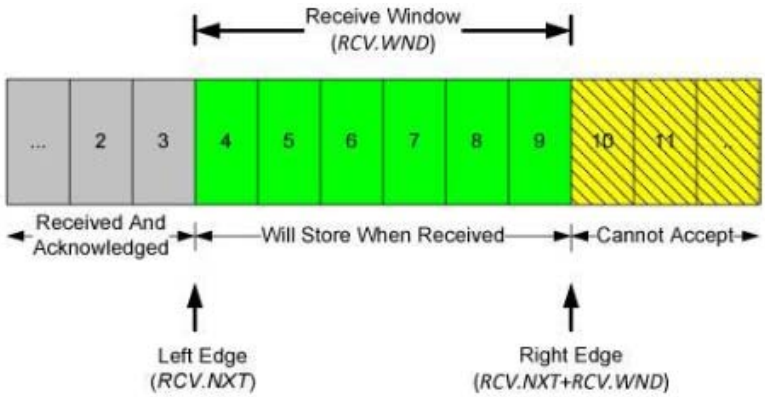


Figura 3.4 Estructura de la ventana deslizante del lado del receptor TCP [30]

Para asegurar la transferencia confiable de datos y detectar la congestión en la red, el protocolo TCP utiliza acuses de recibo (ACK) y números de secuencia que son intercambiados entre el emisor y receptor durante la sesión correspondiente. Los ACKs utilizados por TCP son acumulativos dado que un ACK que indica el número de bytes N, denota que todos los bytes hasta el número N (sin incluirlo) han sido recibidos exitosamente. Cuando llegan al receptor segmentos fuera de orden, éste los almacena y solicita al emisor el(los) segmento(s) faltante(s) a través de un ACK con el mismo número de secuencia que confirma los que ha recibido secuencialmente hasta ese momento. Este reconocimiento denominado ACK duplicado, indica al emisor sobre un posible segmento perdido; sin embargo, en vista de que existe la posibilidad de que dicho paquete sólo se haya retrasado, el emisor espera recibir al menos tres *DupAcks* (4 ACKs idénticos) para deducir que ha ocurrido un evento de pérdida. Es así que algunas variantes de TCP hacen uso de los ACKs duplicados para detectar segmentos perdidos e iniciar un proceso de retransmisión como se describirá más adelante. Además, por cada ronda de segmentos trasmitidos, el emisor inicia un temporizador esperando la confirmación de los paquetes recibidos, por lo tanto, después de la expiración del

temporizador (*timeout*), los segmentos se consideran perdidos si no son reconocidos. Si el emisor no recibe ningún ACK antes de que expire dicho temporizador o que la ventana no pueda crecer más allá de 3 segmentos, percibe tal situación como una seria congestión.

La primera versión del protocolo TCP establece que un receptor TCP envía un acuse de recibo por cada paquete entrante. Este comportamiento fue modificado posteriormente por la RFC 1122 [13] que especifica el algoritmo de ACK retardado, el cual puede incrementar la eficiencia tanto en el Internet como en los hosts enviando menos tráfico a través de la red, que cuando no se retrasan los ACKs porque se utilizan menos reconocimientos. Específicamente, el ACK se debería generar por al menos cada dos segmentos de tamaño completo y dentro de 500 ms desde el arribo del primer paquete no reconocido. La implementación más común es que TCP cuente con un temporizador que expira cada 200 ms. Por consiguiente, un ACK se envía cuando se recibe un segundo segmento de tamaño completo o de lo contrario cuando expira el temporizador de *timeout*. La Figura 3.5 muestra cómo trabaja la implementación del algoritmo de ACK retardado.

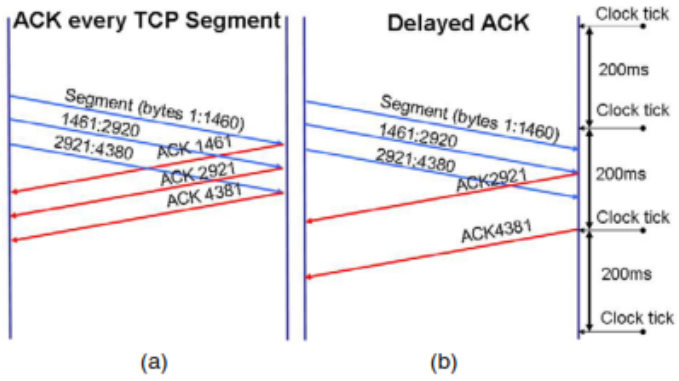


Figura 3.5 Ejemplo del comportamiento de ACKs. (a) Cada segmento es reconocido por el receptor. (b) Funcionamiento de ACK retardado [42]

3.2.1 Fases en el control de congestión

Existen diferentes fases en el mecanismo de control de congestión de TCP, que controlan esencialmente la tasa de transmisión en la red para evitar la congestión y la retransmisión de

paquetes perdidos. Estas fases se encargan de sondear gradualmente el ancho de banda disponible en la red, a medida que se transmiten los datos, con el fin de evitar la congestión. Cuando se pierden algunos paquetes debido a la congestión, la fase de retransmisión/recuperación es utilizada para reenviar tanto los segmentos perdidos como los nuevos paquetes. Adicionalmente, estas fases controlan el tamaño de la ventana de congestión que se utiliza para determinar el número de segmentos que se pueden enviar en cada ronda. A continuación se describen brevemente las distintas fases del control de congestión realizadas por el protocolo TCP:

3.2.1.1 Slow start

La versión original de TCP iniciaba una conexión con el emisor transmitiendo múltiples segmentos dentro la red, hasta el tamaño máximo de la ventana anunciada por el receptor (*rwnd*), donde las redes locales no eran afectadas tanto como la comunicación entre redes diferentes en las que los paquetes debían atravesar distintos enrutadores y enlaces de menores capacidades. Con el fin de superar este inconveniente, se propuso el algoritmo de *slow start* en el que TCP sondea el espacio del *buffer* libre en la red (en busca de capacidad disponible), mediante el incremento gradual del número de segmentos enviados en cada ronda para evitar congestionar el canal; esto en función de que la tasa a la cual se deben inyectar nuevos paquetes dentro de la red, es la tasa a la que retornan los reconocimientos desde el otro extremo.

En este sentido y como se mencionó anteriormente, se define en el emisor otra ventana denominada ventana de congestión *cwnd* con el fin de evitar el desbordamiento del *buffer* en los enrutadores intermedios, que se establece inicialmente a un valor pequeño dependiente del tamaño máximo de segmento, SMSS (*Sender Maximum Segment Size*)[35] y se incrementa en un segmento por cada reconocimiento (ACK) recibido, dando lugar a que *cwnd* duplique su valor

[35] Se debe tener en cuenta el tamaño máximo de segmento (MSS) para el incremento de *cwnd*, puesto que su valor es expresado en bytes y no en segmentos como lo considera el algoritmo de *slow start*.

con cada RTT y continúe creciendo exponencialmente hasta que alcanza un umbral preestablecido denominado *ssthreshold* (*slow start threshold*) o se detecte un evento de pérdida. El valor de *ssthreshold* puede ser alto (algunas variantes lo ajustan al tamaño de la ventana anunciada por el receptor), pero se reduce cuando TCP experimenta congestión.

La fase de *slow start* se utiliza siempre en dos casos: al inicio de una nueva conexión TCP y cuando la transferencia se reanuda después de un período de *timeout*. También se puede emplear para reiniciar la transmisión luego de un largo período de inactividad, en cuyo caso, si TCP no ha recibido ningún segmento por más de un RTO, el valor de *cwnd* se reduce a la ventana de reinicio RW = min (IW, *cwnd*). En caso contrario, *cwnd* se establece al valor de IW.

Normalmente TCP inicia una nueva conexión en *slow start*, eventualmente descarta un paquete y luego pasa al estado estable de operación utilizando el algoritmo de *congestion avoidance*. Iniciar la transmisión en una red con condiciones desconocidas requiere que TCP pruebe lentamente el canal para determinar la capacidad disponible, a fin de evitar congestionar inapropiadamente la red con una gran ráfaga de datos. Un emisor TCP inicia en *slow start* enviando un cierto número de segmentos (después del intercambio SYN), denominado ventana inicial IW, cuyo valor fue originalmente definido como un SMSS, aunque en el RFC 5681 [3] se permite que sea mayor, de acuerdo a lo siguiente:

IW = 2*SMSS y no más de 2 segmentos (Si SMSS > 2190 bytes)

IW = 3*SMSS y no más de 3 segmentos (Si 1095 < SMSS ≤ 2190 bytes)

IW = 4*SMSS y no más de 4 segmentos (Si SMSS ≤ 1095 bytes)

Aunque el valor de la ventana inicial se definió originalmente a lo mucho como 4 segmentos, su incremento a un número mayor puede beneficiar la latencia del tráfico HTTP que en su mayoría corresponde a conexiones de corta duración que finalizan antes de salir de la fase de *slow start*, lo cual se ha derivado de un estudio realizado en los últimos años por algunos autores de Google [28] en el que proponen incrementar el tamaño de IW en al menos diez segmentos, y están realizando esfuerzos para su estandarización por parte de la IETF (*Internet Engineering Task Force*), que al momento se encuentra en estado experimental bajo la RFC 6928 [23]. La

iniciativa implicaría un pequeño cambio frente a un beneficio significativo en cuanto a una mejora en la latencia promedio de las transacciones web cortas, de aproximadamente un 10%, que se traduce finalmente en una mayor velocidad en las conexiones sin la necesidad de recurrir a mecanismos más complejos propuestos en la literatura como TCP Fast Start, CM (*Congestion Manager*), STCP (*Stream Control Transmission Protocol*), TCP-SF (*TCP Smart Framing*), etc. [54], que se encuentran fuera del alcance de este proyecto de tesis.

En vista de que la asignación para IW puede admitir varios paquetes en la ventana inicial, se discutirá el caso donde IW = 1 SMSS por simplicidad. Un emisor TCP que acaba de iniciar, establece su conexión con *cwnd* = 1 SMSS, es decir que la ventana utilizable W es también igual a 1 SMSS. Nótese que en la mayoría de los casos, el SMSS es igual al menor entre el MSS del receptor y el MTU de la ruta. Asumiendo que no se pierden paquetes y que cada uno origina el envío de un ACK como respuesta, un ACK es enviado para el primer segmento, permitiendo al emisor TCP enviar otro segmento, y así sucesivamente. Aunque tradicionalmente las implementaciones de TCP han incrementado (en esta fase) el valor de *cwnd* precisamente en SMSS bytes por cada ACK válido recibido (que reconoce acumulativamente nuevos datos), la RFC 5681 [3] recomienda incrementar la ventana de congestión como *cwnd* +=min (N, SMSS), donde N es el número de bytes reconocidos por el ACK.

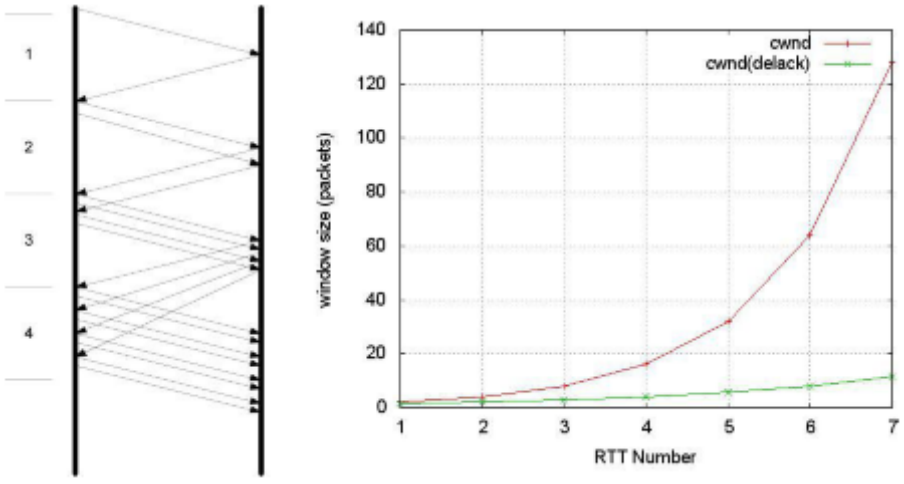


Figura 3.6 Operación del clásico algoritmo slow start [30]

Por lo tanto, después de que un segmento es reconocido, el valor de *cwnd* es ordinariamente incrementado a 2, y dos segmentos son enviados. Si cada uno de estos origina el retorno de nuevos ACK válidos, de 2 se incrementa a 4, de 4 a 8, etc (ver Figura 3.6).

En general, asumiendo que no existen pérdidas y se recibe un ACK por cada paquete, el valor de W después de k intercambios de round-trip es $W = 2^k$. Reescribiendo esta relación de otra manera, se puede decir que $k = \log_2 (W)$ RTTs se requieren para alcanzar una ventana operativa de W.

Este crecimiento parece bastante rápido (incremento como una función exponencial[36]) pero todavía es lento en comparación del que TCP debería experimentar si le fuera permitido enviar inmediatamente una ventana de paquetes igual en tamaño a la ventana anunciada por el receptor, como lo hacía este protocolo en sus inicios; de aquí su nombre de *slow start*.

Cabe mencionar que el crecimiento exponencial de *slow start* debe ser verificado en algún punto, de lo contrario conduciría rápidamente a una congestión de la red. Por lo tanto, se define un algoritmo adicional denominado *congestion avoidance* que se describe a continuación.

3.2.1.2 Congestion avoidance

Cuando se alcanza el nivel de umbral en *slow start*, siempre hay la posibilidad de que exista más capacidad disponible para una conexión. Si dicha capacidad pudiera ser utilizada inmediatamente con grandes ráfagas de tráfico, otras conexiones con paquetes TCP que comparten las mismas colas en los enrutadores, podrían experimentar pérdidas significativas de paquetes, conduciendo a una inestabilidad total de la red dado que muchas conexiones reaccionarían ante esta situación con retransmisiones de datos. Es así que TCP implementa el algoritmo de *congestion avoidance* para abordar el problema de intentar encontrar capacidad adicional que puede llegar a estar disponible, pero no de una manera tan agresiva.

[36] La denominación de *slow start*, aun cuando el crecimiento de *cwnd* es exponencial, se debe a la comparación con TCP sin control de congestión, que como se mencionó al inicio de esta subsección, era la operación por defecto de este protocolo en su primera versión.

Después de que *cwnd* alcanza el nivel de umbral en *slow start*, que se ajusta dinámicamente a través de una variable denominada *ssthreshold*, el protocolo TCP implementa el algoritmo de *congestion avoidance* en busca de capacidad adicional, que impone un crecimiento lineal de *cwnd* durante esta fase de operación de TCP, en la cual, la ventana (p.ej. de tamaño W) se incrementa en 1/W cada vez que se recibe un ACK. Por lo tanto, la ventana se incrementa en un segmento al final de cada RTT. El incremento lineal de la ventana continúa hasta que se alcanza su tamaño máximo o hasta que se detecta una pérdida de paquetes. En caso de que el emisor detecte este evento (indicado por ACKs duplicados) el valor de *ssthreshold* se fija a la mitad del tamaño de la ventana en el momento de la pérdida. Si no se reciben ACKs antes de la expiración del temporizador de retransmisión (conocido como un evento de *timeout*), el tamaño de la ventana se fija a un segmento y el protocolo TCP retorna a la fase de *slow start*. Además, cada vez que el temporizador de retransmisión expira, el emisor duplica el valor del tiempo de espera de retransmisión (RTO), cuya técnica de actualización del RTO se conoce como *backoff exponencial*.

El incremento aditivo de la fase de *congestion avoidance* y el decremento multiplicativo de *ssthreshold* es conocido usualmente como el algoritmo AIMD (*Additive Increase Multiplicative Decrease*).

3.2.1.3 Fast retransmit/Fast recovery

Debido a que la retransmisión de paquetes perdidos detectados por *timeout* origina tiempos inactivos de transferencia y una reducción abrupta en el *throughput* de TCP, se plantea una mejora en este sentido a través de dos mecanismos: *fast retransmit* y *fast recovery*, que normalmente se implementan juntos, para permitir una retransmisión de segmentos sin necesidad de esperar a que expire el temporizador de *timeout* RTO y una recuperación de la ventana de congestión de manera controlada en función del umbral *ssthreshold* y de los segmentos que se encuentran en tránsito *flightSize* (cantidad de datos enviados pero aún no reconocidos), pasando finalmente a la fase de *congestion avoidance* en lugar de reiniciar la transmisión en *slow start*.

Cuando un receptor TCP recibe segmentos fuera de orden debe enviar inmediatamente al emisor un ACK duplicado (*DupAck*), con el fin de informar esta situación y señalar cual es el número de secuencia real esperado, es decir que, vuelve a asentir el último segmento correctamente aceptado como un indicativo de que existe un hueco en la secuencia de datos recibidos. Desde la perspectiva del emisor, los ACKs duplicados pueden ser ocasionados por una de las siguientes razones: 1) paquetes perdidos, en cuyo caso cada segmento recibido después del que se perdió desencadenará un *DupAck*; 2) reordenamiento de segmentos en la red, propio de su envío a través del protocolo IP; 3) replicación de ACKs o segmentos de datos en la red. El mecanismo de *fast retransmit* utiliza los ACKs duplicados para deducir pérdidas de segmentos y tomar decisiones de retransmisión. Puesto a que no se conoce la causa que desencadena un *DupAck*, TCP espera recibir un pequeño número de ACKs de este tipo, denominados *dupthresh* (*DupAck Threshold*) antes de concluir que un segmento se ha perdido e iniciar su retransmisión. El algoritmo de *fast retransmit* se basa en la recepción de tres ACKs duplicados (conocidos como triple *DupAcks* [37]) o más, como una fuerte indicación de que un segmento se ha perdido debido a congestión en la red y retransmite inmediatamente el paquete perdido sin esperar a que expire el temporizador de retransmisión; además, puede transmitir datos adicionales que no han sido enviados todavía.

Después de retransmitir el segmento perdido, el emisor inicia el algoritmo de *fast recovery* que se utiliza para enviar nuevos datos hasta que llega un ACK no duplicado. Este mecanismo se basa en el hecho de que un *DupAck* no indica solamente pérdida o reordenamiento de segmentos, sino también que un segmento ha abandonado la red (el receptor TCP lo ha almacenado en espera de la información faltante en los datos que arriban). *Fast recovery* permite inflar la ventana de congestión temporalmente en un SMSS por cada ACK duplicado recibido mientras se recupera, y enviar nuevos paquetes de datos si la ventana de congestión lo permite hasta que se recibe un ACK válido (no duplicado), en cuyo caso TCP sale de la fase de recuperación reduciendo *cwnd* de nuevo a su valor de pre-inflado [30].

[37] El valor de tres ACK duplicados fue escogido para prevenir retransmisiones espurias causadas por la entrega de segmentos fuera de orden.

El algoritmo de *fast recovery* depende de la implementación de TCP también llamada variante de TCP. A continuación se presentan tres variantes de TCP basadas en pérdida, las cuales difieren básicamente en la forma en que opera dicha fase.

3.2.2 Variantes populares de TCP

En esta subsección se presentan tres variantes populares de TCP basadas en pérdidas, denominadas Reno, New Reno, y SACK. Todas estas implementaciones consideran la pérdida de segmentos en la red como una indicación de congestión, donde las fases de *slow start* y *congestion avoidance* son similares, y difieren únicamente en la fase de *fast retransmit/recovery,* que mejora el *throughput* mediante la reducción del tiempo necesario para retransmitir todos los segmentos perdidos y la recuperación controlada de la ventana de congestión.

Existen también otras variantes tales como TCP Vegas [14] y TCP Westwood [40] que utilizan el retardo estimado a lo largo de la ruta para identificar la congestión; mientras que implementaciones iniciales de TCP, como TCP Tahoe [56] utiliza únicamente la expiración del temporizador de retransmisión para detectar las pérdidas de paquetes, conduciendo a que TCP reinicie la transmisión en la fase de *slow start* con un tamaño de ventana de un segmento y un *ssthresh* reducido a la mitad del tamaño de la ventana en el instante de la pérdida, que como se mencionó anteriormente, no sólo conduce a largos períodos de inactividad en espera de que el temporizador de retransmisión expire sino también a una reducción significativa del *throughput*.

3.2.2.1 TCP Reno

La Figura 3.7 muestra la evolución de la ventana de congestión para el caso de TCP Reno, que utiliza los algoritmos de *slow start* y *congestion avoidance* discutidos anteriormente.

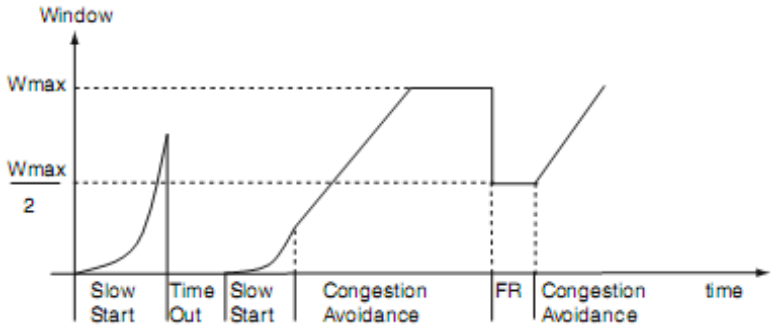


Figura 3.7 Evolución de la ventana de TCP Reno [111]

A diferencia de Tahoe, esta variante de TCP emplea el algoritmo de *fast retransmit* evitando esperar por un *timeout* para detectar la pérdida de paquetes, incorporando además un nuevo algoritmo denominado *fast recovery* para evitar reducir drásticamente la tasa de transmisión de TCP cuando se experimentan pérdidas. Normalmente los algoritmos de *fast retransmit* y *fast recovery* se aplican conjuntamente como se indica a continuación:

1) Una vez detectada la pérdida, se establece el valor de *ssthresh* a la mitad del tamaño de la ventana de transmisión pero nunca menor a 2 veces el tamaño máximo de segmento que puede enviar el emisor, es decir *ssthresh*= max *(flight s ize/* 2,2**SMSS)* .

2) Se retransmite el segmento perdido y se establece *cwnd* al valor de *ssthreshold* más 3 veces el tamaño máximo el segmento que puede enviar el emisor (*ssthreshold* + 3*SMSS). Esto infla artificialmente la ventana de congestión por el número de segmentos recibidos, en este caso 3.

3) Por cada ACK duplicado recibido se incrementa temporalmente *cwnd* en un SMSS. Esto infla artificialmente la ventana de congestión para reflejar el segmento adicional recibido, y se transmite un nuevo segmento siempre y cuando el número de datos en tránsito *flight size* sea menor que el tamaño de la ventana *cwnd*, manteniendo así principio de *self-clocking* en base a los ACKs recibidos y por tanto el flujo del tráfico.

4) Al recibir un ACK parcial[38] o total que reconoce parte o todos los segmentos pendientes respectivamente, se asume que TCP se ha recuperado y por ende se remueve el

[38] ACKs que reconocen nuevos datos, pero no todos los datos pendientes cuando se detectó la pérdida.

incremento temporal de *cwnd* retornando al valor de *ssthresh* (indicado en el paso 1), lo cual implica un desinflado de ventana, para ingresar finalmente en la fase de *congestion avoidance*.

El mecanismo de *fast retransmit/recovery* ofrece un buen desempeño en el caso de pérdidas ocasionales, mientras que cuando existen múltiples pérdidas en la misma ventana de datos, el algoritmo no se recupera con facilidad. Esto se debe al decremento multiplicativo de la ventana de congestión *cwnd* una vez que termina la fase de *fast recovery* al recibir un ACK parcial. Después de estas múltiples reducciones, *cwnd* llega a ser tan pequeña que no hay suficientes *DupAcks* para que ocurra *fast recovery*, conduciendo más bien a un *timeout* que es la única opción que queda para continuar con la retransmisión, lo cual resulta en un desempeño deficiente de TCP Reno para este caso.

3.2.2.2 TCP New Reno

Debido a que en TCP Reno la recepción de un ACK parcial en un escenario de múltiples pérdidas conduce a una reducción progresiva del *throughput* hasta reiniciar completamente su transmisión luego de un *timeout*, la IETF (*Internet Engineering Task Force*) propuso una versión mejorada denominada TCP New Reno especificada en la RFC 6582 [46] que (reemplaza las definiciones previas en RFC 3782 y RFC 2582), modifica la fase de *fast recovery* de Reno para hacerlo más robusto frente al caso de múltiples pérdidas, para lo cual TCP utiliza una nueva variable denominada *recover* a fin de almacenar un punto de recuperación que permite identificar cuando se ha superado dicha situación. Cuando se recibe un ACK parcial (número de secuencia menor al valor de *recover*), se retransmite inmediatamente el siguiente segmento y el emisor no espera por el triple ACK duplicado para detectar la pérdida de otros paquetes en la misma ventana. A diferencia de Reno para el caso de múltiples pérdidas, TCP New Reno no espera por el temporizador de retransmisión sino que continúa retransmitiendo segmentos perdidos por cada ACK parcial recibido y se mantiene en la fase de recuperación.

El valor preliminar de la variable *recover* es establecido al número de secuencia enviado inicialmente. La fase de *fast recovery* inicia con la recepción de tres *DupAcks* al igual que el caso

anterior, pero termina únicamente cuando se produce un *timeout* o si se recibe un ACK que reconoce todos los datos que estaban pendientes hasta e incluido el número de secuencia almacenado en *recover* (antes de que TCP ingresara en la fase de *fast recovery*), evitando en este último caso reducir la ventana de congestión múltiples veces como ocurre con TCP Reno.

En el instante que se recibe el tercer ACK duplicado, el emisor verifica si el campo de reconocimiento acumulativo abarca más que el valor de *recover*. Si es así, *ssthresh* se establece al valor del paso 1 para el caso de TCP Reno, y la variable *recover* se incrementa al valor del número secuencia más alto transmitido hasta ese momento (el último segmento enviado antes de la retransmisión del segmento perdido que condujo a la fase de *fast recovery*). En caso contrario, TCP no ingresa en la fase de *fast retransmit/recovery* y no cambia *ssthresh*. Posteriormente el comportamiento es similar al caso de TCP Reno (pasos 2 y 3 antes mencionados) hasta que llega un asentimiento. Si el ACK recibido es parcial, se retransmite el primer segmento no reconocido, luego se desinfla *cwnd* por el número de nuevos datos asentidos por el campo de reconocimiento acumulativo y se añaden SMSS bytes a la ventana de congestión; además se envía un nuevo segmento si el valor de *cwnd* lo permite. Este intento de "desinflado parcial de ventana" asegura que, cuando *fast recovery* finalmente termine, aproximadamente la cantidad *ssthresh* de datos estará pendiente en la red. En este caso, TCP no abandona la fase de *fast recovery*, por lo que si arriba posteriormente cualquier *DupAck*, se ejecuta el paso 3 antes mencionado para el caso de TCP Reno.

Por el contrario, si el ACK es total, es decir que se reconocen todos los segmentos pendientes hasta e incluyendo el valor de *recover*, la ventana de congestión se puede establecer como *a*) $cwnd = \min(ssthresh, \max(flightsize, SSMS) + SSMS)$ o *b*) $cwnd = ssthresh$[39], donde *sshthresh* es igual al valor determinado en el paso 1 para el caso de TCP Reno; para finalmente salir de la fase de *fast recovery*.

[39] Cuando esta opción es seleccionada, se recomienda que la implementación tome las medidas para evitar una posible ráfaga de datos, en caso que la cantidad de segmentos pendientes en la red sea mucho menor de lo que permite la nueva ventana de congestión. Un mecanismo simple es limitar el número de paquetes de datos que se pueden enviar en respuesta a un solo acuse de recibo y se conoce como *"maxburst_"* en el simulador NS [46].

Cabe mencionar además que para el primer ACK que arriba durante la fase de *fast recovery*, se reinicia el temporizador de retransmisión RTO, y que después de que ocurre un RTO, se almacena en *recover* el número de secuencia más alto transmitido.

TCP New Reno es una variante popular de las implementaciones de TCP modernas, que no sufre de los problemas del algoritmo *fast recovery* original y es significativamente menos complejo de implementar que el esquema de reconocimiento selectivo que se describirá a continuación.

3.2.2.3 TCP SACK

A pesar de la mejora introducida en la fase de *fast recovery* de TCP New Reno para el caso de múltiples pérdidas de paquetes, cuando se descartan segmentos de una sola ventana de datos, un emisor TCP puede recuperarse a lo mucho de un segmento perdido por RTT o disponerse a enviar retransmisiones innecesarias de paquetes que ya podrían haber sido recibidos; esto debido a la información limitada disponible en las notificaciones de reconocimiento acumulativo. Dado que el mecanismo de ACK parcial es un tanto deficiente, TCP es incapaz de distinguir entre *DupAcks* ocasionados por retransmisiones espurias[40] y aquellos que indican adecuadamente un segmento perdido o retrasado. Por ejemplo, si los paquetes se reordenan en la red en más de tres segmentos, TCP ingresa erróneamente en la fase de *fast retransmit/recovery*, pero a medida que se reciben los segmentos reordenados, el número de reconocimiento avanza y el emisor malinterpreta estos ACKs adicionales por ACKs parciales que indican "vacíos", y retransmite los segmentos que ya se han recibido [76].

Para superar estos inconvenientes y mejorar el desempeño del protocolo para el caso de múltiples pérdidas, surge otra variante de TCP denomina SACK (*Selective Acknowledge*), que no

[40] Retransmisiones indeseables que TCP puede iniciar incluso cuando no se han perdido datos, y pueden ser causadas por timeout espurios "tiempos de espera con expiración prematura que ocurren cuando el RTT se ha incrementado significativamente más allá del RTO, y más frecuentemente en entornos donde los protocolos de capas inferiores tienen un rendimiento muy variable, p.ej. wireless", u otras razones como reordenamiento de paquetes, duplicación de paquetes, o pérdida de ACKs [30].

cambia el significado del campo ACK, y utiliza parte del campo de opciones de la cabera TCP para enviar información adicional en los paquetes ACK que simplemente el número de secuencia del siguiente segmento esperado; es decir, el receptor envía al emisor la confirmación de todos los datos que han llegado correctamente. Esta realimentación permite al emisor mantener una imagen de la información existente en la cola del receptor, con la cual el emisor puede deducir los segmentos que faltan y así retransmitir únicamente aquellos que se han perdido sin esperar por el temporizador de retransmisión. Sólo cuando no hay paquetes que requieren ser retransmitidos y la ventana lo permite, el emisor envía nuevos segmentos de datos.

TCP SACK definido en la RFC 2018 [78] y extendido más tarde en la RFC 2883 [33] introduce el uso de la opción SACK para reconocer la retransmisión de segmentos particulares perdidos y tratar con reconocimientos duplicados, respectivamente. Añadir SACK a TCP no cambia la base de los algoritmos de control de congestión descritos en las secciones anteriores. La principal diferencia con respecto a New Reno radica en la fase de *fast recovery*, permitiendo retransmitir más de un segmento perdido por RTT, y su limitación está en la cantidad de bloques SACK que se puede enviar con relación al número de paquetes perdidos para llenar los vacíos en la secuencia de datos en el receptor. Al igual que en los casos anteriores, esta versión de TCP recurre al mecanismo de retransmisión por *timeout* si el emisor no recibe a tiempo los ACKs correspondientes.

La notificación de recepción selectiva, SACK, incluye dos tipos de información en el espacio opciones de la cabecera TCP. La primera es una opción de habilitación de 2 bytes de longitud denominada *"SACK permitted"* (ver Figura 3.8), enviada durante el establecimiento de una conexión en un segmento SYN (o SYN + ACK) para señalar a los *peers* TCP que tienen la capacidad de anunciar/recibir las indicaciones de reconocimiento selectivo. La segunda es la opción SACK que contiene propiamente la información de ACK selectivo que consiste de una lista de bloques no contiguos de los datos recibidos y almacenados en el receptor, donde cada bloque SACK está constituido por dos números enteros sin signo de 32 bits (ver Figura 3.9), los cuales pueden ser transmitidos en cualquier segmento una vez que el emisor ha enviado la opción *"SACK permitted"*.

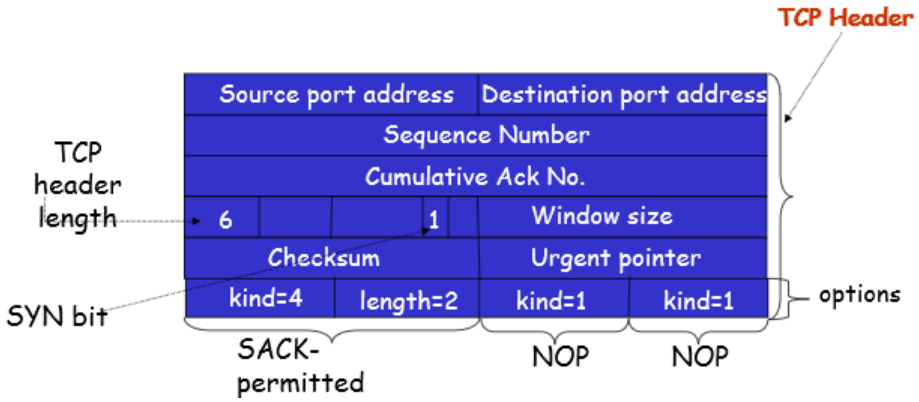


Figura 3.8 Opción SACK-permitted enviada al inicio de una conexión TCP [90]

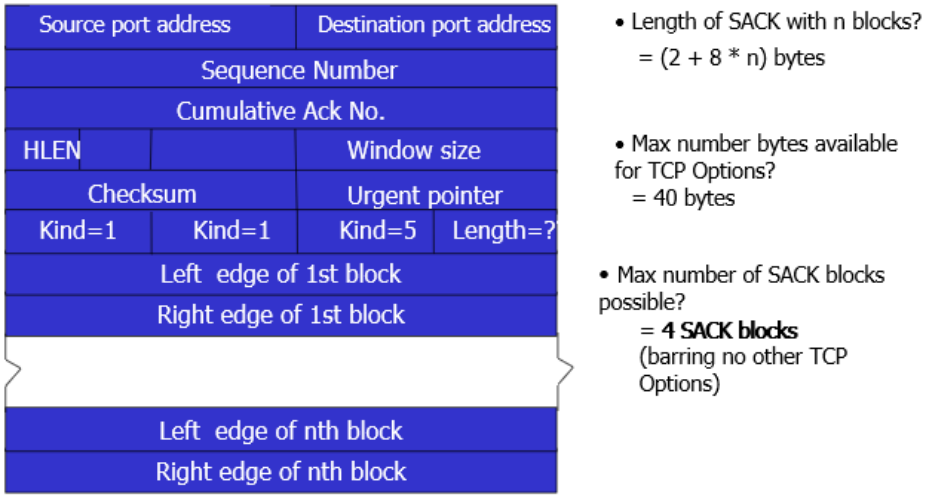


Figura 3.9 Formato de la opción SACK [90]

El primer bloque se utiliza como borde izquierdo para indicar el primer byte reconocido de la secuencia de ese bloque, mientras que el segundo se emplea como borde derecho para indicar el número de la secuencia que sigue inmediatamente al último byte reconocido de ese bloque[41]. Un grupo de n bloques SACK utiliza un total de 8*n+2 bytes de longitud, por lo que considerando los 40 bytes disponibles en el campo opciones de la cabecera TCP, se pueden

[41] Cada bloque representa los bytes de datos recibidos que están contiguos y aislados; es decir, los bytes justo antes del bloque, (borde izquierdo del bloque - 1), y justo después del bloque, (borde derecho del bloque), no han sido recibidos [13].

especificar un máximo de 4 bloques. Sin embargo, como TCP SACK se utiliza normalmente en conjunto con la opción *Timestamp* que emplea 10 bytes adicionales (más 2 bytes de alineación), se podrá utilizar un máximo de 3 bloques por ACK. Un paquete ACK que contiene uno o más bloques SACK es a veces llamado simplemente "SACK", donde el primer bloque incluye siempre la información de los datos reconocidos más recientemente.

Por lo general, un receptor genera bloques SACK siempre que hay algún dato fuera de orden en su *buffer*. El primer bloque debe especificar el rango contiguo de bytes (datos) que contienen el segmento que desencadenó este ACK, a menos que dicho segmento incremente el campo de número de reconocimiento de la cabecera TCP, lo cual asegura que el ACK con la opción SACK reporta el paquete más reciente en la cola del *buffer* del receptor de datos. Los siguientes bloques se rellenan repitiendo los bloques SACK que fueron enviados recientemente según el primer bloque de las opciones SACK anteriores, asegurando que en una operación normal cualquier parte del segmento que quede de un bloque no continuo de datos almacenado por el receptor, se notificará en al menos tres opciones SACK sucesivas. Después del primer bloque SACK, los siguientes bloques de la opción SACK pueden estar listados en orden arbitrario. Por otro lado, el emisor debe tratar estos bloques apropiadamente y ejecutar una retrasmisión selectiva enviando sólo los segmentos que faltan en el receptor, para lo cual realiza un seguimiento de toda la información ACK acumulativa que recibe (como cualquier emisor TCP), más cualquier indicación SACK que recepta, y las utiliza para evitar retransmitir datos que el receptor reporta que ya tiene.

La RFC 6675 [11] (que reemplaza la definición previa en RFC 3517) especifica cómo recuperarse de la pérdida de segmentos cuando se utilizan reconocimientos selectivos, en base a un algoritmo conservador de recuperación de pérdidas que emplea la opción SACK de TCP. Como en los casos anteriores, cuando el emisor recibe tres *DupAcks* TCP ingresa en la fase de *fast retransmit/recovery*, en la cual retransmite el segmento perdido y reduce a la mitad la ventana de congestión, y al igual que el caso de New Reno, la fase de *fast recovery* termina cuando se produce un *timeout* o si se recibe un ACK que reconoce todos los datos hasta e incluido el punto de recuperación *RecoveryPoint*. Para realizar un seguimiento de la información SACK obtenida de los ACK entrantes, se utiliza una estructura de datos denominada *"scoreboard"* que se actualiza a medida que se envían los paquetes, se procesan los SACKs, y se reciben

reconocimientos acumulativos. Además, el algoritmo mantiene una variable de estado denominada *pipe* que estima el número de segmentos pendientes que se encuentran en tránsito en la red, y que dependiendo de su implementación puede contabilizar explícitamente bytes o paquetes; de esta manera se consigue separar el manejo de congestión (cuando enviar) de la selección y el mecanismo de retransmisión de paquetes (que enviar), resultando en una mejor capacidad de estimación del número de segmentos pendientes, a diferencia de las variantes convencionales de TCP sin SACK que combinan estas dos funciones.

La RFC 6675 define seis variables (ver Tabla 3.1) que almacena el emisor y cuatro funciones (ver Tabla 3.2) que debería implementar el *scoreboard*, para gobernar la operación del algoritmo.

Tabla 3.1 Variables almacenadas por un emisor en RFC 6675 [11]

Variable	Descripción
HighACK	Número de secuencia del byte de datos más alto que ha sido reconocido acumulativamente en un punto dado.
HighData	Número de secuencia más alto transmitido en un punto dado.
HighRxt	Número de secuencia más alto que ha sido retransmitido durante la fase actual de recuperación de pérdidas.
RescueRxt	Número de secuencia más alto que se ha retransmitido con optimismo para prevenir el estancamiento del reloj de ACK, cuando hay pérdidas al final de la ventana o no existen nuevos datos disponibles para la transmisión.
Pipe	Estimación del emisor del número de bytes pendientes en la red entre el remitente y receptor TCP, que se utiliza durante la fase de recuperación para limitar la tasa de transmisión del emisor.
DupAcks[42]	Número de reconocimientos duplicados recibidos desde el último ACK acumulativo.

[42] Para efectos de esta especificación, se define un *DupAck* como un segmento que llega transportando un bloque SACK que identifica octetos previamente no reconocidos y no confirmados de manera selectiva entre *HighACK* y *HighData*. Así, un ACK que transporta nueva información SACK es contabilizado como un reconocimiento duplicado incluso si transporta nuevos datos, cambia la ventana anunciada, o mueve el punto de reconocimiento acumulativo, lo cual es diferente de la definición de *DupAck* en la RFC 5681 [11].

Tabla 3.2 Funciones que debería implementar el scoreboard en la RFC 6675 [11]

Función	Descripción
Update()	Marca de manera consecuente cada octeto reconocido acumulativa o selectivamente en el *scoreboard*, según la información provista en un ACK, y registra el número total de octetos acumulados selectivamente.
IsLost (*SeqNum*)	Determina si el número de secuencia dado se considera perdido o no. Devuelve "verdadero" cuando han arribado *DupThresh* secuencias discontinuas reconocidas selectivamente por encima de *'SeqNum'* o se han reconocido selectivamente más de (*DupThresh* - 1) * SMSS bytes con números de secuencia mayores que *'SeqNum'*. En caso contrario devuelve "falso".
SetPipe()	Atraviesa el espacio de secuencias desde *HighACK* hasta *HighData* y debe establecer la variable *pipe* a una estimación del número de octetos que están actualmente en tránsito. Después de inicializar la variable *pipe* a cero (0), por cada octeto *'S1'* en el espacio de secuencias indicado que no ha sido reconocido selectivamente, la variable *pipe* se incrementa en un (1) octeto cuando *IsLost* (*S1*) devuelve "falso" o si *S1* <= *HighRxt*. En el primer caso, *pipe* se incrementa para los paquetes que no han sido reconocidos selectivamente y no se ha determinado que se han perdido, sino que se asume que todavía están en la red. En el segundo caso, *pipe* se incrementa por la retransmisión del octeto.
NextSeg()	Utiliza el *scoreboard* mantenido por la función *Update* () para determinar que transmitir en base a la información SACK que ha llegado desde el receptor (y por ende ha sido marcado en el *scoreboard*. La rutina debe retornar el rango de números de secuencia del siguiente segmento a transmitir, de acuerdo a las siguientes reglas: 1) si existe un número de secuencia 'S2' más pequeño no reconocido selectivamente que no se ha retransmitido pero se determina que está perdido por *IsLost*; es decir, que se satisfacen los siguientes tres criterios para la detección de pérdidas: 1.a) S2 > *HighRxt*; 1.b) S2 < el octeto más alto cubierto por cualquier SACK recibido; 1.c) *IsLost* (*S2*) retorna "verdadero", se debe retornar un segmento de hasta SMSS octetos que empieza con S2; 2) si no existe el número de secuencia 'S2' para la regla 1) pero hay datos disponibles no enviados y la ventana anunciada por el receptor lo permite, se debe retornar un segmento de hasta SMSS octetos de los datos no enviados previamente que empieza con *HighData* + 1; 3) si fallan las condiciones para las reglas 1) y 2), pero existe un número de secuencia 'S3' no reconocido selectivamente que satisface los criterios para la detección de pérdidas dados en 1.a) y 1.b) (excluyendo específicamente el paso 1.c), se debe

Función	Descripción
NextSeg()	retornar un segmento de hasta SMSS octetos que empieza con 'S3'; 4) [43] si fallan las condiciones para las reglas 1), 2), y 3), pero existen datos pendientes no reconocidos selectivamente, se ofrece la oportunidad para una sola retransmisión de "rescate". Si *HighACK* > *RescueRxt* (o *RescueRxt* no está definida), se debe retornar un segmento de hasta SMSS octetos que debe incluir el número de secuencia pendiente más alto no reconocido selectivamente, y establecer *RescueRxt* = *RecoveryPoint*. *HighRxt* no se debe actualizar; 5) si no se satisfacen las condiciones para cada una de las reglas 1), 2), 3), y 4), la rutina debe indicar una condición de fallo, y no retornar ningún segmento.

Después de la recepción de cualquier ACK que contiene información SACK, el *scoreboard* debe actualizarse mediante la rutina *Update* (). Cuando el ACK entrante es un reconocimiento acumulativo, TCP debe resetear *DupAcks* a cero; mientras que si es un ACK duplicado, y TCP no está actualmente en la fase de recuperación de pérdidas, TCP debe incrementar *DupAcks* en uno y llevar a cabo los siguientes pasos [11]:

1) Si *DupAcks* ≥ *DupThresh*, ir al paso 4.

2) Si *DupAcks* < *DupThresh* pero *IsLost* (*HighACK* + *1*) devuelve "verdadero" indicando al menos tres segmentos que han llegado por encima del punto de reconocimiento acumulativo actual, que se toma para indicar la pérdida, ir al paso 4.

3) TCP puede transmitir segmentos de datos previamente no enviados según *Limited Transmit*[44] [3], salvo que el número de octetos que pueden ser enviados se rige por *pipe* y *cwnd* de la siguiente manera:

(3.1) Establecer *HighRxt* a *HighACK*.

[43] Las reglas 3) y 4) son un tipo de retransmisión de "último recurso", ya que permiten la retransmisión de números de secuencia incluso cuando el emisor tiene menos certeza de que un segmento se ha perdido que como con la regla 1). Retransmitir segmentos mediante la regla 3) y 4) ayudará a mantener el reloj ACK de TCP y por lo tanto puede contribuir potencialmente a evitar los *timeouts* de retransmisión.

[44] Modificación definida en la RFC 3042 [34] para mejorar el mecanismo de recuperación de pérdidas de TCP cuando la ventana utilizable es pequeña, permitiendo al emisor enviar hasta 2 segmentos más allá del valor de *cwnd* (que no debe cambiar), luego de la recepción de 2 *DupAcks* consecutivos y si *rwnd* lo permite. Este mecanismo sigue el principio de "conservación de paquetes" y podría desencadenar el tercer *DupAck* para iniciar la fase de *fast retransmit*.

(3.2) Ejecutar *SetPipe* ().

(3.1) Si (*cwnd* − *pipe*) ≥ 1 SMSS, existen datos previamente no enviados, y la ventana anunciada por el receptor lo permite, transmitir hasta 1 SMSS de datos que empieza con el octeto *HighData* + 1 y actualiza *HighData* para reflejar esta transmisión, a continuación volver al paso (3.2).

(3.4) Terminar el procesamiento de este ACK.

4) Invocar a *fast retransmit* e ingresar en la fase de recuperación de pérdidas como se indica a continuación:

(4.1) *RecoveryPoint* = *HighData*. Cuando el emisor TCP recibe un ACK acumulativo para este octeto de datos, la fase de recuperación de pérdidas es terminada.

(4.2) *Ssthresh* = *cwnd* = *FlightSize*/2. La ventana de congestión y el umbral de *slow start* se reducen a la mitad de *FlightSize*[45].

(4.3) Retransmitir el primer segmento de datos que se presume perdido, es decir, aquel que inicia con número de secuencia *HighACK* + 1. Para prevenir la retransmisión repetida de los mismos datos o una retransmisión prematura de rescate, se debe establecer tanto *HighRxt* como *RescueRxt* con el número de secuencia más alto del segmento retransmitido.

(4.4) Ejecutar *SetPipe* (). Establecer la variable *pipe* al número de octetos pendientes actualmente en tránsito, correspondiente a los datos que han sido enviados por el emisor TCP, pero para los cuales no se ha recibido ningún reconocimiento acumulativo o selectivo, y no se ha determinado que los datos se han perdido en la red.

(4.5) Para aprovechar la potencial ventana de congestión (*cwnd*) adicional disponible, proceder con el paso (C) que se indica más adelante.

Una vez que TCP está en la fase de recuperación de pérdidas, se debe utilizar el siguiente procedimiento por cada ACK que arriba:

A. Un ACK acumulativo entrante para un número de secuencia mayor que *RecoveryPoint* señala el final de la recuperación de pérdidas, la cual debe terminar. Cualquier información

[45] La RFC 5681 precisa que cualquier segmento enviado como parte del mecanismo de *Limited Transmit* no se considera en *FlightSize*.

contenida en el *scoreboard* para números de secuencia mayores que el nuevo valor de *HighACK* no se debería borrar al salir de dicha fase.

B. Después de la recepción de un ACK que no incluye *RecoveryPoint*, se deben tomar las siguientes acciones:

(B.1) Utilizar *Update* () para almacenar la nueva información SACK transportada por el ACK entrante.

(B.2) Emplear *SetPipe* () para recalcular el número de octetos que están todavía en la red.

C. Si *cwnd – pipe* ≥ *1 SMSS* el emisor debería transmitir uno o más segmentos de la siguiente manera:

(C.1) Se debe consultar el *scoreboard* a través de *NextSeg* () para el rango de números de secuencia del siguiente segmento a transmitir, y si hay uno, lo transmite. Si *NextSeg* () devuelve un fallo (no hay datos para enviar), retorna sin enviar nada, es decir, termina los pasos (C.1) a (C.5).

(C.2) Si uno de los octetos de datos enviados en (C.1) está por debajo de *HighData*, *High*R*xt* se debe establecer al número de secuencia más alto del segmento retransmitido, a menos que se haya invocado la regla 4 de *NextSeg* ( ) para esta retransmisión.

(C.3) Si uno de los octetos de datos enviados en (C.1) está por encima de *HighData*, esta variable se debe actualizar para reflejar la transmisión de los datos previamente no enviados.

(C.4) La estimación de la cantidad de datos pendientes en la red debe ser actualizada mediante el incremento de *pipe* por el número de octetos transmitidos en (C.1).

(C.5) Si *cwnd – pipe* ≥ *SMSS*, volver a (C.1).

En otras palabras, durante la fase de *fast recovery* se consulta el *scoreboard* y el emisor envía datos nuevos o retransmitidos, sólo cuando el valor de *pipe* es menor que la ventana de congestión *cwnd*. En primera instancia se retransmiten los paquetes perdidos; si no hay segmentos para retransmitir, se envían nuevos segmentos. La variable *pipe* se incrementa en uno cuando el emisor transmite un segmento sea nuevo o retransmitido, mientras que se reduce en uno cuando recibe un ACK duplicado con SACK informando que se han recibido nuevos datos. Para el caso de ACKs parciales, el valor de *pipe* se reduce en dos, puesto que se considera el segmento original transmitido que se perdió, así como también el segmento retransmitido. En la Figura 3.10 y Figura 3.11 se muestra una representación gráfica de la operación de TCP

SACK y de una secuencia de transmisión de datos comparativa entre TCP sin SACK y TCP con SACK, respectivamente.

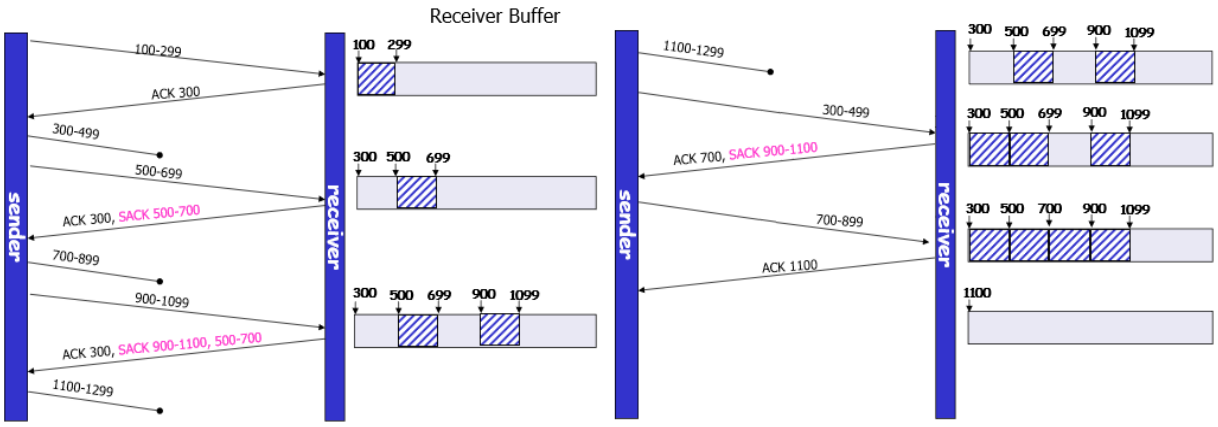


Figura 3.10 Ejemplo de la operación de TCP SACK [90]

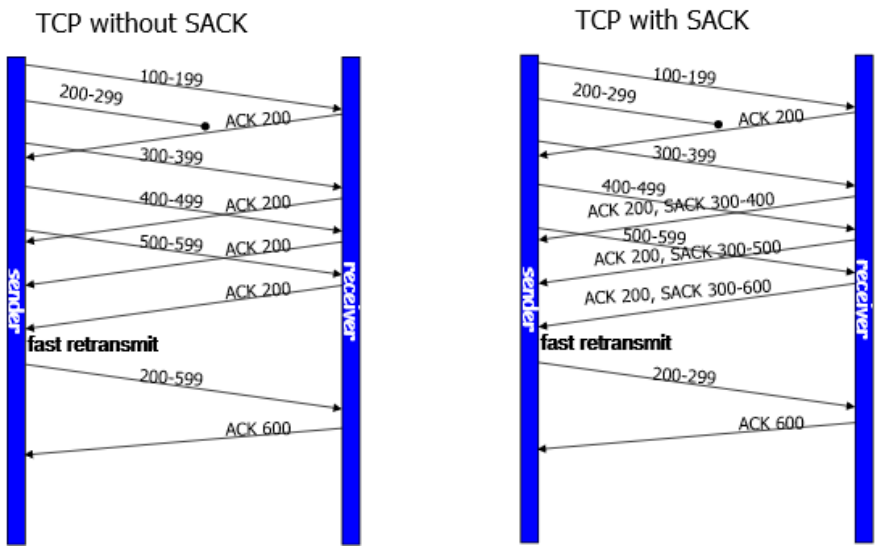


Figura 3.11 Comparativa entre TCP sin SACK y TCP con SACK [90]

El algoritmo de recuperación de pérdidas basado en SACK descrito en la RFC 6675 requiere más recursos computacionales que las estrategias de recuperación de pérdidas anteriores de TCP; sin embargo, su beneficio en cuanto al manejo de la recuperación de errores es más eficiente en el uso del ancho de banda de la red.

3.3 Factores que afectan el desempeño de TCP sobre OBS [41][107][111][127]

A diferencia de las redes tradicionales de conmutación de paquetes, la naturaleza sin almacenamiento *"bufferless"* de las redes OBS conduce a dos cambios importantes en cuanto a las características de transmisión de datos. El primero es el incremento del retardo extremo a extremo debido al ensamblado de ráfagas y señalización, y el segundo es la pérdida de ráfagas por contención que se presentan de manera aleatoria en los nodos intermedios, incluso en condiciones de bajo tráfico. Por lo tanto, el comportamiento de TCP puede variar considerablemente con respecto al que se experimenta actualmente en Internet, dado que el *throughput* de las conexiones que utilizan este protocolo de transporte depende tanto del RTT como de la probabilidad de pérdidas de paquetes como se examinará más adelante. Es así que en esta sección se estudia el impacto del algoritmo de ensamblado de ráfagas en el nodo de *edge* y de la pérdida de ráfagas por contención sobre el tráfico TCP.

3.3.1 Impacto de los algoritmos de ensamblado de ráfagas

Puesto que los datagramas IP que contienen segmentos TCP se agregan en los nodos de ingreso en ráfagas de datos para su posterior transmisión a través de la red OBS, sufren de un retardo adicional debido al tiempo de espera que demanda completar la formación de cada ráfaga, que resulta en un incremento del RTT de cada uno de los segmentos correspondientes y por ende impacta negativamente sobre el *throughput* de TCP. Este efecto se denomina penalidad por retardo, donde el valor del tiempo extra depende del algoritmo de ensamblado (específicamente de los valores de umbral utilizados), así como también de la tasa de llegada de entrada.

Para el algoritmo de ensamblado basados en tiempo, el retardo es establecido por el período de tiempo utilizado para el ensamblado de ráfagas, mientras que para el algoritmo de ensamblado basado en tamaño depende de la tasa de llegada de los paquetes al nodo de ingreso. Para el

caso del algoritmo de ensamblado mixto, el análisis es más complicado y el tiempo de formación de ráfagas está dado por una determinada distribución de probabilidad que puede utilizar como aproximación una distribución Gaussiana o una distribución Gamma [111].

El incremento en el RTT de cada segmento TCP puede ser escrito como [111]:

$$RTT = RTT_0 + 2x\tau \quad \textbf{(3.1)}$$

Donde τ es el tiempo promedio de espera de las ráfagas que se asume es igual en los nodos de borde donde un segmento TCP así como su ACK es agregado en una ráfaga y RTT_0 es el RTT de una ráfaga debido a otros factores que incluyen el retardo de propagación, tiempo de *offset*, tiempo de procesamiento, y tiempo de conmutación en cada nodo. En el caso de utilizar FDLs o cualquier otra técnica de resolución de contenciones en los nodos intermedios, el retardo adicional es añadido al RTT mencionado anteriormente.

Además, puesto a que TCP emplea la medida del retardo (su media y varianza) para algunos cálculos, como por ejemplo el del temporizador de retransmisión, una gran variación en su valor, puede provocar efectos adicionales inesperados en la estimación de los temporizadores [41].

Por otro lado, debido a que OBS extiende la unidad de transmisión a un bloque de datos de mayor tamaño, que resulta de la agregación de varios paquetes en una ráfaga, muchos segmentos TCP podrían ser entregados a la vez. La combinación de este proceso de ensamblado y la naturaleza sin almacenamiento del *core* OBS pueden retrasar el primer evento de pérdida para una determinada conexión TCP dentro de un período entre pérdidas, donde la entrega correlacionada de segmentos de un mismo flujo aumenta la cantidad de datos enviados exitosamente hasta que ocurre una pérdida por contención, conduciendo a un crecimiento de la ventana de congestión por un lapso de tiempo mayor antes de que sea reducida a la mitad debido a la indicación de pérdida, lo cual finalmente se traduce en una mejora en cuanto al *throughput* de TCP. Este efecto positivo de amplificación se denomina ganancia DFL (*Delayed First Loss*) o simplemente beneficio de correlación.

En otras palabras, si comparamos el comportamiento de TCP en una red convencional de conmutación de paquetes con respecto a un entorno OBS considerando la misma probabilidad de pérdidas, OBS ofrece un mayor *throughput* debido al beneficio de correlación, que se ha probado por análisis y simulación que su valor es proporcional a la raíz cuadrada de la longitud de la ráfaga [107]. En un escenario pragmático, se experimentarían entregas y pérdidas correlacionadas de segmentos, donde la pérdida de varios paquetes de una determinada ráfaga se consideraría como un único evento de congestión, mientras que en el caso tradicional serían indicativos de múltiples eventos de congestión, impactando en mayor medida el desempeño de TCP. Por lo tanto, el beneficio de correlación considera la correlación de tiempo entre los eventos de pérdida de segmentos y los eventos de entrega de segmentos; es decir que debido al proceso de agregación, una conexión TCP puede insertar un cierto número de segmentos consecutivos en la misma ráfaga de salida; así, los eventos de pérdida/entrega de ráfagas produce eventos de pérdida/entrega de segmentos consecutivos, donde el número de segmentos agregados en la misma ráfaga depende de la relación entre el ancho de banda de acceso y el período de ensamblado. Como se verá más adelante, el beneficio de correlación se puede obtener de la relación entre el *throughput* de TCP en presencia y ausencia del par ensamblador/desensamblador de ráfagas.

3.3.2 Impacto de la pérdida de ráfagas

Debido a la agregación de muchos segmentos en una ráfaga, la pérdida de una sola ráfaga conduce a una pérdida simultánea de muchos segmentos TCP. Esto afecta drásticamente el crecimiento de la ventana de transmisión en función del número de segmentos por ráfaga y la variante de TCP utilizada. Cuando una ráfaga contiene unos cuantos segmentos TCP de una ventana, su pérdida desencadena la fase de retransmisión rápida y recuperación rápida sin esperar un *timeout* en variantes tales como Reno, New Reno, SACK. En tal caso, los segmentos perdidos son retransmitidos en una o más rondas dependiendo de la variante y el número de segmentos perdidos. Sin embargo, en condiciones de baja carga, una ráfaga podría contener todos los segmentos de una ventana cuya pérdida conduce a la expiración del temporizador de

retransmisión. En este caso, el emisor considera la pérdida como aquella debido a una seria congestión e inicia desde la fase de *slow start*. Este tipo de identificar una sola pérdida o unas cuantas pérdidas debido a la contención en lugar de la congestión se denomina detección falsa de congestión. La fuente TCP percibe congestión en la red debido a la expiración del temporizador de retransmisión, aun cuando la red no está congestionada y no está muy cargada. El *timeout* en tales casos llamado FTO (*False Timeout*) desencadena la fase de *slow start* por cada pérdida de ráfagas reduciendo drásticamente el *throughput* del emisor TCP [127].

Por otro lado, las pérdidas de ráfagas pueden ser persistentes debido a la indisponibilidad de ancho de banda a lo largo del camino. En tal caso, el emisor TCP debería reducir la ventana de congestión de acuerdo con el mecanismo de *congestion avoidance*. Además, debido a los algoritmos de ensamblado de ráfagas y a las técnicas de resolución de contenciones tales como retransmisiones a nivel de ráfagas, segmentación, y enrutamiento por deflexión, las ráfagas podrían entregar los segmentos TCP en diferente orden al que se originó en el emisor, desencadenando los mecanismos de *fast retransmit/recovery*, puesto a que TCP percibe como pérdida de segmentos este tipo de eventos; donde la frecuencia de entrega fuera de orden depende de los umbrales utilizados para el ensamblado y el criterio de QoS utilizado en la agregación. A pesar de esto, se ha probado que los esquemas de resolución de contenciones permiten incrementar la confiabilidad de la transmisión y el *throughput* en redes OBS a expensas de introducir requerimientos adicionales en términos de overhead, retardo, y complejidad computacional.

Las implementaciones de TCP diseñadas para redes OBS utilizan principalmente técnicas para informar al emisor si las pérdidas detectadas son debido a la pérdida de una sola ráfaga o debido a pérdidas de múltiples ráfagas. En otras palabras, se notifica al emisor si un *timeout* es un FTO o es realmente debido a la congestión. Al emisor TCP se le da información adicional sobre el estado de la red para identificar la razón de la pérdida. Por ejemplo, se pueden utilizar técnicas de aprendizaje de máquina en el nodo de *edge* para clasificar las pérdidas en pérdidas por contención y pérdidas debido a la congestión, de manera que es posible mejorar el *throughput* de TCP evitando la reducción de la ventana de congestión debido a pérdidas por contención. Dichas técnicas o variantes de TCP para redes OBS requieren esfuerzos adicionales de señalización entre los emisores y la red OBS, y/o dentro del dominio OBS.

Se encuentra también que el *throughput* de TCP en redes OBS se mejora debido a la utilización de otras técnicas que incluyen el uso de esquemas de resolución de contenciones en los nodos de *core*. Modelar el comportamiento de TCP sobre redes OBS no sólo ayuda a entender el funcionamiento de TCP sino también a diseñar técnicas para mejorar su desempeño. La siguiente sección presenta diferentes modelos matemáticos para TCP sobre redes OBS que ayudan a entender el impacto del ensamblado de ráfagas y las pérdidas de ráfagas sobre el *throughput*.

3.4 Modelos Analíticos para TCP sobre OBS [20,25][30][74][85][92][107][111][126,129][143]

La caracterización exacta de los efectos de las pérdidas de ráfagas en el *throughput* para redes OBS es compleja debido a la falta de buenos modelos y también a la dependencia de la probabilidad de pérdida de ráfagas, BLP (*Burst Loss Probability*) en muchos factores. Sin embargo, los modelos que se presentan en esta sección son bastante precisos para analizar el protocolo TCP con algunas suposiciones sobre la evolución de la ventana y el número de segmentos descartados debido la pérdida de una ráfaga. El principal desafío en la caracterización del impacto de la pérdida de una ráfaga en el *throughput*, es evaluar el número de segmentos de un flujo TCP en dicha ráfaga.

Por otro lado, el retardo medio que sufren los segmentos TCP en una red OBS también requiere el modelado preciso del nodo de *edge* para entender su efecto sobre los valores de RTT y RTO, aspecto que no se abordará ya que se encuentra fuera del alcance de este proyecto de tesis. Debido a la interacción entre la penalidad por retardo y el beneficio de correlación, es difícil analizar el *throughput* de un único flujo TCP cuando el tráfico tiene múltiples flujos TCP, carga debido a otro tráfico de fondo, y si se tiene en cuenta los demás componentes de la red OBS [111].

Los modelos analíticos para evaluar el *throughput* de TCP sobre redes OBS se pueden clasificar en dos categorías principales: a) los que utilizan la teoría de los procesos regenerativos de Markov o teoría de renovación y b) los que usan la teoría de las cadenas de Markov. En el

primer caso, los modelos son simples de obtener y proveen expresiones elegantes para el *throughput* en términos de RTT y BLP, mientras que en el segundo caso, se evalúa el impacto de muchos otros factores con precisión pero se pueden resolver empleando sólo métodos numéricos. Dentro del alcance de esta tesis, se presenta en esta sección únicamente el primero de estos modelos para TCP sobre redes OBS. Las variantes populares de TCP basadas en pérdidas que son modeladas incluyen TCP Reno, New Reno, y SACK, donde su desempeño varía debido a los diferentes métodos de recuperación de pérdidas propios de cada variante.

3.4.1 Modelos basados en la teoría de procesos regenerativos de Markov

Un avance importante en el modelado del *throughput* en el Internet ocurrió con el popular modelo basado en la teoría de renovación para TCP Reno propuesto por Padhye [85]. Este modelo captura el efecto de las fases de retransmisión rápida y *timeout* en el *throughput* de TCP, que a pesar de muchas aproximaciones, da una elegante expresión en forma cerrada para el *throughput* en términos de la probabilidad de pérdida de paquetes y el RTT.

Existen muchos modelos analíticos para el *throughput* de TCP en redes OBS basados en el modelo de Padhye, que estudian el impacto en el *throughput* de TCP de los algoritmos de ensamblado de ráfagas y las pérdidas aleatorias en la red OBS sin almacenamiento [25,107, 126,129].

La evolución de la ventana de congestión se modela en términos de rondas, donde una ronda denota la transmisión back-to-back de una ventana de paquetes, que termina cuando se recibe el primer ACK para la ventana o cuando expira el temporizador de retransmisión. Se asume que la duración de una ronda es un RTT y que es independiente del tamaño de la ventana. Además, se considera que el número de paquetes perdidos en una ronda es independiente del de las demás rondas.

El *throughput* en el estado estacionario de un flujo TCP en una red OBS se puede expresar en términos de la tasa de pérdidas p_b y el RTT como [111]:

$$Throughput = \frac{1}{RTT_0 + 2T_b}\sqrt{\frac{S}{2bp_b}} + o\left(1/\sqrt{p_b}\right)^{46} \quad \textbf{(3.2)}$$

Donde S el tamaño medio de ráfagas en términos del número de paquetes, RTT_0 es el tiempo de ida y vuelta en la red, T_b es el tiempo de ensamblado de ráfagas, y b es el número de rondas reconocidas antes de que se incremente el tamaño de la ventana.

En referencia a la configuración de la conexión TCP utilizada dentro el modelo presentado en la Figura 3.12, la ruta de la red de acceso en la dirección de transmisión desde la fuente hacia el receptor TCP, se modela como un enlace sin pérdidas con un tiempo de retardo igual a d y una tasa de transmisión igual a B_a (denominado ancho de banda de acceso), mientras que la ruta de la red OBS se modela como un enlace con pérdidas y un retardo de propagación T_p, donde la pérdida de ráfagas sigue una distribución de Bernoulli con parámetro p_b. El tiempo de espera de la transmisión de los segmentos TCP y el retardo debido al desensamblado de ráfagas se consideran despreciables en este modelo. Por otro lado, en la dirección inversa se asume que no existe el par ensamblador/desensamblador de ráfagas y se modela esta ruta como un enlace sin pérdidas con un retardo de propagación extremo a extremo igual a T_p+2d.

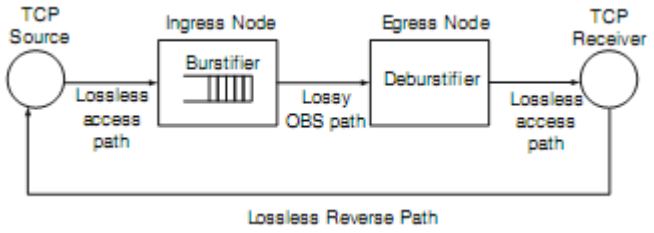


Figura 3.12 Modelo de conexión TCP [111]

En este modelo se pueden definir variables con y sin incluir los retardos contribuidos por los ensambladores de ráfagas [25], así:

[46] Básicamente, f(x) = o(g(x)) significa que f(x) está creciendo más lentamente que g(x) a medida que x se hace grande (o pequeña).

- *RTT* = El valor medio del tiempo de ida y vuelta para una conexión TCP, es decir el intervalo de tiempo entre la transmisión de un segmento y la recepción del ACK correspondiente.
- *RTTVAR* = La desviación estándar del tiempo de ida y vuelta.
- *RTO* = El valor medio del primer *timeout* de retransmisión, es decir el *timeout* de retransmisión sin ninguna duplicación de *backoff*.
- RTT_0 = El valor medio del tiempo de ida y vuelta en ausencia de los ensambladores.
- $RTTVAR_0$= La desviación estándar del tiempo de ida y vuelta en ausencia de los ensambladores.
- RTO_0= El valor medio del primer *timeout* de retransmisión en ausencia de los ensambladores.

Recordando las reglas de TCP Reno para la evaluación del RTO [30], y considerando que en el modelo de la Figura 3.12 la variación en el RTT se debe exclusivamente a los ensambladores de ráfagas, se obtiene que [25]:

$$RTT_0 = 4d + 2T_p \quad \textbf{(3.3)}$$

$$RTTVAR_0 = 0 \quad \textbf{(3.4)}$$

$$RTO_0 = RTT_0 + 4RTTVAR_0 = RTT_0 \quad \textbf{(3.5)}$$

Puesto que el retardo experimentado por un segmento debido al ensamblador está limitado al intervalo [0, T_b], el valor del RTT está limitado por RTT_0+T_b. Además, asumiendo que $RTTVAR \approx RTTVAR_0$, es decir que la variación del retardo introducida por los ensambladores no es tan alta como para incrementar significativamente la desviación estándar del RTT, y consecuentemente, el valor del primer *timeout* de retransmisión, se obtiene que [25]:

$$RTT \approx (1+\alpha) RTT_0 \quad \textbf{(3.6)}$$

$$RTO \approx (1+\alpha) RTT_0 \quad \textbf{(3.7)}$$

Donde $\alpha = \frac{T_b}{RTT_0}$. En estas expresiones el factor (1+α) representa el efecto de los ensambladores en los valores medios del RTT y del primer RTO, por lo cual se puede considerar como una medida de la penalidad por retardo introducida por el proceso de ensamblado de ráfagas.

En este modelo, las fuentes TCP se pueden clasificar de acuerdo a la relación entre el tiempo de ensamblado y el ancho de banda de acceso en clases 'lenta', 'media' y 'rápida', de tal manera que satisfacen las siguientes desigualdades [111]:

$$\textbf{Fuente lenta: } \frac{B_a T_b}{L_p} \leq 1 \quad \textbf{(3.8)}$$

$$\textbf{Fuente rápida: } \frac{B_a}{W_m L_p} T_b \geq 1 \quad \textbf{(3.9)}$$

$$\textbf{Fuente media: } 1 < \frac{B_a T_b}{L_p} < W_m \quad \textbf{(3.10)}$$

Donde W_m es el tamaño máximo de la ventana anunciada por el receptor medida en segmentos, L_p el tamaño de la ráfaga medida en bits, B_a es el ancho de banda de acceso medido en bits/sec, y T_b el tiempo de ensamblado de ráfagas.

Una fuente TCP de clase 'rápida' se caracteriza por un ancho de banda de acceso (B_a) lo suficientemente alto para permitir que todos los segmentos (o ACKs) de una ventana de congestión se agreguen en una sola ráfaga. Por el contrario, una fuente TCP de clase 'lenta' tiene un B_a tan bajo para admitir como máximo un solo segmento (o ACK) de la conexión

TCP dentro una ráfaga. Las fuentes de clase 'media' caen en un punto intermedio dentro de estas dos categorías, por lo cual en este caso varios segmentos de un flujo TCP estarán contenidos dentro de una ráfaga. Dicha clasificación permite que el número de segmentos sea evaluado con precisión pero el modelo desarrollado con estas suposiciones no es capaz de estudiar el desempeño de una fuente TCP de velocidad genérica; particular que ha sido solucionado en el modelo basado en la teoría de renovación para TCP presentado en el siguiente apartado. El *throughput* de una conexión TCP en el intervalo de tiempo [0, t] se define como $B_t = \frac{N_t}{t}$, donde N_t es el número de paquetes enviados en este período. Por lo tanto, el *throughput* en el estado estacionario de largo plazo de una fuente TCP se puede expresar como [111]:

$$B = \lim_{t \to \infty} B_t = \lim_{t \to \infty} \frac{N_t}{t} \qquad \textbf{(3.11)}$$

La evolución de la ventana de TCP se define como un proceso regenerativo (renovación) de Markov $\{W_i\}_i$, donde W denota el tamaño de la ventana. Un proceso regenerativo es un proceso estocástico con la propiedad de que existen puntos en el tiempo en los cuales probabilísticamente se reanuda el proceso. Denotemos por S_i el i-ésimo período de renovación en la evolución de la ventana de congestión, y por M_i la recompensa recibida en cada período regenerativo, igual al número de paquetes enviados en un ciclo. Utilizando el teorema de renovación, la recompensa media por unidad de tiempo es igual al *throughput* de la fuente TCP que se puede definir como [111]:

$$B = \frac{E[M]}{E[S]} \qquad \textbf{(3.12)}$$

Donde el *throughput* de un flujo TCP puede ser calculado en base a la relación del número de paquetes enviados durante un ciclo de crecimiento de la ventana con respecto a la duración media del ciclo; procedimiento que se puede utilizar para calcular el *throughput* de una fuente TCP de clase 'lenta' y 'rápida'.

3.4.1.1 Fuentes TCP de clase 'lenta'

Con una fuente TCP de clase 'lenta', cada ráfaga contiene como máximo un segmento. Así, el *throughput* puede ser modelado al igual que en el caso de las redes conmutadas de paquetes.

La Figura 3.13 muestra la evolución de la ventana de congestión en un ciclo de renovación denotado por S_i que consta de pérdidas tanto por triple ACK duplicado (TD) como por *timeout* (TO).

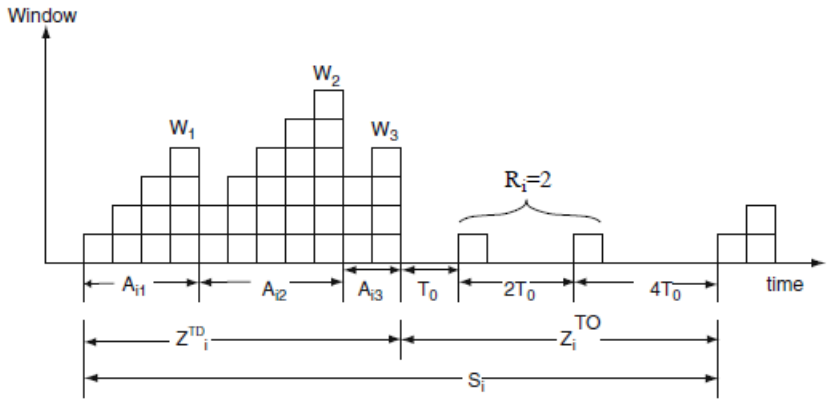


Figura 3.13 Evolución de la ventana de congestión para fuentes TCP de clase 'lenta' [111]

Esta evolución ignora las fases de *slow start* y *fast recovery* asumiendo que la mayor parte de los paquetes son enviados sólo durante la fase de *congestion avoidance*. Por lo tanto, el tamaño de la ventana W se incrementa en $1/W$ con la recepción de cada ACK, y consecuentemente, aumenta en 1 después del envío exitoso de cada ronda. El caso de ACKs retardados es despreciable, de manera que se envía un ACK por cada segmento recibido. Cada pérdida de ráfagas conduce a

la pérdida de un solo segmento, y dado que los eventos de pérdidas de ráfagas son independientes y distribuidos según Bernoulli, las pérdidas de segmentos también son estadísticamente independientes.

Denotemos por Z_i^{TO} la duración de una secuencia de *timeouts* que incluye el tiempo para la primera retransmisión a T_0, luego a $2T_0$, etc. hasta alcanzar $64T_0$; por Z_i^{TD} el intervalo de tiempo entre dos secuencias de *timeout* consecutivas, correspondiente a varios períodos de triple ACK duplicado (TDPs); y por n_i el número de TDPs en el intervalo Z_i^{TD} que equivale a tres en el ejemplo que se muestra en la Figura 3.13. De igual manera, denotemos por Y_{ij} el número de segmentos enviados durante el j-ésimo TDP del intervalo Z_i^{TD} de duración A_{ij}; por X_{ij} el número de rondas en el período; por R_i el número de segmentos enviados durante el período Z_i^{TO}; y por W_i el tamaño de la ventana al final del i-ésimo TDP. Con estas definiciones, el número total de segmentos enviados durante el i-ésimo período de renovación denotado por M_i y la duración de cada período de renovación S_i se pueden expresar como [111]:

$$\boldsymbol{M_i = \sum_{j=1}^{n_i} Y_{ij} + R_i} \tag{3.13}$$

$$\begin{aligned} S_i &= Z_i^{TD} + Z_i^{TO} \\ &= \sum_{j=1}^{n_i} \boldsymbol{A_{ij}} + \boldsymbol{Z_i^{TO}} \end{aligned} \tag{3.14}$$

Puesto que la ocurrencia de los períodos TDP y TO son independientes entre sí, el número medio de paquetes enviados durante el período S_i y la duración media del período pueden ser escritos como [111]:

$$\begin{aligned} \boldsymbol{E[M] = E\left[\sum_{j=1}^{n_i} Y_{ij}\right] + E[R]} \\ \boldsymbol{E[S] = E\left[\sum_{j=1}^{n_i} A_{ij}\right] + E[Z^{TO}]} \end{aligned} \tag{3.15}$$

Si se asume que $\{n_i\}_i$ es una secuencia de variables aleatorias independientes e idénticamente distribuidas o i.i.d. de $\{Y_{ij}\}$ y $\{A_{ij}\}$, entonces

$$E\left[\left(\sum_{j=1}^{n_i} Y_{ij}\right)\right] = E[n] * E[Y]$$
$$E\left[\left(\sum_{j=1}^{n_i} A_{ij}\right)\right] = E[n] * E[A] \quad \textbf{(3.16)}$$

Para derivar $E[n]$ nótese que, durante Z_i^{TD}, el tiempo entre dos secuencias de *timeout* consecutivas, hay n_i TDPs, donde cada uno de los primeros n_i -1 terminan en un TDP y el último TDP termina en un TO. De esto se deduce que en Z_i^{TD} hay un TO de n_i indicaciones de pérdidas. Por lo tanto, si denotamos por Q la probabilidad de que una indicación de pérdida en la que termina un TDP es un TO, se tiene que $Q = \frac{1}{E[n]}$. Consecuentemente, al sustituir las ecuaciones (3.15) y (3.16) en (3.12) se tiene que [111]:

$$B = \frac{E[Y] + Q * E[R]}{E[A] + Q * E[Z^{TO}]} \quad \textbf{(3.17)}$$

Para calcular el *throughput B* de una fuente TCP utilizando la ecuación (3.17), se debe evaluar el número exacto de segmentos enviados en cada uno de los períodos TDP y TO junto con su duración media, así como también la probabilidad Q. Considerando en primera instancia únicamente el TDP en un ciclo de renovación como se muestra en la Figura 3.14, donde la evolución de la ventana durante un i-ésimo TDP (A_i como se ilustra en la Figura 3.13) inicia inmediatamente después de una indicación de pérdida por TD. Al comienzo del i-ésimo TDP, el tamaño de la ventana está dado por $W_{i-1}/2$, donde W_{i-1} es el tamaño de la ventana al final del

i-ésimo - 1 TDP. En cada ronda, la ventana es incrementada en 1/*b* y el número de paquetes enviados por ronda se incrementa en uno cada *b* rondas.

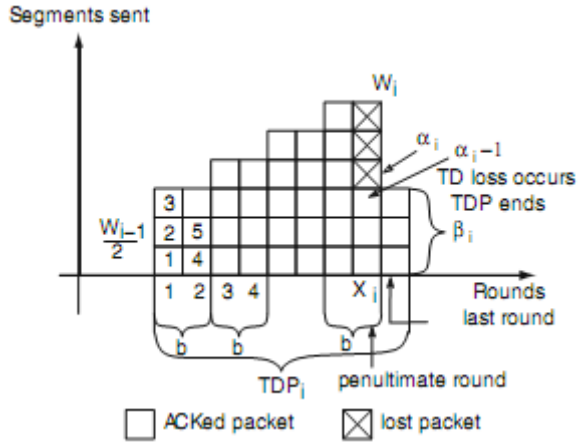


Figura 3.14 Paquetes enviados durante TDPi [111]

Sea X_i la ronda cuando ocurre la pérdida de paquetes por primera vez y α_i el primer paquete perdido en el TDP_i; después del paquete α_i, $W_i - \alpha_i + \beta_i$ segmentos más son enviados antes de que ocurra una indicación de pérdida por TD (y termine el TDP actual), donde β_i es igual a (α_i -1) segmentos transmitidos en la última ronda en respuesta a los ACKs recibidos que confirman los segmentos enviados correctamente en la penúltima ronda. De este modo, el número total de paquetes enviados en el i-ésimo TDP en $X_i + 1$ rondas, $Y_i = \alpha_i + W_i - 1$. Por lo tanto [111]:

$$\boldsymbol{E[Y] = E[\alpha] + E[W] - 1} \qquad \textbf{(3.18)}$$

Para derivar $E[\alpha]$, considere el proceso aleatorio $\{\alpha_i\}_i$, donde α_i corresponde al número de paquetes enviados en un TDP hasta e incluyendo el primer paquete que se perdió. En base a la hipótesis de que un paquete se pierde en una ronda independientemente de cualquier paquete perdido en otras rondas, $(\alpha_i)_i$ es una secuencia de variables aleatorias independientes e

idénticamente distribuidas (i.i.d.). Dado el modelo de pérdidas paquetes correlacionadas entre las transmisiones back-to-back dentro de una ronda, en el que si un paquete se pierde todos los paquetes transmitidos hasta el final de esa ronda también se pierden, la probabilidad de que α_i = k se puede expresar como la probabilidad de que k-1 paquetes sean reconocidos exitosamente antes de que ocurra una pérdida, y se puede calcular en base a una distribución geométrica.

$$P[\alpha = k] = (1 - p_b)^{k-1} p_b$$

$$E[\alpha] = \sum_{k=1}^{\infty} kP[\alpha = k] = \sum_{k=1}^{\infty} k(1 - p_b)^{k-1} p_b = p_b \sum_{k=1}^{\infty} k(1 - p_b)^{k-1}$$

$$E[\alpha] = p_b \sum_{k=1}^{\infty} -\left(\frac{d}{dp_b}(1 - p_b)^k\right) = -p_b \frac{d}{dp_b}\left(\sum_{k=1}^{\infty}(1 - p_b)^k\right)$$

$$E[\alpha] = -p_b \frac{d}{dp_b}\left(\sum_{k=0}^{\infty}(1 - p_b)^k - 1\right) = -p_b \frac{d}{dp_b}\left(\frac{1}{1 - (1 - p_b)} - 1\right)$$

$$E[\alpha] = -p_b \frac{d}{dp_b}\left(\frac{1}{p_b} - 1\right) = -p_b\left(\frac{-1}{p_b^2}\right)$$

$$\boldsymbol{E[\alpha] = \frac{1}{p_b}} \qquad \textbf{(3.19)}$$

De (3.18) y (3.19) se tiene que [85]:

$$\boldsymbol{E[Y] = \frac{1 - p_b}{p_b} + E[W]} \qquad \textbf{(3.20)}$$

Donde p_b es la probabilidad de pérdida de paquetes o probabilidad de pérdida de ráfagas, ya que cada ráfaga consiste únicamente de un solo paquete para fuentes de clase 'lenta'. De la duración del i-ésimo TDP, $A_i = \sum_{j=1}^{X_i+1} r_{ij}$, donde r_{ij} son variables aleatorias i.i.d que se asume son

independientes del tamaño de la ventana de congestión, y por lo tanto independientes del número de ronda, j; se puede deducir que [111]:

$$\boldsymbol{E[A]=(E[X]+1)E[r]} \tag{3.21}$$

que resulta del hecho de que un período TDP consiste de X_i + 1 rondas, cada una de duración RTT_{ij}, que denota la duración de la j-ésima ronda del i-ésimo TDP. Por consiguiente, se expresa por RTT = $E[r]$ como el valor medio del tiempo de ida y vuelta. Finalmente, para obtener $E[X]$ y $E[W]$, se debe considerar la evolución de la ventana de congestión en términos del número de rondas. Ya que en la fase de *congestion avoidance* el tamaño de la ventana se incrementa linealmente entre $W_{i-1}/2$ y W_i con pendiente $1/b$, el valor de la ventana al final de un TDP denotada por W_i se puede expresar como [111]:

$$\boldsymbol{W_i=\frac{W_{i-1}}{2}+\frac{X_i}{b}, \qquad i=1,2,...} \tag{3.22}$$

Entonces el número de segmentos enviados durante el i-ésimo TDP, Y_i puede ser escrito como [111]:

$$Y_i=\sum_{k=0}^{\frac{X_i}{b}-1}\left(\frac{W_{i-1}}{2}+k\right)b+\beta_i$$

$$Y_i=\frac{W_{i-1}}{2}b+\left(\frac{W_{i-1}}{2}+1\right)b+\left(\frac{W_{i-1}}{2}+2\right)b+...+\left(\frac{W_{i-1}}{2}+\left(\frac{X_i}{b}-1\right)\right)b+\beta_i$$

$$Y_i=\frac{W_{i-1}}{2}b\frac{X_i}{b}+b\left(1+2+...+\left(\frac{X_i}{b}-1\right)\right)b+\beta=\frac{W_{i-1}}{2}X_i+b\left(\sum_{i=1}^{\frac{X_i}{b}-1}i\right)+\beta_i$$

$$Y_i = \frac{W_{i-1}}{2} X_i + b \frac{\frac{X_i}{b}\left(\frac{X_i}{b} - 1\right)}{2} + \beta = \frac{X_i}{2}\left(W_{i-1} + \left(\frac{X_i}{b} - 1\right)\right) + \beta_i$$

Reemplazando $\frac{X_i}{b}$ utilizando (3.22), $Y_i = \frac{X_i}{2}\left(W_{i-1} + \left(W_i - \frac{W_{i-1}}{2} - 1\right)\right) + \beta_i$

$$Y_i = \frac{X_i}{2}\left(\frac{W_{i-1}}{2} + W_i - 1\right) + \beta_i$$

$$\boldsymbol{E[Y] = \frac{E[X]}{2}\left(\frac{E[W]}{2} + E[W] - 1\right) + E[\beta]} \qquad \textbf{(3.23)}$$

Donde β_i es el número de segmentos enviados en la última ronda, que se asume es uniformemente distribuido entre 1 y W_i. Por lo tanto $E[\beta] = E[W]/2$. Tomando la media de ambos lados de (3.22) y asumiendo que $\{X_i\}$ y $\{W_i\}$ son secuencias mutuamente independientes de variable aleatorias i.i.d., se tiene que [85]:

$$E[W] = \frac{E[W]}{2} + \frac{E[X]}{b}$$

$$\boldsymbol{E[W] = \frac{2}{b} E[X]} \qquad \textbf{(3.24)}$$

Utilizando las ecuaciones (3.20), (3.23), (3.24) y $E[\beta] = E[W]/2$ se obtiene que

$$\frac{1 - p_b}{p_b} + E[W] = \frac{E[X]}{2}\left(\frac{E[W]}{2} + E[W] - 1\right) + E[\beta]$$

$$\frac{1 - p_b}{p_b} + E[W] = \frac{b}{4} E[W]\left(\frac{3}{2} E[W] - 1\right) + \frac{E[W]}{2}$$

$$\frac{3b}{8} E[W]^2 - \left(\frac{b}{4} + \frac{1}{2}\right) E[W] - \left(\frac{1 - p_b}{p_b}\right) = 0$$

$$E[W] = \frac{\frac{2+b}{4} \pm \sqrt{\left(\frac{2+b}{4}\right)^2 + \frac{3b}{2}\left(\frac{1-p_b}{p_b}\right)}}{\frac{3b}{4}}$$

$$E[W] = \frac{2+b}{3b} + \sqrt{\frac{\left(\frac{2+b}{4}\right)^2}{\left(\frac{3b}{4}\right)^2} + \frac{\frac{3b}{2}\left(\frac{1-p_b}{p_b}\right)}{\frac{(3b)^2}{16}}}$$

$$\boldsymbol{E[W] = \frac{2+b}{3b} + \sqrt{\frac{8(1-p_b)}{3bp_b} + \left(\frac{2+b}{3b}\right)^2}} \quad \textbf{(3.25)}$$

La razón de descartar el signo "–" es que el término de la raíz cuadrada es mayor que el primer término, conduciendo a un valor negativo de la ventana de congestión. Si b es igual a 2, el valor de $(2+b)/3b$ es sólo 2/3, mientras que el primer término de la raíz cuadrada puede ser bastante grande si el valor de p_b es pequeño. Así, ignorando $(2+b)/3$ tanto dentro como fuera de la raíz cuadra, se puede aproximar el valor de la ecuación (3.25) como [111]:

$$\boldsymbol{E[W] \approx \sqrt{\frac{8}{3bp_b}} + o\left(\frac{1}{\sqrt{p_b}}\right)} \quad \textbf{(3.26)}$$

Usando la expresión de $E[W]$, se puede evaluar $E[X]$ de la ecuación (3.24). $E[X]$ se puede sustituir a su vez en la ecuación (3.21) para calcular $E[A]$. Si las pérdidas son únicamente pérdidas por TD y no existen *timeouts*, la expresión para el *throughput* llega a ser:

$$B = \frac{E[Y]}{E[A]} = \frac{\frac{1 - p_b}{p_b} + E[W]}{\left(\frac{b}{2}E[W] + 1\right)\text{RTT}} = \frac{\frac{1 - p_b}{p_b} + \frac{2+b}{3b} + \sqrt{\frac{8(1 - p_b)}{3bp_b} + \left(\frac{2+b}{3b}\right)^2}}{\text{RTT}\frac{b}{2}\left(\frac{2+b}{3b} + \sqrt{\frac{8(1 - p_b)}{3bp_b} + \left(\frac{2+b}{3b}\right)^2} + 1\right)}$$

$$\boldsymbol{B = \frac{\frac{1 - p_b}{p_b} + \frac{2+b}{3b} + \sqrt{\frac{8(1 - p_b)}{3bp_b} + \left(\frac{2+b}{3b}\right)^2}}{\text{RTT}\left(\frac{2+b}{6} + \sqrt{\frac{2b(1 - p_b)}{3p_b} + \left(\frac{2+b}{6}\right)^2} + 1\right)}} \quad \textbf{(3.27)}$$

que se puede simplificar como [111]:

$$\boldsymbol{B = \frac{1}{\text{RTT}}\sqrt{\frac{3}{2p_b}} + o\left(\frac{1}{\sqrt{p_b}}\right)} \quad \textbf{(3.28)}$$

Para incluir en el ciclo de renovación presentado en la Figura 3.13, el caso de los *timeouts* que ocurren cuando se pierden paquetes (o ACKs) y se reciben menos de tres ACKs duplicados, se deben determinar la probabilidad Q, el número esperado de paquetes enviados en los estados de *timeout*, $E[R]$, y el tiempo esperado utilizado en los estados de TO, $E[Z^{TO}]$.

Con el fin de derivar una expresión para Q, considere la ronda de paquetes donde ocurre una indicación de pérdida, referida como penúltima ronda (ver Figura 3.15). Sea W el tamaño de la ventana de congestión actual, por lo tanto $f_1 \ldots f_w$ paquetes son enviados en la penúltima ronda, de los cuales $f_1 \ldots f_k$ son reconocidos y f_{k+1} es el primer paquete perdido. Teniendo en cuenta que las pérdidas son correlacionadas, entonces todos los paquetes que siguen a f_{k+1} también se pierden. El reconocimiento de los paquetes $f_1 \ldots f_k$ conduce a la transmisión de $s_1 \ldots s_k$ nuevos paquetes en la siguiente ronda, referida como última ronda, que puede incluir otras pérdidas. Al igual que en el caso anterior, $s_1 \ldots s_m$ paquetes son reconocidos y $s_{m+1} \ldots s_k$ se pierden, por lo tanto, los m paquetes enviados exitosamente resultan en ACKs duplicados para el paquete f_k.

Se considera que estos ACKs no son retardados, por lo tanto el número de ACKs duplicados es igual al número de paquetes recibidos exitosamente en la última ronda. Si el número de estos ACKs es mayor que tres, entonces ocurre una indicación de TD; en caso contrario ocurre un TO, y en ambos casos termina el actual período entre pérdida, TDP.

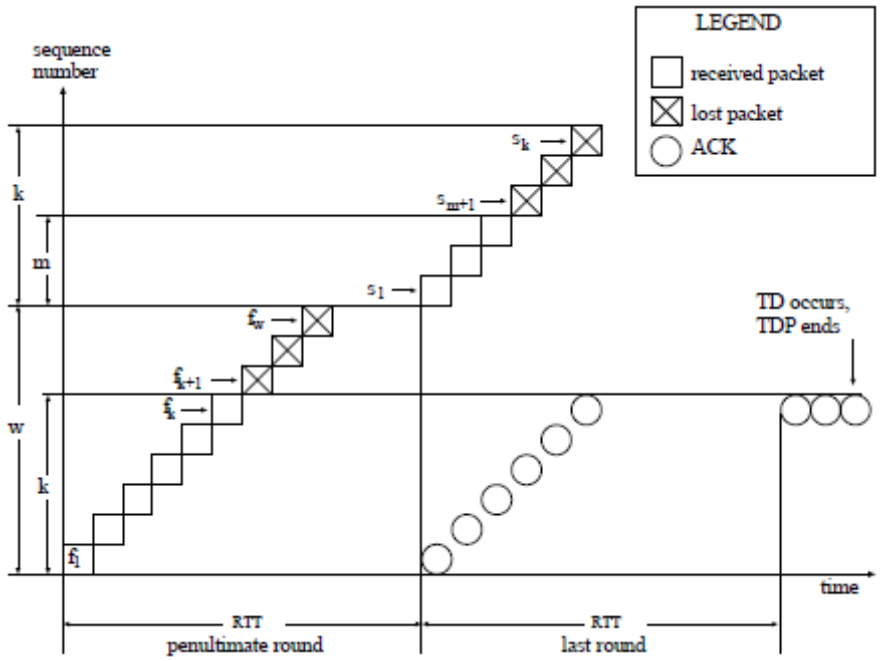


Figura 3.15 Transmisiones de paquetes y ACKs que preceden a una indicación de pérdida [111]

Sea $A(w,k)$ la probabilidad de que los primeros k paquetes sean reconocidos en una ronda de w paquetes, dado que hay una secuencia de una o más pérdidas en la ronda. Así [143]:

$$A(w,k) = P[primeros\ k\ paquetes\ no\ perdidos \mid pérdidas\ en\ W\ \ paquetes]$$

$$A(w,k) = \frac{P[(primeros\ k\ paquetes\ no\ perdidos)\ AND\ (pérdidas\ en\ W\ \ paquetes)]}{P[pérdidas\ en\ W\ \ paquetes]}$$

$$A(w,k) = \frac{P[(primeros\ k\ paquetes\ no\ perdidos)\ AND\ (paquete\ k+1\ perdido)]}{1 - P[no\ pérdidas\ en\ W\ \ paquetes]}$$

$$A(w,k) = \frac{(1-p)^k\, p}{1 - (1 - p)^w}$$

Por otro lado, se denota por $C(n,m)$ a la probabilidad de que m paquetes son reconocidos en secuencia en la última ronda, donde n paquetes fueron enviados, y los demás, si hay, se perdieron. Por lo tanto [143]:

$$C(n,m) = P[(\ primeros \quad m \ \ paquetes \quad no \ \ perdidos\ )\ AND\ (\ paquete \quad m+1 \ \ perdido\ )]$$

$$C(n,m) = \begin{cases} (1-p)^m\, p, & m < n \\ (1-p)^n, & m = n \end{cases}$$

Entonces, $\hat{Q}(w)$, la probabilidad de que una pérdida en una ventana de tamaño w sea un TO se puede deducir considerando los siguientes casos [74]:

1) *Si* $w \leq 3$

$\hat{Q}(w) = 1$, dado que se recibirán a lo mucho 2 ACKs duplicados

2) *Si* $w > 3$

a) Cuando k, el número de paquetes enviados exitosamente en la penúltima ronda es menor o igual que 2 ($k \leq 2$). En la última ronda se enviarán 2 o menos paquetes, por lo tanto la misma cantidad de ACKs serán recibidos, lo cual significa que no se pueden recibir 3 ACKs duplicados y por ende la indicación de pérdida es con un TO.

b) Cuando k, el número de paquetes enviados exitosamente en la penúltima ronda es mayor o igual que 3 ($k \geq 3$). Pero si en la última ronda únicamente 0, 1, o 2 de los k paquetes enviados son exitosos, entonces no se pueden enviar tres paquetes en la última ronda y por ende no se recibirán 3 ACKs duplicados, resultando en un TO. El término $A(w,k)C(n,m)$ es la probabilidad de que hay una pérdida en f_{k+1} en la penúltima ronda y una pérdida en s_{m+1} en la última ronda (ver Figura 3.15).

En base a lo anterior se tiene que [74][143]:

$P[$*paquete k en la ventana W se pierde*$] = A(w,k)$

$P[$*paquete 1, 2 o 3 de n paquetes se pierde*$] = C(n,0)+C(n,1)+C(n,2)$

$$\hat{Q}(w) = A(w,0) + A(w,1) + A(w,2) + A(w,3)[C(3,0) + C(3,1) + C(3,2)]$$
$$+ A(w,4)[C(4,0) + C(4,1) + C(4,2)] + ... + A(w,w)[C(w,0) + C(w,1) + C(w,2)]$$

$$\hat{Q}(w) = \begin{cases} 1 & Si\ w \le 3 \\ \sum_{k=0}^{2} A(w,k) + \sum_{k=3}^{w} A(w,k) \sum_{m=0}^{2} C(k,m) & En\ caso\ contrario \end{cases}$$

$$\sum_{k=0}^{2} A(w,k) = A(w,0) + A(w,1) + A(w,2)$$

$$\sum_{k=0}^{2} A(w,k) = \frac{p + p(1-p) + p(1-p)^2}{1-(1-p)^w}$$

$$A(w,3)\sum_{m=0}^{2} C(3,m) = \frac{p(1-p)^3}{1-(1-p)^w}\left[p + p(1-p) + p(1-p)^2\right]$$

Dado que $\sum_{m=0}^{2} C(3,m) = \sum_{m=0}^{2} C(4,m) = \sum_{m=0}^{2} C(k,m) = p + p(1-p) + p(1-p)^2$

$$\sum_{k=3}^{w}\left[A(w,k)\sum_{m=0}^{2} C(k,m)\right] = \frac{p + p(1-p) + p(1-p)^2}{1-(1-p)^w}\left[p(1-p)^3 + p(1-p)^4 + ... + p(1-p)^w\right]$$

$$\hat{Q}(w) = \frac{p + p(1-p) + p(1-p)^2}{1-(1-p)^w}\left[1 + p(1-p)^3 + p(1-p)^4 + ... + p(1-p)^w\right]$$

$$\hat{Q}(w) = \frac{2p - p^2 + p - 2p^2 + p^3}{1-(1-p)^w}\left[1 + p(1-p)^3\left[1 + (1-p) + (1-p)^2 + ... + (1-p)^{w-3}\right]\right]$$

Considerando que $1 + r + r^2 + r^3 + ... + r^{n-1} = \frac{1-r^n}{1-r}$

$$\hat{Q}(w) = \frac{3p - 3p^2 + p^3}{1-(1-p)^w}\left[1 + p(1-p)^3\left(\frac{1-(1-p)^{w-2}}{1-(1-p)}\right)\right]$$

$$\hat{Q}(w) = \frac{1-(1-p)^3}{1-(1-p)^w}\left[1 + (1-p)^3\left(1-(1-p)^{w-2}\right)\right]$$

$$\hat{Q}(w) = min\left(1, \frac{\left[1-(1-p)^3\right]\left[1+(1-p)^3\left(1-(1-p)^{w-2}\right)\right]}{1-(1-p)^w}\right) \quad \textbf{(3.29)}$$

Aplicando la regla de L'Hopital para encontrar $\lim_{p\to 0} \hat{Q}(w)$ se tiene que [74]:

$$\lim_{p\to 0} \hat{Q}(w) = \lim_{p\to 0} \frac{\frac{d}{dp}\left(1-(1-p)^3\right)\left[1+(1-p)^3\left(1-(1-p)^{w-2}\right)\right]}{\frac{d}{dp}\left(1-(1-p)^w\right)}$$

$$\lim_{p\to 0} \hat{Q}(w) = \lim_{p\to 0} \frac{3(1-p)^2\left[1+(1-p)^3\left(1-(1-p)^{w-2}\right)\right]+\left(1-(1-p)^3\right)\left[1+(1-p)^3\left(1-(1-p)^{w-2}\right)\right]'}{w(1-p)^{w-1}}$$

$$\lim_{p\to 0} \hat{Q}(w) = \frac{3}{w}$$

$$\hat{Q}(w) \approx min\left(1, \frac{3}{w}\right) \quad \textbf{(3.30)}$$

Recordando que:

Q = La probabilidad de que un período TDP termine en un *timeout*

$\hat{Q}(w)$ = La probabilidad de que un período TDP con un tamaño de ventana de congestión w termine en un *timeout*

Q se puede calcular de $\hat{Q}(w)$ usando la ley de probabilidad total [143] como:

$$Q = \hat{Q}(1) * P[W=1] + \hat{Q}(2) * P[W=2] + ...$$

$$Q = \sum_{w=1}^{\infty} \hat{Q}(w) P[W=w]$$

Dado que $E[W]=\sum_{w=1}^{\infty} wP[W=w]$, y la propiedad de que $E[f(W)]=\sum_{w=1}^{\infty} f(w)P[W=w]\cong f(E[W])$[47], se tiene que [143]:

$$\boldsymbol{Q = E[\hat{Q}(W)] \approx \hat{Q}(E[W])} \qquad \textbf{(3.31)}$$

Donde $E[W]$ está dado por (3.25).

Por otro lado, para calcular $E[R]$ se requiere la distribución del número de *timeouts* en una secuencia de TO. Una secuencia de estados de *timeout* siempre es seguida por una recepción exitosa de paquetes en el receptor (k *timeouts* ocurren cuando hay k-1 pérdidas consecutivas, seguidas por un paquete transmitido exitosamente). Por lo tanto, el número de *timeouts* en una secuencia de TO sigue una distribución geométrica con parámetro p_b de modo que [111]:

$$P[R=k]=p_b^{k-1}(1-p_b)$$

$$E[R]=\sum_{k=1}^{\infty} kP[R=k]=\sum_{k=1}^{\infty} kp_b^{k-1}(1-p_b)=(1-p_b)\sum_{k=1}^{\infty} kp_b^{k-1}$$

$$E[R]=(1-p_b)\sum_{k=1}^{\infty}\left(\frac{d}{dp_b}(p_b)^k\right)=(1-p_b)\frac{d}{dp_b}\left(\sum_{k=1}^{\infty} p_b{}^k\right)$$

$$E[R]=(1-p_b)\frac{d}{dp_b}\left(\sum_{k=0}^{\infty} p_b{}^k-1\right)=(1-p_b)\frac{d}{dp_b}\left(\frac{1}{1-p_b}-1\right)$$

$$E[R]=-(1-p_b)\frac{d}{dp_b}\left[(1-p_b)^{-1}-1\right]=-(1-p_b)\left[-1(1-p_b)^{-2}\right]$$

$$E[R]=(1-p_b)\left[\frac{1}{(1-p_b)^2}\right]$$

[47] Esta ecuación indica que $E[f(W)]$ se puede aproximar por el valor de la función f(x) en el punto $x=E[W]$.

$$E[R] = \frac{1}{1 - p_b} \tag{3.32}$$

Adicionalmente, utilizando el principio de *backoff exponencial* durante *timeouts* consecutivos, la duración de una secuencia de k *timeouts* está dada por:

Tabla 3.3 Valores de backoff del TO [143]

Número de timeouts	Valor de timeout Utilizado	Inicio al final de la secuencia de timeout
1er. timeout	TO	TO
2do. timeout	2TO	3TO
3er. timeout	4TO	7TO
4to. timeout	8TO	15TO
5to. timeout	16TO	31TO
6to. timeout	32TO	63TO
7to. timeout	64TO	(63+64)TO
8vo. timeout	64TO	(63+2*64)TO
9no. timeout	64TO	(63+3*64)TO
...		

$$H_k = \begin{cases} (2^k - 1)\text{RTO}, & k \le 6 \\ (63 + 64(k - 6))\text{RTO}, & k \ge 7 \end{cases}$$

Por lo tanto, la duración media del período TO está dada por [74]:

$$E[\mathrm{Z}^{\mathrm{TO}}] = \sum_{k=1}^{\infty} H_k P[R=k]$$

$$E[\mathrm{Z}^{\mathrm{TO}}] = \sum_{k=1}^{6} (2^k - 1)\mathrm{RTO} * P[R=k] + \sum_{k=7}^{\infty} (63 + 64(k-6))\mathrm{RTO} * P[R=k]$$

$$E[\mathrm{Z}^{\mathrm{TO}}] = \mathrm{RTO}\left[\sum_{k=1}^{6} (2^k - 1)p_b^{k-1}(1-p_b) + 63\sum_{k=7}^{\infty} p_b^{k-1}(1-p_b) + 64\sum_{k=7}^{\infty} (k-6)p_b^{k-1}(1-p_b)\right]$$

$$E[\mathrm{Z}^{\mathrm{TO}}] = \mathrm{RTO}(A + B + C)$$

$$A = (1-p_b)\sum_{k=1}^{6} (2^k - 1)p_b^{k-1} = (1-p_b)(1 + 3p_b + 7{p_b}^2 + 15{p_b}^3 + 31{p_b}^4 + 63{p_b}^5)$$

$$\text{Sea x} = 1 + 3p_b + 7{p_b}^2 + 15{p_b}^3 + 31{p_b}^4 + 63{p_b}^5$$

$$p_b x = p_b + 3{p_b}^2 + 7{p_b}^3 + 15{p_b}^4 + 31{p_b}^5 + 63{p_b}^6$$

$$A = (1-p_b)x = 1 + 2p_b + 4{p_b}^2 + 8{p_b}^3 + 16{p_b}^4 + 32{p_b}^5 - 63{p_b}^6$$

$$\Rightarrow A = \frac{1-(2p_b)^6}{1-2p_b} - 63{p_b}^6$$

$$B = 63(1-p_b)\sum_{k=7}^{\infty} p_b^{k-1} = 63(1-p_b)(p_b^6 + p_b^7 + p_b^8 + \ldots)$$

$$B = 63(1-p_b)\left[\sum_{k=0}^{\infty} p_b^k - (1 + p_b + p_b^2 + p_b^3 + p_b^4 + p_b^5)\right]$$

$$B = 63(1-p_b)\left[\left(\frac{1}{1-p_b}\right) - (1 + p_b + p_b^2 + p_b^3 + p_b^4 + p_b^5)\right]$$

$$B = 63(1-p_b)\left[\left(\frac{1}{1-p_b}\right) - \frac{(1-{p_b}^6)}{1-p_b}\right]$$

$$B = 63(1-p_b)\frac{{p_b}^6}{1-p_b} \Rightarrow B = 63{p_b}^6$$

$$C = 64(1 - p_b)\sum_{k=7}^{\infty}(k - 6)p_b^{k-1} = 64(1 - p_b)(p_b^{6} + 2p_b^{7} + 3p_b^{8} + ...)$$

$$Sea\ x = p_b^{6} + 2p_b^{7} + 3p_b^{8} + ...$$

$$p_b x = p_b^{7} + 2p_b^{8} + 3p_b^{9} + ...$$

$$C = 64(1 - p_b)x = 64(p_b^{6} + p_b^{7} + p_b^{8} + ...)$$

$$C = 64\left[\sum_{k=0}^{\infty} p_b^{k} - (1 + p_b + p_b^2 + p_b^3 + p_b^4 + p_b^5)\right]$$

$$C = 64\left[\frac{1}{1 - p_b} - (1 + p_b + p_b^2 + p_b^3 + p_b^4 + p_b^5)\right]$$

$$C = 64\left[\frac{1}{1 - p_b} - \frac{(1 - p_b^{6})}{1 - p_b}\right] \Rightarrow C = \frac{64p_b^{6}}{1 - p_b}$$

$$E[Z^{TO}] = RTO\left[\frac{1 - (2p_b)^{6}}{1 - 2p_b} - 63p_b^{6} + 63p_b^{6} + \frac{64p_b^{6}}{1 - p_b}\right]$$

$$E[Z^{TO}] = RTO\left[\frac{(1 - 2p_b)(1 + 2p_b + 4p_b^{2} + 8p_b^{3} + 16p_b^{4} + 32p_b^{5})}{1 - 2p_b} + \frac{64p_b^{6}}{1 - p_b}\right]$$

$$E[Z^{TO}] = RTO\left[\frac{(1 - p_b)(1 + 2p_b + 4p_b^{2} + 8p_b^{3} + 16p_b^{4} + 32p_b^{5}) + 64p_b^{6}}{1 - p_b}\right]$$

$$E[Z^{TO}] = RTO\,\frac{1 + p_b + 2p_b^{2} + 4p_b^{3} + 8p_b^{4} + 16p_b^{5} + 32p_b^{6}}{1 - p_b}$$

Donde $f(p) = 1 + p_b + 2p_b^{2} + 4p_b^{3} + 8p_b^{4} + 16p_b^{5} + 32p_b^{6}$. Por lo tanto,

$$\boldsymbol{E[Z^{TO}] = RTO\frac{f(p_b)}{1 - p_b}} \qquad \textbf{(3.33)}$$

Hasta ahora, no se ha considerado ninguna limitación en el tamaño de la ventana de congestión. Sin embargo, al inicio del establecimiento de una sesión TCP el receptor anuncia un tamaño máximo de *buffer* que determina un valor límite de la ventana de congestión W_{max}.

Como consecuencia, durante un período sin indicaciones de pérdidas, el tamaño de la ventana puede crecer hasta W_{max} pero no más allá de este valor.

Sea W_u el tamaño de la ventana sin restricción, su media derivada de la ecuación (3.25) está dada por [85]:

$$E[W_u] = \frac{2+b}{3b} + \sqrt{\frac{8(1-p_b)}{3bp_b} + \left(\frac{2+b}{3b}\right)^2} \qquad \textbf{(3.34)}$$

Si $E[W_u] < W_{max}$, se tiene la aproximación $E[W] \approx E[W_u]$, es decir que la ventana anunciada por el receptor tiene un efecto despreciable en la media del *throughput* de TCP y por tanto su valor que está dado por (3.25). Por otro lado, si $E[W_u] \geq W_{max}$, se puede aproximar como $E[W] \approx W_{max}$. En este caso, considere un intervalo Z^{TD} entre dos secuencias de *timeout* que consiste de una serie de períodos de TD como se muestra en la Figura 3.16. Al inicio del TDP la ventana crece linealmente hasta llegar a W_{max} por U_1 rondas, permanece constante por V_1 rondas, y luego ocurre una indicación de pérdida por TD, por lo tanto la ventana se reduce a $W_{max}/2$ y el proceso se repite.

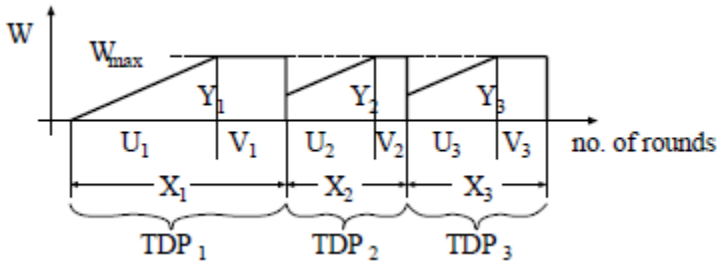


Figura 3.16 Fast retransmit con limitación de ventana [111]

$W_{max} = \frac{W_{max}}{2} + \frac{U_i}{b} \Rightarrow E[U] = \frac{b}{2} W_{max}$. Además, considerando la Figura 3.16 se puede calcular el número de paquetes enviados en el i-ésimo período TDP como [85]:

$$Y_i = \frac{U_i}{2}\left(\frac{W_{max}}{2} + W_{max}\right) + V_i W_{max}$$

$$E[Y] = \frac{3}{4} W_{max} E[U] + W_{max} E[V] = \frac{3b}{8} W_{max}^2 + W_{max} E[V]$$

De (3.20) se tiene que $E[Y] = \frac{1 - p_b}{p_b} + W_{max}$, por lo tanto

$$\frac{1 - p_b}{p_b} + W_{max} = \frac{3b}{8} W_{max}^2 + W_{max} E[V]$$

$$E[V] = \frac{1 - p_b}{p_b W_{max}} + 1 - \frac{3b}{8} W_{max}$$

Finalmente, ya que $X_i = U_i + V_i$ se tiene que [85]:

$$E[X] = E[U] + E[V]$$

$$E[X] = \frac{b}{2} W_{max} + \frac{1 - p_b}{p_b W_{max}} + 1 - \frac{3b}{8} W_{max}$$

$$\boldsymbol{E[X] = \frac{b}{8} W_{max} + \frac{1 - p_b}{p_b W_{max}} + 1} \qquad \textbf{(3.35)}$$

Por lo tanto, el *throughput* para una fuente TCP de clase 'lenta' se puede obtener sustituyendo todos los términos evaluados anteriormente en la ecuación (3.17), donde la tasa de transmisión resultante denominada B_{ku} incluye las pérdidas debido a *timeouts* y triple ACK duplicados, así como también un límite para el tamaño máximo de la ventana.

$$B^s = B_{ku}(W_{max}, \mathrm{RTT}, p, \mathrm{RTO}) = \tag{3.36}$$

$$\begin{cases} \dfrac{\dfrac{1-p_b}{p_b} + E[W_u] + \hat{Q}(E[W_u])\dfrac{1}{1-p_b}}{\mathrm{RTT}\left(\dfrac{E[W_u]}{2}+1\right) + \hat{Q}(E[W_u])\mathrm{RTO}\dfrac{f(p_b)}{1-p_b}} & \text{Para } E[W_u] < W_m \\ \dfrac{\dfrac{1-p_b}{p_b} + W_{max} + \hat{Q}(W_{max})\dfrac{1}{1-p_b}}{\mathrm{RTT}\left(\dfrac{W_{max}}{8} + \dfrac{1-p_b}{p_b W_{max}} + 2\right) + \hat{Q}(W_{max})\mathrm{RTO}\dfrac{f(p_b)}{1-p_b}} & \text{En caso contrario} \end{cases}$$

que puede ser escrita también como [25]:

$$B^s = B_{ku}(W_{max}, (1+\alpha)\mathrm{RTT}_0, p, (1+\alpha)\mathrm{RTT}_0) \tag{3.37}$$

Donde,

$$E[W_u] = 1 + \sqrt{\frac{8(1-p_b)}{3p_b} + 1}$$

$$\hat{Q}(u) = \min\left(1, \frac{3}{u}\right)$$

$$f(p) = 1 + p_b + 2p_b^2 + 4p_b^3 + 8p_b^4 + 16p_b^5 + 32p_b^6$$

3.4.1.2 Fuentes TCP de clase 'rápida'

Al igual que en el caso anterior, en este tipo de fuentes el crecimiento de la ventana de congestión se modela como una secuencia de rondas, donde según la definición de una fuente rápida, descrita en la ecuación (3.9), toda la ventana de segmentos en una ronda se agrega en una sola ráfaga. De esta manera, cuando se pierde una ráfaga se pierde una ventana TCP completa y el emisor pasa al estado de *timeout*. Dicha ronda se denomina ronda con pérdidas, mientras que la ronda en la cual la ráfaga alcanza con éxito el receptor se denomina ronda exitosa.

La Figura 3.17 muestra un ciclo de renovación durante la evolución de la ventana de congestión para fuentes TCP de clase 'rápida', que se puede describir simplemente como el conjunto de una secuencia de rondas exitosas cuando las ráfagas son entregadas correctamente, seguida por una secuencia de rondas con pérdidas cuando las ráfagas se pierden. El i-ésimo ciclo de renovación se denomina el i-ésimo período de *timeout* (TOP) mostrado como TOP_i en la Figura 3.17. Debido a que las pérdidas conducen al estado de *timeout*, las fases de *fast retransmit* y *fast recovery* no son activadas para el caso de fuentes TCP 'rápidas'. En cada pérdida de ráfagas, el emisor reinicia la ventana a un valor de 1 y empieza desde la fase de *slow start* en la cual trata de retransmitir todos los segmentos perdidos en la ronda anterior. Para cada ronda consecutiva con pérdidas, el emisor duplica el valor de RTO hasta alcanzar 64*RTO.

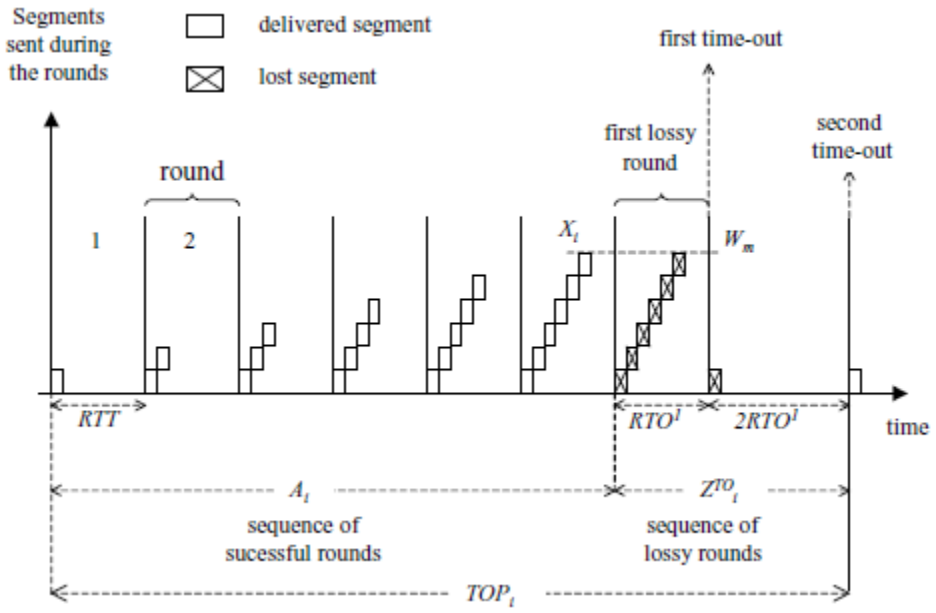


Figura 3.17 Evolución de la ventana de congestión para fuentes TCP de clase 'rápida' [111]

Durante el período TOP_i, sea Y_i el número de segmentos enviados en una secuencia de rondas exitosas con una duración total de A_i; H_i el número de segmentos enviados después de la primera pérdida por *timeout*; y Z_i^{TO} la duración de la secuencia de rondas con pérdidas.

Utilizando la expresión del *throughput* de la ecuación (3.12), para una fuente rápida se tiene que [111]:

$$B^f = \frac{E[Y] + E[H]}{E[A] + E[Z^{TO}]} \tag{3.38}$$

La duración media del período TO, $E[Z^{TO}]$, es igual a la dada en la ecuación (3.33). $E[R]$ evaluada en la ecuación (3.32) indica el número esperado de pérdidas por *timeout* en un ciclo de renovación. De este modo, el número de segmentos enviados después del primer *timeout* viene dado por el número de *timeouts* menos uno. Por lo tanto [111],

$$E[H] = E[R] - 1 = \frac{p_b}{1 - p_b} \tag{3.39}$$

Denotemos por X_i al número de rondas exitosas en el i-ésimo TOP, y ya que se asume un modelo de Bernoulli para la pérdida de ráfagas, la variable aleatoria X es distribuida geométricamente con parámetro p_b, y se obtiene como sigue:

$$P[X = k] = (1 - p_b)^k p_b$$

$$E[X] = \sum_{k=0}^{\infty} kP[X = k] = \sum_{k=0}^{\infty} k(1 - p_b)^k p_b = p_b \sum_{k=0}^{\infty} k(1 - p_b)^{k-1}(1 - p_b)$$

$$E[X] = p_b(1 - p_b)\sum_{k=0}^{\infty} k(1 - p_b)^{k-1}$$

$$E[X] = \frac{1 - p_b}{p_b} \tag{3.40}$$

Es así que la duración media de la secuencia de rondas exitosas, se puede determinar como [111]:

$$E[A]=E[X]\text{RTT}=\frac{1-p_b}{p_b}\text{RTT} \tag{3.41}$$

Para calcular $E[Y]$, se deben considerar los siguientes casos extremos: uno cuando la probabilidad de pérdida de ráfagas, BLP es alta y otro cuando es baja, en los que se evaluará los valores relevantes de este parámetro, es decir $E^h[Y]$ y $E^l[Y]$, respetivamente. Luego se propone una expresión general para $E[Y]$ y por ende para B^f.

1) E[Y] para una probabilidad de pérdidas alta: E^h

En este escenario, se supone que la probabilidad de pérdida de ráfagas es tan alta como para asumir que la limitación de la ventana puede ser despreciable dado que la saturación de *cwnd* es un evento bastante raro, y que el mecanismo de *slow start* no opera (*ssthresh=1*), por lo cual *cwnd* siempre se incrementa linealmente, y en consecuencia se puede expresar como [111]:

$$W_i = X_i + 1$$

$$E[W] = E[X] + 1 = \frac{1-p_b}{p_b} + 1$$

$$E[W]=\frac{1}{p_b} \tag{3.42}$$

$$Y_i=\sum_{k=1}^{Wi} k = W_i\left(\frac{W_i+1}{2}\right)=\frac{1}{2}(W_i^2+W_i) \tag{3.43}$$

Dado que $P[W=k]=(1-p_b)^{k-1}p_b$

$$E[W]=\sum_{k=1}^{\infty}kP[W=k]=\sum_{k=1}^{\infty}k(1-p_b)^{k-1}p_b=\frac{1}{p_b}$$

$$E[W^2]=p_b\sum_{k=1}^{\infty}k^2(1-p_b)^{k-1}$$

Puesto que $k^2(1-p_b)^{k-1}=k(1-p_b)^{k-1}-(1-p_b)\frac{d}{dp}\left(k(1-p_b)^{k-1}\right)$

$$E[W^2]=p_b\left(\sum_{k=1}^{\infty}k(1-p_b)^{k-1}-(1-p_b)\sum_{k=1}^{\infty}\frac{d}{dp}\left(k(1-p_b)^{k-1}\right)\right)$$

$$E[W^2]=p_b\sum_{k=1}^{\infty}k(1-p_b)^{k-1}-p_b(1-p_b)\frac{d}{dp}\left(\sum_{k=1}^{\infty}k(1-p_b)^{k-1}\right)$$

$$E[W^2]=p_b\left(\frac{1}{p_b^{\,2}}\right)-p_b(1-p_b)\frac{d}{dp}\left(\frac{1}{p_b^{\,2}}\right)=\frac{1}{p_b}+2(1-p_b)\left(\frac{1}{p_b^{\,2}}\right)$$

$$\boldsymbol{E[W^2]=\frac{2-p_b}{p_b^{\,2}}} \qquad \textbf{(3.44)}$$

Por lo tanto, $E[Y]$ cuando p_b es alta denotada por $E^h[Y]$ se puede evaluar utilizando las ecuaciones (3.42), (3.43) y (3.44) como [111]:

$$E^h[Y]=\frac{1}{2}\left(E[W^2]+E[W]\right)=\frac{1}{2}\left(\frac{2-p_b}{p_b^{\,2}}+\frac{1}{p_b}\right)$$

$$\boldsymbol{E^h[Y]=\frac{1}{p_b^{\,2}}} \qquad \textbf{(3.45)}$$

2) E[Y] para una probabilidad de pérdidas baja: E^l

En este caso, se supone que la probabilidad de pérdida de ráfagas es tan baja como para asumir que antes del primer evento de pérdida *cwnd* ya ha alcanzado su valor máximo $W_i=W_{max}$, y se

mantiene en ese valor durante un tiempo mucho mayor que el necesario para alcanzar este nivel máximo.

Consecuentemente, se puede despreciar el crecimiento de la ventana durante las fases de *slow start* y *congestion avoidance*; y asumir que al inicio del TDP, *cwnd* está en el estado W_{max}. Por lo tanto [111],

$$Y_i = W_{max}\, X_i,$$

$$E^l[Y] = W_{max} E[X] = W_{max}\left(\frac{1-p_b}{p_b}\right)$$

$$\boldsymbol{E^l[Y] \approx \frac{W_{max}}{p_b}} \qquad \textbf{(3.46)}$$

De lo anterior se puede observar que para una probabilidad de pérdidas alta $E^l[Y] > E^h[Y]$, mientras que cuando la probabilidad de pérdidas es baja $E^h[Y] > E^l[Y]$. Por estas razones, la expresión general para $E[Y]$ se puede aproximar como [111],

$$\boldsymbol{E[Y] = min(E^h[Y], E^l[Y]) = \begin{cases} E^h[Y] & Para\ p_b > \frac{1}{W_{max}} \\ E^l[Y] & En\ caso\ contrario \end{cases}} \qquad \textbf{(3.47)}$$

Por lo tanto, el *throughput* de una fuente TCP de clase 'rápida' está dado entonces por [25]:

$$\boldsymbol{B^f(W_{max}, RTT, p, RTO) =} \qquad \textbf{(3.48)}$$

$$\begin{cases} \dfrac{p_b^3 - p_b + 1}{(1+\alpha)RTT_0\left[p_b(1-p_b)^2 + p_b^2 f(p_b)\right]} & Para\ p_b > \dfrac{1}{W_m} \\ \dfrac{W_{max} - p_b W_{max} + p_b^2}{(1+\alpha)RTT_0\left[(1-p_b)^2 + p_b f(p_b)\right]} & En\ caso\ contrario \end{cases}$$

Aunque el *throughput* de una fuente de clase 'media' no se puede evaluar de una manera similar, es indudable que el *throughput* de dicha fuente se encuentra en un valor entre B^s y B^f como se puede apreciar en la Figura 3.18.

$$B^s < B^m < B^f \tag{3.49}$$

En un modelo de este tipo, el *throughput* de B^s y B^f son independientes del ancho de banda de acceso (B_a), mientras que B^m depende de este parámetro.

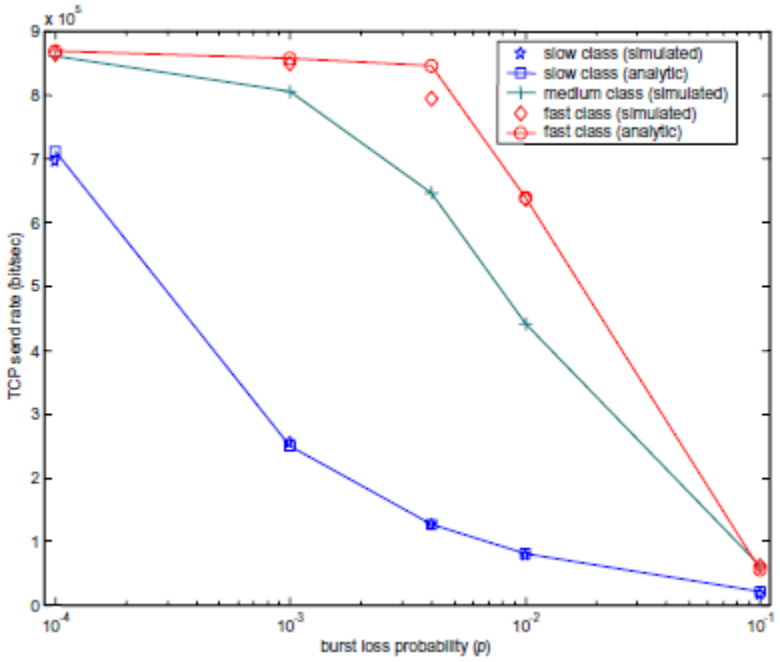


Figura 3.18 Tasa de transmisión de TCP vs. Probabilidad de pérdida de ráfagas (p) con T_b= 3ms, RTT0=600 ms, W_{max}=128, L=512 para diferentes fuentes de tráfico p.ej. 200 Mbps (clase rápida); 100 Mbps (clase media); 1 Mbps (clase lenta) [25]

Como se puede apreciar en la Figura 3.18, la diferencia en de desempeño entre las clases es debido sólo al número de segmentos por ráfaga, puesto que todas están sometidas a las mismas penalidades por retardo (igual valor de T_b), por lo que se puede concluir que mientras mayor es el número de segmentos agregados en una ráfaga, mayor es el la tasa de envío.

El beneficio de correlación y la penalidad por retardo se pueden cuantificar comparando el *throughput* de TCP en presencia y ausencia del ensamblador de ráfagas. La figura de mérito que es utilizada para la comparación es el tan llamado factor de ensamblado (F), definido como la relación entre B y NB, cuyo objetivo es medir la atenuación (si F<1) o amplificación (si F>1) de la tasa de transmisión debido a la presencia del ensamblador [25].

$$F = \frac{B}{NB} \tag{3.50}$$

Donde:

B= La tasa de transmisión de TCP (medida en segmentos por segundos) alcanzada en el escenario de red de la Figura 3.12.

NB= La tasa de transmisión de TCP (medida en segmentos por segundos) alcanzada en el escenario de red de la Figura 3.12 cuando no existe el par ensamblador/desensamblador de ráfagas.

En ausencia del ensamblador de ráfagas, el escenario de la Figura 3.12 es consistente con la hipótesis adoptada en [85]. Además, el RTT y RTO obtienen el mismo valor, igual para RTT_0 como se definió anteriormente en las ecuaciones (3.6) y (3.7). Por lo tanto, se puede utilizar la ecuación (3.37) para calcular NB como $B^s = B_{ku}(W_u, RTT_0, p_b, RTT_0)$. En caso de que el ensamblador esté presente, la ecuación (3.50) se puede derivar distinguiendo las diferentes clases de fuentes. Por lo tanto,

$$F^{s,m,f} = \frac{B^{s,m,f}}{NB} \tag{3.51}$$

Como se mencionó anteriormente, la penalidad por retardo reduce el *throughput* de una fuente TCP por un factor D_p que es proporcional a la relación del tiempo de ida y vuelta incluyendo el período de formación de las ráfagas con respecto al tiempo de ida y vuelta en ausencia del ensamblador. Por lo tanto [25],

$$D_p = \frac{B(\text{RTT en ausencia del ensamblado de ráfagas})}{B(\text{RTT incrementado debido al ensamblado de ráfagas})} \tag{3.52}$$

$$D_p = \frac{RTT}{RTT_0} = \frac{RTT_0 + T_b}{RTT_0} = 1 + \frac{T_b}{RTT_0} \tag{3.53}$$

Considerando (3.37), (3.48) y (3.52) se tiene que [25]:

$$B^s = \frac{B_0^s}{D_p} = \frac{NB}{D_p} \tag{3.54}$$

$$B^f = \frac{B_0^f}{D_p} \tag{3.55}$$

Donde B_0^s denota el *throughput* de TCP en ausencia del ensamblador de ráfagas y B_0^f el *throughput* de TCP cuando $\alpha = \frac{T_b}{\text{RTT}_0} = 0$. Pero B_0^f no se puede considerar como el *throughput* en ausencia del ensamblador de ráfagas ya que los supuestos utilizados para fuentes 'rápidas', donde una ráfaga contiene todos los segmentos de una ventana TCP, no se mantienen si se desprecia el proceso de ensamblado. Sustituyendo (3.54) y (3.55) en (3.51) se obtiene que:

$$F^{s,m,f} = \frac{B^{s,m,f}}{NB} = \frac{B_0^{s,m,f}}{D_p NB} = \frac{C_b^{s,m,f}}{D_p} \tag{3.56}$$

Por lo tanto, el beneficio de correlación para fuentes 'lentas' y 'rápidas' se define como [25]:

$$C_b^s = \frac{B_0^s}{NB} \tag{3.57}$$

$$C_b^f = \frac{B_0^f}{NB} \tag{3.58}$$

Al igual que el caso anterior expresado en la ecuación (3.49) con relación al *throughput*, el beneficio de correlación de C_b^s y C_b^f es independiente del ancho de banda de acceso (B_a), mientras que C_b^m depende de este parámetro; y se puede deducir que su valor se encuentra en un nivel intermedio entre C_b^s y C_b^f como se puede apreciar en la Figura 3.19.

$$C_b^s < C_b^m < C_b^f \tag{3.59}$$

De la ecuación (3.57), se puede deducir que $C_b^s = 1$. Por lo tanto, una fuente TCP de clase 'lenta' no tiene ningún beneficio de correlación, mientras que para fuentes TCP de tipo 'rápida' y 'media', este efecto positivo incrementa su valor conforme mayor número de segmentos son agregados en una ráfaga. En el primer caso el beneficio de correlación presenta su mayor contribución, mientras que en el segundo caso el beneficio de correlación se encuentra en un punto intermedio entre las de fuentes de tráfico 'lentas' y 'rápidas'. Como se puede observar en la Figura 3.19, el beneficio de correlación se maximiza para el valor de la probabilidad de pérdida igual al inverso de la ventana de congestión máxima $p_b = 1/W_{max}$ y se desvanece en los valores extremos de las probabilidades de pérdida [25].

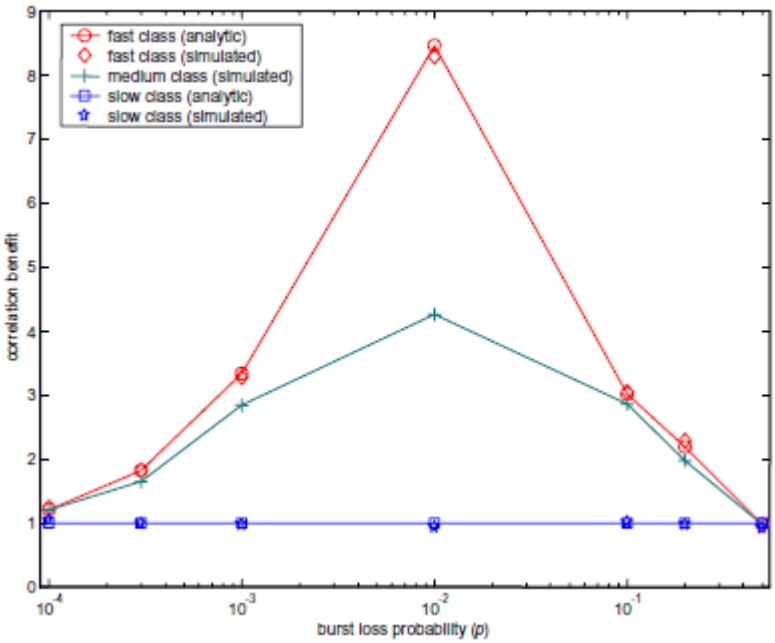


Figura 3.19 Beneficio de correlación vs. Probabilidad de pérdida de ráfagas (p) con T_b= 60 ms, RTT0=600 ms, W_{max}=128, L=512 para diferentes fuentes de tráfico p.ej. 10 Mbps (clase rápida); 3 Mbps (clase media); 50 Kbps (clase lenta) [25]

De acuerdo a los valores del ancho de banda de acceso, el período de ensamblado, y las pérdidas en la red, el beneficio de correlación puede o no superar las penalidades por retardo [25]. Por lo tanto existe un compromiso entre estos dos efectos opuestos.

3.4.2 Modelo basado en la teoría de renovación para las variantes basadas en pérdidas

El modelo presentado en el apartado anterior proporciona una expresión de forma cerrada para el *throughput* de TCP sólo para los casos de fuentes TCP 'lentas' y 'rápidas'. Como se mencionó anteriormente, tal clasificación no modela el *throughput* de TCP independiente del ancho de banda; por lo cual, en esta subsección se presenta el modelo basado en la teoría de renovación que se extiende a cualquier fuente genérica para las variantes populares de TCP basadas en

pérdidas TCP Reno, New Reno y SACK considerando el efecto del ensamblado de ráfagas y las pérdidas de ráfagas en el *throughput* [20,92,107].

En este apartado, el *throughput* de un único flujo TCP se considera que trabaja en el supuesto que la BLP, p_b, es insensible a la tasa de llegada de entrada a la red. Sea λ_a la tasa de llegada de los segmentos TCP en el nodo de ingreso que depende del ancho de banda de acceso y S el número de segmentos TCP en una ráfaga (valor medio para mantenerlo independiente del algoritmo de ensamblado de ráfagas). Las variables Y_i, X_i, A_i, H_i, y Z_i^{TO} tienen el mismo significado que en el apartado anterior. El tamaño de ráfaga S es al menos uno incluso cuando el tiempo de formación de ráfagas $T_b = 0$ y es a lo mucho igual al máximo tamaño de la ventana W_{max}. Bajo estas suposiciones, el tamaño de ráfaga está dado por [111]:

$$S = min(\lambda_a T_b + 1, W_{max}) \tag{3.60}$$

Además, se asume que el mecanismo de ACK retardado está presente de manera que la ventana se incrementa por cada b rondas reconocidas.

3.4.2.1 TCP SACK

Como se discutió anteriormente, TCP SACK utiliza un campo en los paquetes ACK que indica los segmentos perdidos en cada ráfaga ayudando a su recuperación más rápidamente. Existen dos tipos de técnicas utilizadas para la indicación de pérdidas en SACK: una es por triple ACKs duplicados y la otra es por *timeout*. Por consiguiente, el *throughput* de una fuente TCP SACK se puede expresar utilizando la ecuación (3.17).

La Figura 3.20 muestra el proceso de retransmisión de paquetes que se pierden en un TDP, en el cual todos los paquetes perdidos en una ronda se deben a la pérdida de una sola ráfaga y se retransmiten todos a la vez en la siguiente ronda junto con algunos nuevos segmentos. El número total de paquetes en una ronda se determina por el tamaño de la ventana en curso.

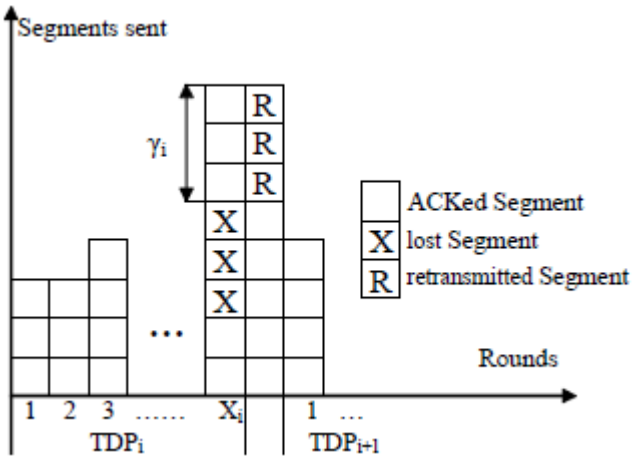


Figura 3.20 Retransmisión de TCP SACK sobre redes OBS [111]

Considere que la (β_i +1)-ésima ráfaga es la primera que se pierde en el TDP_i, cuyo primer segmento es el (α_i +1)-ésimo; además que X_i es la ronda donde ocurre la primera pérdida, y W_{Xi} es el tamaño de la ventana al final del TDP_i. Después de que se envía la ráfaga (β_i +1) y se pierde, γ_i segmentos adicionales serán enviados en la misma ronda. Una vez que se reciben tres ACKs duplicados, la fuente TCP retransmite inmediatamente los segmentos perdidos en la siguiente ronda (de acuerdo a la información contenida en los ACKs), junto con $W_{Xi} - S$ nuevos segmentos como se muestra en la Figura 3.20. Después de que la ráfaga perdida se retransmite exitosamente, el TDP_{i+1} empieza con un tamaño de ventana de transmisión de $W_{Xi}/2$.

En base a lo anterior, el número total de segmentos transmitidos en un TDP_i, es decir Y_i, se puede expresar como $Y_i = \alpha_i + \gamma_i + W_{Xi} - S$. Ya que $0 \leq \gamma_i \leq W_{max}$, se puede aproximar como $E[\gamma_i] \approx E[W_x]/2$ y así se obtiene que [111]:

$$E[Y] = E[\alpha] + \frac{3}{2}E[W_X] - S \qquad \textbf{(3.61)}$$

Puesto que las pérdidas de ráfagas se supone que son independientes entre sí, la variable aleatoria βi sigue una distribución geométrica. Además, $\alpha_i = S\beta_i$ de modo que $E[\alpha]$ está dada por [111]:

$$P[\beta = k] = (1 - p_b)^{k-1} p_b$$

Como $E[\alpha] = S\ E[\beta]$

$$E[\alpha] = SE[\beta] = S\sum_{k=1}^{\infty} kP[\beta = k] = \sum_{k=1}^{\infty} k(1 - p_b)^{k-1} p_b$$

$$\boldsymbol{E[\alpha] = \frac{S}{p_b}} \qquad \textbf{(3.62)}$$

Nótese que en una red de conmutación de paquetes $E[\alpha] = 1/p_b$ [85], es menor que $E[\alpha] = S/p_b$ para $S > 1$; esto debido a que el incremento en el número de segmentos que pueden ser enviados en un TDP antes de la primera pérdida, resulta en una ganancia DFL (*Delay First Loss*) en el *throughput* como se mencionó anteriormente [107].

Sustituyendo (3.62) en (3.61) se tiene que [111]:

$$\boldsymbol{E[Y] = \frac{3}{2} E[W_X] + \frac{1 - p_b}{p_b} S} \qquad \textbf{(3.63)}$$

Para evaluar el tamaño medio de la ventana al final del TDP, se consideran dos casos. El primero es cuando W_{max} es relativamente grande tal que W_{max} es raramente alcanzado y la

mayor parte del tiempo $W_X < W_{max}$. El segundo caso es cuando $W_X = W_{max}$ en la mayor parte de las rondas, lo cual sucede cuando la probabilidad de pérdida de ráfagas es muy baja.

1) Cuando $W_X < W_m$

Analizando la evolución de la ventana durante la fase de *congestion avoidance* similar a la explicada en la ecuación (3.22) se obtiene que [111]:

$$W_{X_i} = \frac{W_{X_{i-1}}}{2} + \frac{X_i}{b} \tag{3.64}$$

de la cual el valor medio de X se puede escribir como [111]:

$$E[X] = \frac{b}{2} E[W_X] \tag{3.65}$$

Ya que el número de segmentos Y_i enviados en el TDP_i se puede expresar también mediante la suma del número de segmentos enviados en todas las rondas X_i previas, y el de las S rondas adicionales, que es $W_{Xi} - S$, se tiene que [111]:

$$Y_i = \sum_{k=0}^{\frac{X_i}{b}-1} \left(\frac{W_{X_{i-1}}}{2} + k \right) b + W_{X_i} - S$$

$$Y_i = b \frac{W_{X_{i-1}}}{2} \frac{X_i}{b} + b \sum_{k=1}^{\frac{X_i}{b}-1} k + W_{X_i} - S$$

$$Y_i = X_i \frac{W_{X_{i-1}}}{2} + b \frac{\left(\frac{X_i}{b} - 1 \right) \frac{X_i}{b}}{2} + W_{X_i} - S$$

$$Y_i = \frac{X_i}{2}\left(W_{X_{i-1}} + \left(\frac{X_i}{b} - 1\right)\right) + W_{X_i} - S$$

Utilizando (3.64), se tiene que $Y_i = \frac{X_i}{2}\left(W_{X_{i-1}} + \left(W_{X_i} - \frac{W_{X_{i-1}}}{2} - 1\right)\right) + W_{X_i} - S$

$$Y_i = \frac{X_i}{2}\left(\frac{W_{X_{i-1}}}{2} + W_{X_i} - 1\right) + W_{X_i} - S$$

$$E[Y] = \frac{E[X]}{2}\left(\frac{E[W_X]}{2} + E[W_X] - 1\right) + E[W_X] - S$$

Por consiguiente, y utilizando la ecuación (3.65) se tiene que [107]:

$$E[Y] = \frac{b}{4}E[W_X]\left(\frac{3}{2}E[W_X] - 1\right) + E[W_X] - S$$

$$\boldsymbol{E[Y] = \frac{3bE[W_X]^2}{8} + \left(1 - \frac{b}{4}\right)E[W_X] - S} \qquad \textbf{(3.66)}$$

Combinando las ecuaciones (3.63) y (3.66) se encuentra que [107]:

$$\frac{3}{2}E[W_X] + \frac{1 - p_b}{p_b}S = \frac{3bE[W_X]^2}{8} + \left(1 - \frac{b}{4}\right)E[W_X] - S$$

$$\frac{3b}{8}E[W_X]^2 - \left(\frac{1}{2} + \frac{b}{4}\right)E[W_X] - \frac{S}{p_b} = 0$$

$$E[W_X] = \frac{\left(\frac{b+2}{4}\right) \pm \sqrt{\left(\frac{b+2}{4}\right)^2 + \frac{3b}{2}\frac{S}{p_b}}}{\frac{3b}{4}}$$

$$E[W_X] = \frac{b+2}{3b} + \sqrt{\frac{\left(\frac{b+2}{4}\right)^2 + \frac{3b}{2}\frac{S}{p_b}}{\left(\frac{3b}{4}\right)^2}}$$

$$\boldsymbol{E[W_X] = \frac{b+2}{3b} + \sqrt{\left(\frac{b+2}{3b}\right)^2 + \frac{8S}{3bp_b}}} \quad \textbf{(3.67)}$$

Sustituyendo (3.67) en (3.63) se obtiene la siguiente expresión para $E[Y]$

$$E[Y] = \frac{3}{2}\left[\frac{b+2}{3b} + \sqrt{\left(\frac{b+2}{3b}\right)^2 + \frac{8S}{3bp_b}}\right] + \frac{1-p_b}{p_b}S$$

$$E[Y] = \frac{1-p_b}{p_b}S + \frac{b+2}{2b} + \sqrt{\left(\frac{3}{2}\right)^2\left[\left(\frac{b+2}{3b}\right)^2 + \frac{8S}{3bp_b}\right]}$$

$$\boldsymbol{E[Y] = \frac{1-p_b}{p_b}S + \frac{b+2}{2b} + \sqrt{\left(\frac{b+2}{2b}\right)^2 + \frac{6S}{bp_b}}} \quad \textbf{(3.68)}$$

Además, reemplazando (3.67) en (3.65) se obtiene que [107]:

$$E[X] = \frac{b}{2}\left[\frac{b+2}{3b} + \sqrt{\left(\frac{b+2}{3b}\right)^2 + \frac{8S}{3bp_b}}\right]$$

$$E[X] = \frac{b+2}{6} + \sqrt{\left(\frac{b}{2}\right)^2\left[\left(\frac{b+2}{3b}\right)^2 + \frac{8S}{3bp_b}\right]}$$

$$\boldsymbol{E[X] = \frac{b+2}{6} + \sqrt{\left(\frac{b+2}{6}\right)^2 + \frac{2bS}{3p_b}}} \quad \textbf{(3.69)}$$

Por otro lado, $E[A]$ se puede calcular en base a la ecuación (3.21), por tanto [107]:

$$E[A] = \mathrm{RTT}(E[X] + 1)$$

$$\boldsymbol{E[A] = \mathrm{RTT}\left(\frac{b+8}{6} + \sqrt{\left(\frac{b+2}{6}\right)^2 + \frac{2bS}{3p_b}}\right)} \quad \textbf{(3.70)}$$

Nótese que para una tasa de pequeña de pérdidas p_b, $E[A]$ se puede aproximar como [107]:

$$\boldsymbol{E[A] \approx RTT\sqrt{\frac{2bS}{3p_b}}} \quad \textbf{(3.71)}$$

2) Cuando Wx = Wm

La ecuación (3.64) no se puede utilizar para estimar el número de rondas en un TDP. Como la ventana tiene un tamaño de W_{max} la mayor parte del tiempo, $E[Y]$ es evaluado estableciendo $E[W_x] = W_{max}$ en la ecuación (3.63) para llegar a [111]:

$$\boldsymbol{E[Y] = \frac{3}{2}W_{max} + \frac{1-p_b}{p_b}S} \quad \textbf{(3.72)}$$

Ya que el número de paquetes enviados exitosamente antes de que ocurra un evento de pérdida por TD es S/p_b, obtenido de la ecuación (3.62), y que durante cada TDP la ventana se incrementa linealmente desde $W_{max}/2$ a W_{max}, para (W_{max}- $W_{max}/2$) rondas y luego permanece constante en W_{max}, para (X_i-(W_{max}- $W_{max}/2$)) rondas, se tiene la siguiente relación [107]:

$$\frac{\left(\frac{W_{max}}{2}+W_{max}\right)\frac{W_{max}}{2}}{2}+W_{max}\left(X_i-\left(W_{max}-\frac{W_{max}}{2}\right)\right)-\gamma_i=\frac{S}{p_b}$$

$$\frac{3}{8}W_{max}^2+W_{max}\left(X_i-\frac{W_{max}}{2}\right)-\frac{W_{max}}{2}=\frac{S}{p_b}$$

$$W_{max}X_i-\frac{W_{max}^2}{8}-\frac{W_{max}}{2}=\frac{S}{p_b}$$

$$\boldsymbol{E[X]=\frac{S}{p_bW_{max}}+\frac{W_{max}}{8}+\frac{1}{2}} \qquad \textbf{(3.73)}$$

A partir de $E[X]$, se puede calcular $E[A]$ usando la ecuación (3.21).

$$E[A]=(E[X]+1)E[r]$$

$$\boldsymbol{E[A]=\mathrm{RTT}\left(\frac{S}{p_bW_{max}}+\frac{W_{max}}{8}+\frac{3}{2}\right)} \qquad \textbf{(3.74)}$$

Ya que en $W_x = W_{max}$, también podría haber pérdidas por TO que se pueden modelar de manera similar que en el apartado anterior. $E[H]$ se puede evaluar mediante la ecuación (3.39) y $E[\mathrm{ZTO}]$ utilizando la ecuación (3.33).

Por otra parte, se debe notar que en una red OBS los eventos de TO pueden ocurrir con un patrón bastante diferente que el de una red IP de conmutación de paquetes, porque en este último caso, la pérdida de paquetes en los enrutadores IP es frecuentemente por desbordamiento de *buffer* [85] y por tanto una pérdida de paquetes es a menudo seguida por la pérdida de todos los paquetes subsecuentes enviados durante la misma ronda. En consecuencia, la probabilidad de que un evento de TO ocurra se puede aproximar con la probabilidad de que menos de tres paquetes sean entregados exitosamente en la última ronda.

Por otro lado, si T_b es lo suficientemente grande de manera que cada ráfaga contiene al menos tres segmentos (es decir, S ≥ 3), un evento de TO ocurre si y sólo si todas las ráfagas en la última ronda se pierden.

Además, ya que no existe *buffer* en ningún nodo de *core* OBS, la correlación entre las pérdidas de ráfagas es pequeña. En este sentido, después de que una ráfaga se pierde, es difícil determinar cuántas rondas adicionales habrá antes de que suceda el evento de TO ya que puede ocurrir pérdida(s) de ráfagas subsecuentes durante rondas adicionales.

En lugar de asumir que una vez que un paquete se pierde, todos los paquetes subsecuentes en la misma ronda también se pierden como en [85], en estos modelos se supone que no hay pérdidas de ráfagas posteriores en rondas adicionales después de una pérdida por TD. En otras palabras, un evento de TO ocurre sólo cuando se pierden todas las ráfagas en la última ronda (X_i) de un TDP, donde la probabilidad de que ocurra tal evento está dada por [107]:

$$P(W_X) = p_b^{\frac{W_X}{s}-1} \tag{3.75}$$

Ya que un TDP estará seguido por un TOP con probabilidad $P(Wx)$ o por otro TDP con probabilidad 1- $P(Wx)$, y un TOP siempre estará seguido por un TDP, la relación esperada de la probabilidad de un *timeout* con respecto a la probabilidad de una pérdida por triple ACK duplicado es [107]:

$$Q(E[W_X]) = E[P(W_X)] = E[p_b^{\frac{W_X}{s}-1}] \cong p_b^{\frac{E[W_X]}{s}-1} \tag{3.76}$$

Sustituyendo las expresiones de $E[Y]$, $E[A]$, $E[H]$, $Q = Q(E[W_x])$ y $E[Z^{TO}]$ en la ecuación (3.17) se obtiene que el *throughput* para una fuente TCP SACK es [107]:

$B(p,T_b) =$ **(3.77)**

$$\begin{cases} \dfrac{\frac{3}{2}E[W_x] + \frac{1-p_b}{p_b}S + Q(E[W_x])\frac{p_b}{1-p_b}}{\mathrm{RTT}\left(\frac{b}{2}E[W_x]+1\right) + Q(E[W_x])\mathrm{RTO}\frac{f(p_b)}{1-p_b}} & Para\ W_x < W_{max} \\ \dfrac{\frac{3}{2}W_{max} + \frac{1-p_b}{p_b}S + Q(W_{max})\frac{p_b}{1-p_b}}{\mathrm{RTT}\left(\frac{S}{p_b W_{max}} + \frac{W_{max}}{8} + \frac{3}{2}\right) + Q(W_{max})\mathrm{RTO}\frac{f(p_b)}{1-p_b}} & Para\ W_x = W_{max} \end{cases}$$

Si p_b es pequeña, la probabilidad Q del evento de TO será muy pequeña, pudiéndose ignorar el segundo término tanto en el numerador como en el denominador de (3.17), con lo cual (3.77) se puede simplificar a [107]:

$B(p, T_b) =$ **(3.78)**

$$\begin{cases} \dfrac{\frac{S}{p_b}}{\mathrm{RTT}\left(\sqrt{\frac{2bS}{3p_b}}+1\right)} + o\left(\frac{1}{\sqrt{p_b}}\right) = \dfrac{1}{\mathrm{RTT}_0 + 2T_b}\sqrt{\dfrac{3S}{2bp_b}} + o\left(\dfrac{1}{\sqrt{p_b}}\right) & Para W_x < W_{max} \\ \dfrac{W_{max}}{\mathrm{RTT}} & Para W_x = W_{max} \end{cases}$$

Donde RTT_0 es el valor del tiempo de ida y vuelta de TCP sin ensamblado de ráfagas, y $RTT = RTT_0 + 2T_b$ porque tanto los segmentos de datos como los ACKs experimentan un retardo de ensamblado igual a T_b. Cabe indicar además que, en el tiempo de vida de una conexión TCP, los dos casos discutidos anteriormente pueden ocurrir, y por tanto, el *throughput* esperado estará entre los resultados de las ecuaciones (3.77 para $W_m < W_{max}$) y (3.77 para $W_m = W_{max}$).

3) Tiempo óptimo de ensamblado

En esta subsección, se considera un caso práctico en una red TCP sobre OBS donde el número máximo de segmentos contenidos en una ráfaga es $1 < S = T_b\lambda+1 < W_{max}$, y por ende, mientras mayor sea T_b, mayor será S; en cuyo caso, tanto el numerador como el denominador de la ecuación (3.78 para $W_x < W_{max}$) contienen T_b, que en el primer caso representa la ganancia DFL (en la forma de $\sqrt{S}$) en el *throughput* de TCP que incrementa con el tiempo de ensamblado; mientras que en el segundo caso, representa la penalidad por retardo producto del proceso ensamblado de ráfagas que experimenta el *throughput* de TCP y que disminuye con el tiempo de ensamblado. Ya que están presentes estos dos parámetros, puede existir un tiempo óptimo de ensamblado que maximice el *throughput* de TCP.

Se debe notar que para las ecuaciones (3.78 para $W_x < W_{max}$) y (3.78 para $W_x = W_{max}$), incluso aunque el numerador varía sub-linealmente con T_b y el denominador varía linealmente con T_b, existe un T_b óptimo cuando el factor constante RTT_0 en el denominador es grande. Más específicamente, para el caso $W_x < W_{max}$, calculando la raíz de la siguiente ecuación, se obtiene el tamaño óptimo de ráfaga $S = T_b\lambda+1$ que maximiza B de la ecuación (3.78 para $W_x < W_{max}$) como se indica a continuación [107]:

$$max\{B(T_b)\} \approx max\left\{\frac{\sqrt{3\lambda T_b}}{RTT_0 + 2T_b}\right\} \approx max\left\{\frac{\frac{\sqrt{3\lambda T_b}}{\sqrt{T_b}}}{\frac{RTT_0}{\sqrt{T_b}} + \frac{2T_b}{\sqrt{T_b}}}\right\} \approx max\left\{\frac{\sqrt{3\lambda}}{\frac{RTT_0}{\sqrt{T_b}} + \frac{2T_b}{\sqrt{T_b}}}\right\}$$

$$\approx max\left\{\frac{1}{\frac{RTT_0}{\sqrt{T_b}} + 2\sqrt{T_b}}\right\} = min\left\{\frac{RTT_0}{\sqrt{T_b}} + 2\sqrt{T_b}\right\} \Rightarrow \frac{d}{dT_b}\left(\frac{RTT_0}{\sqrt{T_b}} + 2\sqrt{T_b}\right) = 0$$

$$-\frac{RTT_0}{2\sqrt{T_b^3}} + \frac{1}{\sqrt{T_b}} = 0 \Rightarrow \frac{1}{\sqrt{T_b}} = \frac{RTT_0}{2T_b\sqrt{T_b}}$$

$$T_{Tb}^{opt} = \frac{RTT_0}{2} \tag{3.79}$$

Reemplazando la ecuación (3.79) en la ecuación (3.78 para $W_x < W_{max}$) se obtiene que el *throughput* óptimo es [107]:

$$B_m(p_b) \approx \frac{1}{2RTT_0}\sqrt{\frac{3\lambda RTT_0}{4bp_b}}$$

$$B_m(p_b) \approx \frac{1}{4RTT_0}\sqrt{\frac{3\lambda\, RTT_0}{bp_b}} \tag{3.80}$$

Para el caso $W_x = W_{max}$, el T_b óptimo se obtiene maximizando B de la ecuación (3.77 para $W_x = W_{max}$) como sigue [107]:

$$\max\{B(T_b)\} \approx \max\left\{\frac{\frac{S}{p_b}}{(RTT_0 + 2T_b)\left(\frac{S}{p_b W_{max}} + \frac{W_{max}}{8} + \frac{3}{2}\right)}\right\}$$

$$\approx \max\left\{\frac{1}{\frac{RTT_0}{W_{max}} + \frac{p_b W_{max} RTT_0}{8S} + \frac{3p_b}{2S}RTT_0 + \frac{2T_b}{W_{max}} + \frac{p_b T_b W_{max}}{4S} + \frac{3T_b p_b}{S}}\right\}$$

$$\approx \max\left\{\frac{1}{\frac{RTT_0}{W_{max}} + \frac{p_b W_{max} RTT_0}{8T_b\lambda} + \frac{3p_b RTT_0}{2T_b\lambda} + \frac{2T_b}{W_{max}} + \frac{p_b W_{max}}{4\lambda} + \frac{3p_b}{\lambda}}\right\}$$

$$\approx max\left\{\frac{1}{\frac{2}{W_{max}}T_b+\frac{\left(\frac{p_bW_{max}RTT_0}{8\lambda}+\frac{3p_bRTT_0}{2\lambda}\right)}{T_b}+\left(\frac{RTT_0}{W_{max}}+\frac{p_bW_{max}+12p_b}{4\lambda}\right)}\right\}$$

$$=min\left\{\frac{2}{W_{max}}T_b+\frac{\left(\frac{p_bW_{max}RTT_0}{8\lambda}+\frac{3p_bRTT_0}{2\lambda}\right)}{T_b}+\left(\frac{RTT_0}{W_{max}}+\frac{p_bW_{max}+12p_b}{4\lambda}\right)\right\}$$

$$=\frac{d}{dT_b}\left\{\frac{2}{W_{max}}T_b+\frac{\left(\frac{p_bW_{max}RTT_0}{8\lambda}+\frac{3p_bRTT_0}{2\lambda}\right)}{T_b}+\left(\frac{RTT_0}{W_{max}}+\frac{p_bW_{max}+12p_b}{4\lambda}\right)\right\}=0$$

$$\Rightarrow\frac{2}{W_m}-\frac{\left(\frac{p_bW_{max}RTT_0}{8\lambda}+\frac{3p_bRTT_0}{2\lambda}\right)}{T_b{}^2}=0\Rightarrow T_b{}^2=\frac{W_{max}}{2}\left(\frac{p_bW_{max}RTT_0}{8\lambda}+\frac{3p_bRTT_0}{2\lambda}\right)$$

$$\boldsymbol{T_b^{opt}=\sqrt{\frac{p_bW_{max}}{2\lambda}RTT_0\left(\frac{W_{max}}{8}+\frac{3}{2}\right)}} \qquad \textbf{(3.81)}$$

Reemplazando la ecuación (3.81) en la (3.77 para $W_x = W_{max}$) se obtiene que el *throughput* óptimo es [107]:

$$B_m(p_b)\approx\frac{1}{\frac{2}{W_{max}}T_b+\frac{\left(\frac{p_bW_{max}RTT_0}{8\lambda}+\frac{3p_bRTT_0}{2\lambda}\right)}{T_b}+\left(\frac{RTT_0}{W_{max}}+\frac{p_bW_{max}+12p_b}{4\lambda}\right)}$$

$$B_m(p_b) \approx \frac{1}{\frac{p_b(W_{max}+12)}{4\lambda} + \frac{RTT_0}{W_{max}} + \frac{2T_b^{\,2} + \left(\frac{p_b W_{max}^2 RTT_0}{8\lambda} + \frac{3p_b W_{max} RTT_0}{2\lambda}\right)}{W_{max} T_b}}$$

$$B_m(p_b) \approx \frac{1}{\frac{p_b(W_{max}+12)}{4\lambda} + \frac{RTT_0}{W_{max}} + \frac{\frac{p_b W_{max}}{\lambda} RTT_0\left(\frac{W_{max}}{8}+\frac{3}{2}\right) + \frac{p_b W_{max}}{\lambda} RTT_0\left(\frac{W_{max}}{8}+\frac{3}{2}\right)}{W_{max}\sqrt{\frac{p_b W_{max}}{2\lambda} RTT_0\left(\frac{W_{max}}{8}+\frac{3}{2}\right)}}}$$

$$B_m(p_b) \approx \frac{1}{\frac{p_b(W_{max}+12)}{4\lambda} + \frac{RTT_0}{W_{max}} + \frac{2\frac{p_b W_{max}}{\lambda} RTT_0\left(\frac{W_{max}}{8}+\frac{3}{2}\right)}{W_{max}\sqrt{\frac{p_b W_{max}}{2\lambda} RTT_0\left(\frac{W_{max}}{8}+\frac{3}{2}\right)}}}$$

$$B_m(p_b) \approx \frac{1}{\frac{p_b(W_{max}+12)}{4\lambda} + \frac{RTT_0}{W_{max}} + \sqrt{\frac{4\frac{p_b W_{max}}{\lambda} RTT_0\left(\frac{W_{max}}{8}+\frac{3}{2}\right)}{\frac{W_{max}^2}{2}}}}$$

$$\boldsymbol{B_m(p_b) \approx \frac{1}{\frac{p_b(W_{max}+12)}{4\lambda} + \frac{RTT_0}{W_{max}} + \sqrt{\frac{8p_b RTT_0\left(\frac{W_{max}}{8}+\frac{3}{2}\right)}{\lambda W_{max}}}}} \quad \textbf{(3.82)}$$

Nótese que si el ancho de banda de acceso es tan bajo o tan alto que S=1 o S=W_{max}, la ganancia DFL $\sqrt{S}$ es fija (igual a 1 o $\sqrt{W_{max}}$), pero la penalidad por retardo (en RTTs) se incrementa con T_b. Por lo tanto, el *throughput* decrementa monótonamente con T_b.

3.4.2.2 TCP New Reno

A diferencia de SACK, New Reno retransmite sólo un segmento perdido de una ráfaga perdida después de recibir cada ACK parcial. Durante la fase de *fast retransmit* donde el número de rondas es igual al número de paquetes perdidos (que es igual al tamaño de la ráfaga *S*), con sólo un ACK parcial, el número de paquetes pendientes sin acuse de recibo pronto excederá la ventana de transmisión y pocos paquetes nuevos se pueden transmitir. Consecuentemente, se puede ignorar el número de nuevos paquetes enviados en la fase de retransmisión para el análisis, como se muestra en la Figura 3.21.

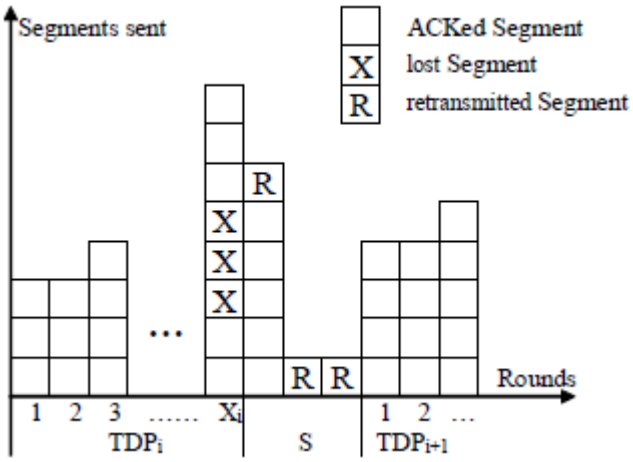


Figura 3.21 Retransmisión de TCP New Reno sobre redes OBS [111]

El número total de paquetes transmitidos en un TDP es igual que en el caso de SACK, pero el TDP en New Reno contiene *S* rondas adicionales. Por ende, el *throughput* de TCP New Reno para un valor pequeño de p_b y cuando $W_x < W_{max}$ está dado por [111]:

$$B = \frac{\frac{S}{p_b}}{\mathrm{RTT}\left(\sqrt{\frac{2bS}{3p_b}} + S\right)} + o\left(\frac{1}{\sqrt{p_b}}\right) \qquad \textbf{(3.83)}$$

que es menor que el *throughput* de SACK expresado en (3.78 para $W_x < W_{max}$). Sin embargo, si el tamaño promedio de la ventana es mucho mayor que la longitud de la ráfaga, es decir, $E[W_x] >> S$ o equivalentemente $\sqrt{\frac{2bS}{3p_b}} >> S$, S puede ser ignorada en el denominador y (3.83) es igual que la ecuación (3.78 para $W_x < W_{max}$); en otras palabras, New Reno y SACK tendrían el mismo *throughput*. Por otra parte, si $Wx = W_{max}$, $\sqrt{\frac{2bS}{3p_b}} >> S$ para una p_b pequeña, el *throughput* de New Reno es el mismo que el indicado en la ecuación (3.78 para $W_x = W_{max}$) [107].

3.4.2.3 TCP Reno

En TCP Reno, el mecanismo de retransmisión es diferente que en SACK y New Reno, por ende, la fase de *fast retransmit/recovery* podría ser distinta para diferentes longitudes *S* de ráfagas. Más específicamente, se pueden discutir a continuación dos patrones de pérdidas en un TDP, es decir, pérdidas sólo por TD y pérdidas mixtas por TD y TO, respectivamente.

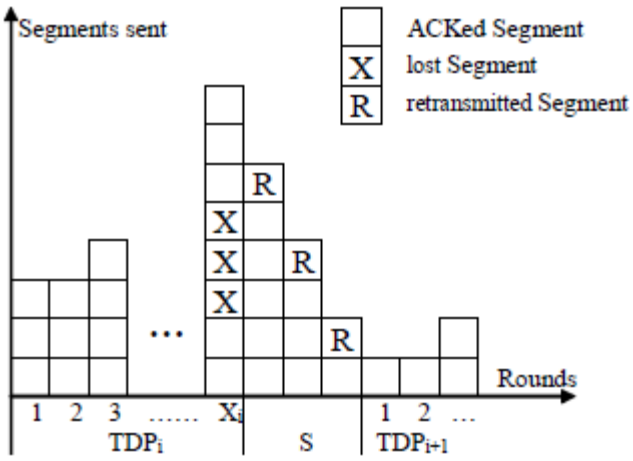


Figura 3.22 Retransmisión de TCP Reno sobre redes OBS [111]

1) Pérdidas sólo por TD

Cuando la longitud *S* de la ráfaga perdida es pequeña, TCP Reno se puede recuperar de la fase de *fast recovery* mediante la retransmisión de múltiples paquetes perdidos en cada ronda, como se ilustra en la Figura 3.22.

Mientras que en el modelo de SACK se ignoró el período de retransmisión que sigue a un TDP, aquí se requiere analizar el caso con *S* retransmisiones del siguiente modo. La primera retransmisión ocurrirá al final de la primera ronda adicional, y después de cada ronda subsecuente, mientras que el emisor reducirá el tamaño de su ventana a la mitad después de la retransmisión de cada segmento perdido. En este caso, Reno siempre mantiene una ventana de congestión mayor que tres de manera que se pueden recibir ACKs duplicados después de la retransmisión de cada segmento de la ráfaga pérdida.

En la Figura 3.22 se pueden apreciar *S* rondas adicionales requeridas para retransmitir los *S* paquetes comparado con el caso de New Reno. Asumiendo que la tasa de pérdida p_b es pequeña y que sólo una ráfaga se pierde en la ronda X_i, el número total de nuevos segmentos Y_a enviados en las rondas de transmisión adicionales siguientes a X_i se puede calcular como [111]:

$$Y_a = \sum_{k=0}^{S-1} \frac{1}{2^k} W_{X_i} - S = \sum_{k=0}^{S-1} (2^{-1})^k W_{X_i} - S = \left(\frac{1 - 2^{-S}}{1 - 2^{-1}} \right) W_{X_i} - S$$

$$\boldsymbol{Y_a = \left(2 - \frac{1}{2^{S-1}} \right) W_{X_i} - S} \qquad \textbf{(3.84)}$$

Después de que arriba un nuevo ACK, el TDP_{i+1} inicia con un tamaño de ventana de transmisión de $W_{Xi}/2^S$. El número total de segmentos transmitidos en el TDP_i es por tanto $Y_i = \alpha_i + \gamma_i + Y_a$, y de manera similar al cálculo de SACK, se tiene que [107]:

$$E[Y] = E[\alpha] + E[\gamma] + E[Y_a]$$

$$E[Y] = \frac{S}{p_b} + \frac{1}{2}E[W_X] + \left(2 - \frac{1}{2^{S-1}}\right)E[W_X] - S$$

$$\boldsymbol{E[Y] = \frac{1-p_b}{p_b}S + \left(\frac{5}{2} - \frac{1}{2^{S-1}}\right)E[W_X]} \quad \textbf{(3.85)}$$

Además, el tamaño de la ventana de transmisión se debería calcular como [107]:

$$\boldsymbol{W_{X_i} = max\left(1, \frac{W_{X_{i-1}}}{2^S}\right) + \frac{X_i}{b}} \quad \textbf{(3.86)}$$

Para el caso de pérdidas sólo por TD, la ventana de congestión se puede reducir a la mitad múltiples veces pero aún debe permanecer mayor que 3 para desencadenar *fast retransmit/recovery* en Reno, antes de que todos los S segmentos sean retransmitidos exitosamente, es decir $\log_2 W_x > S$[48], por lo que la ventana inicial del siguiente TDP es siempre mayor que 1. Por tanto [107],

$$\boldsymbol{W_{X_i} = \frac{W_{X_{i-1}}}{2^S} + \frac{X_i}{b}} \quad \textbf{(3.87)}$$

De la ecuación (3.87) se tiene que [107]:

$$\boldsymbol{E[X] = bE[W_X] - \frac{bE[W_X]}{2^S}} \quad \textbf{(3.88)}$$

[48] Durante la fase de *slow start* el valor de W_X después de k intercambios de ida y vuelta (RTT) es $W_X=2^k$. Por lo tanto, el número de k rondas que le toma a TCP en alcanzar el valor de Wx es $k=\log_2 (Wx)$.

Los parámetros restantes necesarios para calcular el *throughput* se pueden obtener igual que el caso de SACK. Por lo tanto, y dado que Y_i también se puede expresar sumando el número de segmentos en todas las rondas X_i previas y en las S rondas adicionales, se obtiene que [107]:

$$Y_i = \sum_{k=0}^{\frac{X_i}{b}-1}\left(\frac{W_{X_{i-1}}}{2^S}+k\right)b+\left[W_{X_i}-S+\sum_{j=1}^{S-1}\frac{1}{2^j}W_{X_i}-S\right]$$

$$Y_i = \frac{W_{X_{i-1}}}{2^S}b\left(\frac{X_i}{b}\right)+b\left(1+2+...+\frac{X_i}{b}-1\right)+\left[W_{X_i}-S+\left(\sum_{j=0}^{S-1}\frac{1}{2^j}W_{X_i}-W_{X_i}\right)-S\right]$$

$$\boldsymbol{Y_i = \frac{X_i W_{X_{i-1}}}{2^S}+\frac{X_i}{2}\left(\frac{X_i}{b}-1\right)+\left(2-\frac{1}{2^{S-1}}\right)W_{X_i}-2S} \qquad \textbf{(3.89)}$$

siempre y cuando S sea bastante grande tal que $\frac{1}{2^S} << 1$. Además, tomando la media en ambos lados de la ecuación anterior, y aplicando (3.88) en ella, se obtiene que [107]:

$$E[Y] = \frac{E[X]^2}{2b}+\frac{E[X]}{2}+2E[W_X]-2S$$

$$E[Y] = \frac{(bE[W_X]-b)^2}{2b}+\frac{bE[W_X]-b}{2}+2E[W_X]-2S$$

$$\boldsymbol{E[Y] = \frac{bE[W_X]^2}{2}-\frac{b-4}{2}E[W_X]-2S} \qquad \textbf{(3.90)}$$

Combinando las ecuaciones (3.85) y (3.90), se tiene que [107]:

$$\frac{1-p_b}{p_b}S+\frac{5}{2}E[W_X] = \frac{b}{2}E[W_X]^2-\frac{b-4}{2}E[W_X]-2S$$

$$\frac{b}{2}E[W_X]^2 - \frac{b+1}{2}E[W_X] - \frac{1+p_b}{p_b}S = 0$$

$$E[W_X] = \frac{\frac{b+1}{2} + \sqrt{\left(\frac{b+1}{2}\right)^2 + 2b\left(\frac{1+p_b}{p_b}S\right)}}{b}$$

$$E[W_X] = \frac{1}{2} + \frac{1}{2b} + \sqrt{\frac{\left(\frac{b+1}{2}\right)^2}{b^2} + \frac{2b\left(\frac{1+p_b}{p_b}S\right)}{b^2}}$$

$$\boldsymbol{E[W_X] = \frac{1}{2} + \frac{1}{2b} + \sqrt{\left(\frac{1}{2} + \frac{1}{2b}\right)^2 + \frac{2S}{bp_b}}} \quad \textbf{(3.91)}$$

$$\boldsymbol{E[W_X] \approx \sqrt{\frac{2S}{bp_b}}} \quad \textbf{(3.92)}$$

Sustituyendo (3.91) en (3.85) se tiene que [107]:

$$E[Y] = \frac{5}{2}E[W_X] + \frac{1-p_b}{p_b}S$$

$$\boldsymbol{E[Y] = \frac{5}{4} + \frac{5}{4b} + \sqrt{\left(\frac{5}{4} + \frac{5}{4b}\right)^2 + \frac{25S}{2bp_b}} + \frac{1-p_b}{p_b}S} \quad \textbf{(3.93)}$$

Además, reemplazando (3.91) en (3.88) se obtiene $E[X]$ como sigue [107]:

$$E[X] = bE[W_X] - b$$

$$\boldsymbol{E[X] = -\frac{b}{2} + \frac{1}{2} + \sqrt{\left(\frac{b}{2} + \frac{1}{2}\right)^2 + \frac{2bS}{p_b}}} \quad \textbf{(3.94)}$$

De manera similar, se obtiene la expresión para $E[A]$. Por lo tanto,

$$E[A] = \mathrm{RTT}(E[X] + S)$$

$$\boldsymbol{E[A] = RTT\left(-\frac{b}{2} + \frac{1}{2} + S + \sqrt{\left(\frac{b}{2} + \frac{1}{2}\right)^2 + \frac{2bS}{p_b}} \right)} \quad \textbf{(3.95)}$$

Con una tasa de pérdida p_b pequeña, $E[A]$ se puede aproximar como [107]

$$\boldsymbol{E[A] \approx RTT\sqrt{\frac{2bS}{p_b}}} \quad \textbf{(3.96)}$$

2) Pérdidas mixtas por TD y TO en Reno

Considere un período de renovación en la evolución de la ventana de congestión con Reno que consiste tanto de TDP y TOP como se ilustra en la Figura 3.23, que muestra dos TDPs separados por un período de TO junto con la fase de recuperación.

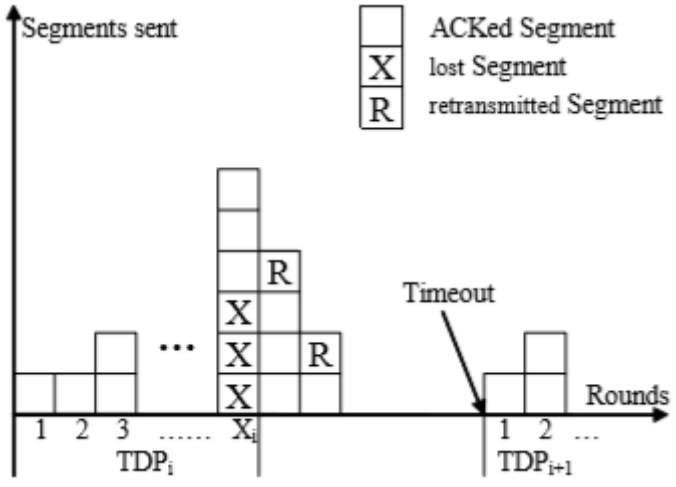


Figura 3.23 Evolución de la ventana de TCP Reno con TO y TDP [111]

Cuando la longitud *S* de la ráfaga perdida es grande, la retransmisión de múltiples paquetes perdidos puede resultar en un evento de TO ocasionado por pérdidas indicadas mediante TD consecutivos, debido a la falta del número necesario de ACKs duplicados para activar la fase de *fast retransmit*. Esto ocurre en el caso de Reno porque después de cada pérdida por TD, la ventana de congestión se reducirá a la mitad; con varios cortes consecutivos, su valor disminuirá a 1 y no se pueden recibir ACKs duplicados antes de que todos los segmentos perdidos puedan ser retransmitidos. Por lo tanto, la ventana de transmisión de Reno se estanca y finalmente el temporizador expira, resultando en la activación del *timeout* de retransmisión[49].

Este escenario de pérdidas mixtas por TD y TO sucede cuando la ventana de congestión se reduce a menos de tres antes de que todos los segmentos perdidos sean retransmitidos, es decir, $\log_2 W_x < S$. En este caso, un TDP es seguido por $\log_2 (W_X) - 1$[50] retransmisiones y un TOP. El siguiente TDP inicia con un tamaño de ventana de uno, es decir, en la fase de *slow start*. El número total de segmentos enviados durante el TDP y el siguiente TOP es $E[Y^*] = E[Y] + E[H]$, con una duración $E[\text{TDP}^*] = E[A] + E[Z^{\text{TO}}]$.

Para estimar $E[Y]$, se sigue todavía el mismo método utilizado para el caso con pérdidas sólo por TD. Así, el número medio de segmentos transmitidos está dado por [107]:

$$Y_a = \sum_{k=0}^{\log_2 W_{X_i} - 2} \frac{1}{2^k} W_{X_i} - (\log_2 W_{X_i} - 1) = \sum_{k=0}^{\log_2 W_{X_i} - 2} (2^{-1})^k W_{X_i} - \log_2 W_{X_i} + 1$$

$$Y_a = \left(\frac{1 - 2^{-(\log_2 W_{X_i} - 1)}}{1 - 2^{-1}} \right) W_{X_i} - \log_2 W_{X_i} + 1$$

$$\boldsymbol{Y_a = \left(2 - \frac{2}{2^{\log_2 W_{X_i} - 1}} \right) W_{X_i} - \log_2 W_{X_i} + 1} \qquad \textbf{(3.97)}$$

[49] Tal situación de pérdida no fue considerada en [85] ya que ignoró la retransmisión que sigue a un TDP.

[50] Puesto que el valor de Wx se reduce a la mitad por cada pérdida por TD, cuando $W_x/2^k=2$ TCP no podrá recibir 3 ACK duplicados y por ende desencadenaría un TO. Por lo tanto, el número de retransmisiones hasta ese instante se puede estimar como $k=\log_2(W_x/2)=\log_2(W_x) - 1$.

$$E[Y] = E[\alpha] + E[\gamma] + E[Y_a]$$

$$E[Y] = \frac{S}{p_b} + \frac{1}{2}E[W_X] + \left(2 - \frac{2}{2^{\log_2 E[W_X]-1}}\right)E[W_X] - \log_2 E[W_X] + 1$$

$$\boldsymbol{E[Y] = \left(\frac{5}{2} - \frac{2}{2^{\log_2 E[W_X]-1}}\right)E[W_X] - \log_2 E[W_X] + \frac{S}{p_b} + 1} \quad \textbf{(3.98)}$$

y asumiendo que el *slow start threshold* es 1

$$\boldsymbol{W_{X_i} = \frac{X_i}{b}} \quad \textbf{(3.99)}$$

$$\boldsymbol{E[X] = bE[W_X]} \quad \textbf{(3.100)}$$

Al igual que el caso de pérdidas sólo por TD, se obtiene $E[W_x]$ de la misma forma que en (3.91) y se tiene que [107]:

$$\boldsymbol{E[Y] = \frac{5}{2}E[W_X] - \log_2 E[W_X] + \frac{S}{p_b} + 1 \approx \frac{S}{p_b}} \quad \textbf{(3.101)}$$

y

$$E[A] = (E[X] + \log_2 E[W_X])\text{RTT}$$

$$\boldsymbol{E[A] \approx \text{RTT}\left(\sqrt{\frac{2bS}{p_b}} + \log_2 E[W_X]\right)} \quad \textbf{(3.102)}$$

Por tanto [107]:

$$E[Y^*] = E[Y] + E[H]$$

$$\boldsymbol{E[Y^*] \approx \frac{S}{p_b} + \frac{p_b}{1-p_b} \approx \frac{S}{p_b}} \qquad \textbf{(3.103)}$$

Y por ende se tiene que [107]:

$$E[\text{TDP*}] = E[A] + E[\text{Z}^{\text{TO}}]$$

$$E[\text{TDP*}] \approx \text{RTT}\left(\sqrt{\frac{2bS}{p_b}} + \log_2 E[W_X]\right) + \text{RTO}\frac{f(p_b)}{1-p_b}$$

$$\boldsymbol{E[\text{TDP*}] \approx \text{RTT}\left(\sqrt{\frac{2bS}{p_b}} + \log_2 \sqrt{\frac{2S}{bp_b}}\right) + \text{RTO}} \qquad \textbf{(3.104)}$$

En base a los dos casos anteriores, y utilizando las expresiones de $E[Y]$, $E[X]$, y $E[W_x]$ para evaluar los demás parámetros para la estimación del *throughput* de TCP Reno, este se puede expresar como [107]:

$\boldsymbol{B(p, T_b) =}$ **(3.105)**

$$\begin{cases} \dfrac{\dfrac{S}{p_b}}{\text{RTT}\left(\sqrt{\dfrac{2bS}{p_b}} + S\right)} + o\left(\dfrac{1}{\sqrt{p_b}}\right) & \textit{Para} \log_2 W_X > S \;\; o \;\; \log_2 \sqrt{\dfrac{2S}{bp_b}} > S \\ \dfrac{\dfrac{S}{p_b}}{\text{RTT}\left(\sqrt{\dfrac{2bS}{p_b}} + \log_2 \sqrt{\dfrac{2S}{bp_b}}\right) + \text{RTO}} + o\left(\dfrac{1}{\sqrt{p_b}}\right) & \textit{Para} \log_2 W_X < S \;\; o \;\; \log_2 \sqrt{\dfrac{2S}{bp_b}} < S \end{cases}$$

Cabe señalar que para el primer caso donde $\log_2 \sqrt{\frac{2S}{bp_b}} > S$, Reno siempre tiene un *throughput*

menor que New Reno (aunque tienen el mismo número de rondas de retransmisión), ya que Reno recibe más penalidad por retransmisión reduciendo a la mitad la ventana inicial del siguiente TDP múltiples veces. Para el segundo caso donde $\log_2 \sqrt{\frac{2S}{bp_b}} < S$, existen compromisos en el *throughput* de Reno y New Reno como se puede apreciar en (3.83) y (3.105 para $\log_2 Wx < S$). En comparación con (3.105 para $\log_2 Wx > S$), el denominador de (3.105 para $\log_2 Wx < S$) se incrementa en un factor más RTO, donde el *throughput* de TCP se degrada por el *timeout* de retransmisión después de múltiples retransmisiones por TD. Cuando p_b es pequeña y S es grande tal que $\mathrm{RTT} * \log_2 \sqrt{\frac{2bS}{p_b}} >> \mathrm{RTO}$, (3.105 para $\log_2 Wx < S$) se aproxima a (3.105 para $\log_2 Wx > S$) [107].

Además, si $E[W_x]$ es suficientemente grande tal que $E[W_x] >> \min(S, \log_2 W_X)$ y RTT * $E[W_x]$ >> RTO, entonces tanto (3.105 para $\log_2 Wx > S$) como (3.105 para $\log_2 Wx < S$) (así como también (3.83)) se pueden simplificar aún más a [107]:

$$B(p_b, T_b) = \frac{\frac{S}{p_b}}{\mathrm{RTT}\left(\sqrt{\frac{2bS}{p_b}}\right)} + o\left(\frac{1}{\sqrt{p_b}}\right)$$

$$\boldsymbol{B(p_b, T_b) = \frac{1}{\mathrm{RTT_0} + 2T_b}\sqrt{\frac{S}{2bp_b}} + o\left(\frac{1}{\sqrt{p_b}}\right)} \qquad \textbf{(3.106)}$$

3.4.2.4 Ganancias y penalidades sobre el throughput en redes OBS

Utilizando los modelos del *throughput* de TCP descritos anteriormente, se pueden cuantificar las ganancias y penalidades en el *throughput* de los flujos TCP sobre redes OBS. En este apartado, además de las penalidades tratadas en la subsección 3.4.1, se identifican lo que se denomina penalidad por pérdida, y penalidad por retransmisión para TCP Reno y New-Reno, que afectan

el *throughput* de TCP sobre redes OBS. En las siguientes subsecciones, se evalúan las ganancias y penalidades fijando todos los demás factores, excepto el que se encuentra bajo consideración.

Penalidad por pérdida.- La penalidad por pérdida LP (*Loss Penality*) está relacionada con la reducción en el *throughput* debido a la pérdida de ráfagas (cuya tasa es p_b). Intuitivamente, cuanto mayor es la tasa de pérdida de ráfagas, menor será el *throughput* de TCP. La relación LP se puede definir de acuerdo con la siguiente expresión [107]:

$$LP\ Ratio = \frac{B(sin\ pérdidas)}{B(con\ una\ tasa\ de\ pérdida\ de\ ráfagas\ p_b)}$$

$$LP\ Ratio \approx \frac{\frac{W_{max}}{RTT}}{\frac{1}{RTT}\sqrt{\frac{3S}{2bp_b}}}$$

$$\boldsymbol{LP\ Ratio \approx W_{max}\sqrt{\frac{2bp_b}{3S}}} \qquad \textbf{(3.107)}$$

Penalidad por retardo.- Como se mencionó anteriormente, la penalidad por retardo DP (*Delay Penality*) es ocasionada principalmente por el proceso de ensamblado de ráfagas, que incrementa el tiempo de ida y vuelta de TCP, disminuyendo su *throughput*. En particular, cuanto mayor sea T_b, mayor será dicha penalidad. A continuación, se cuantifica la relación DP fijando todo lo que incluye el tiempo de la primera pérdida de paquetes (excluyendo así cualquier ganancia DFL) excepto el RTT que se incrementa con T_b [107].

$$DP\ Ratio = \frac{B(con\ RTT_0\ original)}{B(RTT\ prolongado\ debido\ al\ ensamblado\ de\ ráfagas)}$$

$$\boldsymbol{LP\ Ratio \approx \frac{RTT_0 + 2T_b}{RTT_0}} \qquad \textbf{(3.108)}$$

Penalidad por retransmisión.- La penalidad por retransmisión RP (*Retransmission Penality*) es la penalización de un período prolongado de retransmisión que es ocasionado por múltiples rondas de retransmisión, durante las cuales se pueden enviar muy pocos paquetes nuevos. Dado que TCP SACK siempre puede retransmitir todos los paquetes de la ráfaga perdida en una o pocas rondas (ya que la información de los bloques faltantes está contenida en los ACKs recibidos), no hay RP puesto que el número de retransmisiones es aproximadamente el mismo independientemente de si se pierde una ráfaga o un paquete. Sin embargo, para TCP New-Reno, se requieren múltiples rondas de retransmisión (S) para los S paquetes contenidos en una ráfaga perdida, antes de que inicie el siguiente TDP. De las ecuaciones (3.78 para $W_x < W_{max}$) y (3.83), para una p_b pequeña y S grande se puede calcular la relación RP (RPR) como se indica a continuación [107]:

$$RPR(New\text{-}Reno) = \frac{B(una\ retransmisión)}{B(S\ retransmisiones)}$$

$$RPR(New\text{-}Reno) \approx 1 + \frac{S-1}{\sqrt{\frac{2bS}{3p_b}} + 1} \approx 1 + \frac{S}{\sqrt{\frac{2bS}{3p_b}}}$$

$$\boldsymbol{RPR(New\text{-}Reno) \approx 1 + \sqrt{\frac{3Sp_b}{2b}}} \qquad \textbf{(3.109)}$$

Nótese que la ecuación (3.109) incrementa con S. Para TCP Reno, la retransmisión no sólo prolonga la duración del TDP como en New-Reno, sino también decrementa la ventana inicial del siguiente TDP reduciendo múltiples veces a la mitad el tamaño de la ventana de congestion durante la retransmisión. Para el caso log $\log_2 \sqrt{\frac{2S}{bp_b}} > S$, para una p_b pequeña se puede calcular la RPR utilizando el modelo definido en la ecuación (3.105 para $\log_2 Wx > S$)

como se indica a continuación [107]:

$$RPR(Reno) = \frac{B(una\ retransmisión)}{B(S\ retransmisiones)}$$

$$RPR(Reno) \approx \frac{\sqrt{\frac{2bS}{p_b}} + S}{\sqrt{\frac{2bS}{3p_b}} + 1} \approx \frac{\sqrt{3}\left(\sqrt{\frac{2bS}{3p_b}} + 1\frac{S}{\sqrt{3}} - 1\right)}{\sqrt{\frac{2bS}{3p_b}} + 1}$$

$$RPR(Reno) \approx \sqrt{3} + \sqrt{3}\frac{S}{\sqrt{\frac{2bS}{p_b}}}$$

$$\boldsymbol{RPR(Reno) \approx \sqrt{3}\left(1 + \sqrt{\frac{Sp_b}{2b}}\right)} \qquad \textbf{(3.110)}$$

Y para el caso donde $\log_2 \sqrt{\frac{2S}{bp_b}} < S$, para una p_b pequeña se puede calcular la RPR utilizando el modelo definido en (3.105 para $\log_2 Wx < S$) como sigue [107]:

$$RPR(Reno) = \frac{B(una\ retransmisión)}{B(S\ retransmisiones)}$$

$$RPR(Reno) \approx \frac{\sqrt{\frac{2bS}{p_b}} + \log_2 \sqrt{\frac{2S}{bp_b}} + \frac{RTO}{RTT}}{\sqrt{\frac{2bS}{3p_b}} + 1}$$

$$RPR(Reno) \approx \frac{\sqrt{3}\left(\sqrt{\frac{2bS}{3p_b}} + 1 + \frac{1}{\sqrt{3}}\log_2 \sqrt{\frac{2S}{bp_b}} + \frac{1}{\sqrt{3}}\frac{RTO}{RTT} - 1\right)}{\sqrt{\frac{2bS}{3p_b}} + 1}$$

$$RPR(Reno) \approx \sqrt{3} + \frac{\sqrt{3}\left(\frac{1}{\sqrt{3}}\log_2\sqrt{\frac{2S}{bp_b}} + \frac{1}{\sqrt{3}}\frac{RTO}{RTT} - 1\right)}{\sqrt{\frac{2bS}{3p_b}}}$$

$$RPR(Reno) \approx \sqrt{3} + \frac{\sqrt{3}\left(\frac{1}{\sqrt{3}}\log_2\sqrt{\frac{2S}{bp_b}} + \frac{1}{\sqrt{3}}\frac{RTO}{RTT}\right)}{\sqrt{\frac{2bS}{3p_b}}}$$

$$RPR(Reno) \approx \sqrt{3} + \sqrt{3}\left(\log_2\sqrt{\frac{2S}{bp_b}} + \frac{RTO}{RTT}\right) \times \sqrt{\frac{p_b}{2bS}}$$

$$\boldsymbol{RPR(Reno) \approx \sqrt{3}\left[1 + \left(\frac{RTO}{RTT} + log_2\sqrt{\frac{2S}{bp_b}}\right) \times \sqrt{\frac{p_b}{2bS}}\right]} \quad \textbf{(3.111)}$$

Nótese que para el primer caso, RPR(Reno)>RPR(New-Reno), y por tanto New-reno siempre presenta un mayor *throughput* que Reno. En el segundo caso, si $1 + \sqrt{\frac{3Sp_b}{2b}} < \sqrt{3}\left[1 + \left(\frac{RTO}{RTT} + log_2\sqrt{\frac{2S}{bp_b}}\right) \times \sqrt{\frac{p_b}{2bS}}\right]$, New-Reno presenta un *throughput* mayor, caso contrario, Reno tiene un mejor *throughput*.

Ganancia DFL.- Si se compara la ecuación (3.78) correspondiente al modelo del *throughput* de TCP SACK en una red OBS con el modelo del *throughput* definido en [85] sin ensamblado de ráfagas (y sin considerar la limitación de ventana W_{max}), cuya expresión es [107]:

$$\boldsymbol{B(p) = \frac{1}{RTT_0}\sqrt{\frac{3}{2bp}} + o\left(\frac{1}{\sqrt{p}}\right)} \quad \textbf{(3.112)}$$

y se ignora la diferencia en cuanto al RTT, se puede apreciar que cuanto mayor sea *S* en la

ecuación (3.78), mayor será el *throughput* y la ganancia DFL, cuya relación se puede cuantificar como se indica a continuación (al igual que antes, se debe ignorar la penalidad por retardo cuando se considera sólo la ganancia DFL) [107]:

$$DFL\ Gain\ Ratio = \frac{B(primera\ \ pérdida\ es\ retrasada)}{B(primera\ \ pérdida\ no\ es\ retrasada)}$$

$$\boldsymbol{DFL\ Gain\ Ratio \approx \sqrt{S}\ (para\ una\ tasa\ de\ pérdida\ p_b\ \ pequeña\ \ fija)} \quad \textbf{(3.113)}$$

La combinación de la ganancia DFL y la penalidad por retransmisión es equivalente a lo que se denomina "Ganancia de correlación" en [25], que no analizó ninguna de las tres implementaciones de TCP en detalle.

3.4.2.5 Impacto de múltiples ganancias y penalidades en TCP

Ya que la ganancia y las penalidades pueden afectar a la vez el *throughput* de TCP, se pueden definir los siguientes rangos especiales para cuantificar el impacto del proceso de ensamblado o tasa de pérdida de ráfagas sobre el *throughput* de TCP.

1) *Rango óptimo de tiempo de ensamblado ATO* (*Assembly Time Optimal*): Para un flujo TCP con un alto ancho de banda de acceso y una tasa de pérdida dada por p_b, puede existir un rango de tiempos de ensamblado donde la ganancia DFL es mayor que las penalidades de retardo y retransmisión, en el cual el flujo TCP alcanza un mayor *throughput* en una red OBS que en una red conmutada de paquetes sin ensamblado de ráfagas (ambas con la misma probabilidad de pérdida); dicho rango se puede definir de acuerdo a [107]:

$$\boldsymbol{ATORange = \bigcup T_b : \frac{DFL\,Gain}{DP \times RP} > 1} \quad \textbf{(3.114)}$$

2) *Rango insensible al tiempo de ensamblado ATI* (*Assembly Time Insensitive*): Para un flujo TCP con un bajo ancho de banda de acceso, podría existir un rango de tiempos de ensamblado para los cuales la ganancia DFL es compatible con todas las penalidades de retardo y retransmisión, en el cual el *throughput* de un flujo TCP es prácticamente independiente de T_b, y por ende, una red OBS tiene aproximadamente el mismo *throughput* que una red conmutada de paquetes sin ensamblado de ráfagas; tal rango se puede determinar como [107]:

$$\boldsymbol{ATI\ Range} = \bigcup \boldsymbol{T_b} : \frac{\boldsymbol{DFL\,Gain}}{\boldsymbol{RP} \times \boldsymbol{DP}} \approx \boldsymbol{1} \tag{3.115}$$

3) *Rango insensible a la tasa de pérdida LRI* (*Loss Rate Insensitive*): Si se toma en cuenta todas las ganancias y penalidades (incluyendo la penalidad por retardo constante para un T_b fijo), existirá un rango donde el *throughput* de TCP es prácticamente independiente de la tasa de pérdida de ráfagas p_b, que se puede evaluar como [107]:

$$\boldsymbol{LRI\ Range} = \bigcup \boldsymbol{p_b} : \frac{\boldsymbol{DFL\,Gain}}{\boldsymbol{DP} \times \boldsymbol{RP} \times \boldsymbol{LP}} \approx \boldsymbol{1} \tag{3.116}$$

CAPÍTULO IV SIMULACIÓN Y RESULTADOS NUMÉRICOS

Existe un sin número de herramientas de simulación de redes que permiten evaluar el comportamiento y desempeño de múltiples tecnologías y protocolos de comunicaciones como *Ethernet*, WDM, MPLS, GPRS, GSM, UMTS, LTE, WiMAX, TCP/IP, UDP, FTP, RTP, etc. empleados en redes de telecomunicaciones cableadas e inalámbricas tales como redes Ad Hoc, WLAN, Mobile-IP, UMTS, Satelitales y Wireless, frente a distintos tipos de tráfico, entre los cuales se pueden mencionar TCP, CBR (*Constant Bit Rate*), On/Off con distribución Exponencial, On/Off con distribución de Pareto, Autosimilar, entre otros; siendo las herramientas más comunes *ns-2*/3, OMNET++ y OPNET, muy útiles dentro del modelado y simulación de sistemas de comunicaciones.

En este capítulo se describirá rápidamente la herramienta de simulación *ns-2* que fue la escogida para para evaluar el comportamiento de TCP sobre OBS, además se presentará el desarrollo de la simulaciones aplicadas sobre distintos escenarios enfocando los resultados en términos de: 1) el *throughput* de TCP en función de la probabilidad de pérdidas de ráfagas, que se contrastará con los modelos analíticos obtenidos de las diferentes expresiones matemáticas descritas en el capítulo anterior; y 2) el número de paquetes ensamblados por ráfaga en el nodo de *edge*, utilizando el modelo de la Figura 2.1 con un solo flujo TCP sin proceso de ensamblado en la dirección de ACK (escenario simplificado) y con proceso de ensamblado en la dirección de ACK (escenario básico).

4.1 Método de simulación [31][53][93][115]

Dos puntos cruciales durante la investigación y desarrollo de sistemas de comunicaciones (que incluso ni siquiera pueden existir), y que básicamente incluyen protocolos, algoritmos, arquitecturas, etc. son: a) La estimación de su desempeño; y b) La comprensión y visualización de su comportamiento a nivel macro (sistema en cuestión) y micro (sus diferentes componentes), los cuales generalmente se pueden llevar a cabo aplicando tres

metodologías diferentes: 1) experimentos con sistemas reales y/o prototipos; 2) análisis matemático; y 3) simulación por ordenador; siendo las dos últimas las de mayor uso, debido a limitaciones de carácter técnico y financiero que se pueden presentar en el primer caso [115].

El método analítico usualmente resulta en expresiones matemáticas que denotan las características límite del sistema, como umbrales inferiores y superiores dentro de los cuales podría variar su desempeño para un caso en particular. Sin embargo, dentro de un análisis de "grano fino" este método conduce a una complejidad inaceptable, a diferencia de la técnica de simulación que puede capturar la dinamicidad de un sistema en ambientes con diferentes escenarios y parámetros con un esfuerzo comparativamente menor.

El método de simulación utiliza siempre un modelo en particular que debe ser creado previamente, y que corresponde a una abstracción del sistema en estudio, es decir que, representa sólo propiedades y características seleccionadas que conducen a una representación reducida del sistema basada en simplificaciones y suposiciones, ya que por lo general el sistema de interés suele ser bastante complejo. La simulación por ordenador se aplica ampliamente en distintos campos científicos y de investigación, donde se tienen varias alternativas como la simulación de eventos discretos, simulación continua, simulación de Monte Carlo, simulación de hojas de cálculo, simulación de rastreo, etc [115].

La simulación de eventos discretos es la técnica más ampliamente utilizada y su éxito radica en que se aplica fácilmente y se ajusta a la mayoría de sistemas donde su estado cambia sólo en ciertos puntos discretos en el tiempo, denominados eventos[51], permitiendo cubrir el análisis de todas las capas de las redes de datos, como por ejemplo el procesamiento de señales en la capa física, acceso al medio en la capa de enlace, enrutamiento en la capa de red, y protocolos y temas de diseño en las capas de transporte y de aplicación. La característica principal de un simulador de eventos discretos es saltar de un evento al siguiente, donde la aparición de un evento puede provocar cambios en el estado del sistema y/o la generación de nuevos avisos de eventos que son llamados posteriormente. Dichos eventos se registran como notificaciones de

[51] Lo que es un evento con exactitud, depende principalmente del sistema y del objetivo de estudio, como ejemplos se pueden mencionar, el envío de un paquete, la recepción de un paquete o la selección de un hipervínculo en una página web [115].

eventos[52] en una entidad denominada Lista Eventos Futuros FEL (*Future Event List*), que es una estructura de datos apropiada para gestionar todos los eventos previstos durante la simulación, a través de funciones eficientes que permiten insertar, buscar y eliminar los avisos de eventos, colocados en la FEL [115].

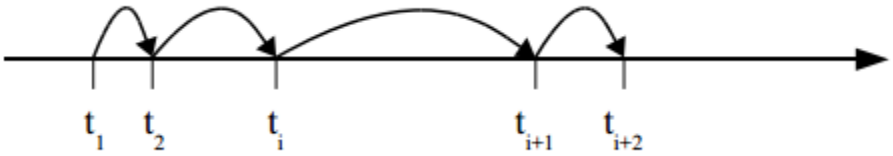


Figura 4.1 Principio de la simulación de eventos discretos [115]

La Figura 4.1 ilustra la evolución de una simulación de eventos discretos, los cuales ocurren en el tiempo t_i donde se crea un estado instantáneo del sistema (que puede cambiar) en la memoria del ordenador y que contiene todos los datos necesarios para que pueda avanzar la simulación. De manera general todos los simuladores de eventos discretos incluyen los siguientes componentes [115]:

- *Estado del sistema*: Conjunto de variables que describen el estado del sistema.
- *Reloj*: Provee el tiempo actual durante la simulación.
- *Lista de eventos futuros*: Estructura de datos apropiada para administrar los eventos.
- *Contadores estadísticos*: Conjunto de variables que contienen información estadística sobre el rendimiento del sistema.
- *Rutina de inicialización:* Utilizada para inicializar el modelo de simulación y ajustar el reloj a cero (0).
- *Rutina de tiempo*: Permite recuperar el siguiente evento de la lista de eventos futuros y avanzar el reloj al instante de ocurrencia del evento.

[52] Un aviso o notificación de evento está compuesto de al menos dos datos "tiempo" y "tipo", donde el primero especifica el instante en que ocurrirá un evento y el segundo hace referencia al tipo de evento [115].

- *Rutina de eventos*: Se llama cuando ocurre un evento en particular durante la simulación. Normalmente, para cada tipo de evento se define una rutina de evento conocido como *"Handler"* en la literatura.

A continuación se presenta una descripción muy breve del simulador de eventos discretos *ns-2* (*network simulator 2*), en base al cual se desarrolló el modelo de simulación para este proyecto de investigación con la ayuda de un módulo que provee el soporte para el análisis de redes OBS, y que se describirá con mayor detalle en el siguiente apartado.

El simulador de redes *ns-2* es una aplicación *Open Source* basada en eventos discretos, desarrollado originalmente dentro del proyecto VINT [31], principalmente para simulaciones orientadas a redes de conmutación de paquetes del stack TCP/IP, que se encuentra disponible en múltiples plataformas tales como Solaris, Linux y Windows; y se utiliza ampliamente en el ámbito educativo y de investigación para el modelado y simulación de redes.

La herramienta *ns-2* consiste de dos lenguajes de programación: C++ y OTcl (*Object-oriented Tool Command Language*), donde el primero corresponde al núcleo del simulador que provee el mecanismo interno de la simulación (backend) que se invoca simplemente tecleando "ns" en la línea de comandos, mientras que el segundo se utiliza como una interfaz (front end) para construir el modelo de simulación mediante la configuración de los objetos tales como nodos, enlaces, agentes, aplicaciones, etc. así como también la planificación de eventos discretos. El simulador incluye un gran número de clases C++ y OTcl denominadas "jerarquía compilada" y "jerarquía interpretada" respectivamente, que pueden estar o no enlazadas entre si, mediante una interfaz OTcl/C++ denominada TclCL (Tcl with Classes) que permite vincular las clases de C++ con las de OTcl.

Dentro de la arquitectura del simulador *ns-2*, los agentes actúan como generadores y consumidores de paquetes, que pueden estar conectados con nodos u otros agentes, generalmente a nivel de las capas de transporte y aplicación, donde una sesión TCP es unidireccional y se emula mediante un agente que actúa como una fuente TCP *"Source"* conectado a un nodo origen y un agente que actúa como un sumidero TCP *"Sink"* conectado a un nodo destino [93].

Conforme avanza la simulación se genera un conjunto de datos que se almacenan en un archivo de traza, a partir del cual se puede procesar la información de interés mediante otros lenguajes de programación como Perl y AWK, por medio de los cuales se filtra la traza y se obtienen los resultados de interés que finalmente se suelen representar mediante gráficos que muestran parámetros o variables de la red tales como *throughput*, retardo, *jitter*, pérdida de paquetes, etc. y su variación en función del tiempo o de algún otro tipo de variable; así como también la visualización del desarrollo de la simulación de manera gráfica mediante la herramienta "nam" (que trabaja en conjunto con *ns-2*), específicamente el envío/recepción de paquetes, descarte de paquetes, estado de colas, etc.

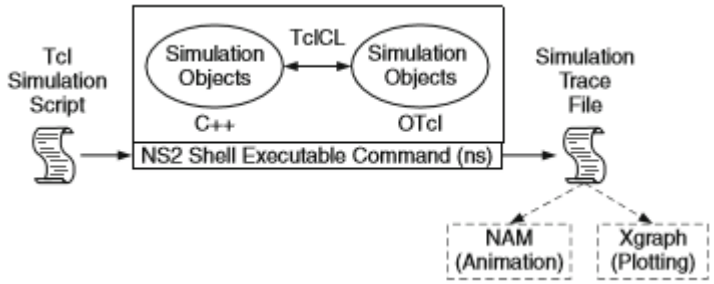


Figura 4.2 Arquitectura básica de ns-2 [53]

Figura 4.3 Formato de una traza de paquetes [53]

La Figura 4.2 ilustra un diagrama esquemático de la arquitectura básica de *ns-2*, mientras que la Figura 4.3 presenta el formato normal (no wireless) de una traza de paquetes, donde [53]:

- *Event Type:* El tipo (variable *_type*) del objeto Trace que genera la cadena de caracteres *"string"* de la traza. La lista completa de los tipos de eventos está especificada en el archivo "˜ns/tcl/lib/ns-trace.tcl", de los cuales los más utilizados son:
  - "+" representa el evento de ingreso de un paquete en la cola
  - "-" representa el evento de egreso de un paquete de la cola

- "r" representa el evento de recepción de un paquete
- "d" representa el evento de descarte de un paquete (es decir enviado a *"dropHead_"*)
- "c" representa el evento de colisión de un paquete en el nivel MAC

- *Time:* Instante en el que se crea el registro de la traza del paquete.
- *Sending Node and Receiving Node:* IDs de los nodos localizados antes y después, respectivamente, del objeto de traza que crea este registro.
- *Payload Type:* Nombre del tipo de Payload tal como TCP, UDP, CBR, EXP, etc.
- *Packet Size:* Tamaño del paquete en bytes
- *Flags:* Una cadena de caracteres de 7 dígitos, cada uno establecido a "-" cuando está deshabilitado; en caso contrario será fijado de acuerdo a lo siguiente:
  - 1ero: Fijado a "E" si un eco de ECN (*Explicit Congestion Notification*) está habilitado.
  - 2do: Establecido a "P" si la prioridad en la cabecera IP está habilitada.
  - 3ero: No se utiliza.
  - 4to: Fijado a "A" si el segmento TCP correspondiente toma una acción ante una condición de congestión (p. ej. cerrar la ventana de congestión).
  - 5to: Establecido a "E" si ha ocurrido congestión.
  - 6to: Fijado a "F" si se utiliza TCP *fast start*.
  - 7mo: Establecido a "N" cuando el protocolo de la capa de transporte es capaz de utilizar ECN (*Explicit Congestion Notification*).
- *Flow ID: El campo field "fid_" de la cabecera del paquete IP.*
- *Source Address and Destination Address:* Direcciones origen y destino de un paquete especificado en la cabecera de un paquete IP. Para un esquema de direccionamiento plano, el formato de estos dos campos es "a.b", donde "a" es la dirección y "b" es el puerto, mientras que para un direccionamiento jerárquico es "x.y.z.w", donde "x.y.z" es la dirección con una connotación de tres niveles y "w" es el puerto.
- *Sequence Number:* El número de secuencia correspondiente al protocolo especificado en el tipo de Payload del paquete.
- *Packet ID Unique:* Identificador único de un paquete, almacenado en su cabecera común *hdr_cmn*.

4.2 Simulador adoptado [31][77][93]

La herramienta de simulación utilizada para este proyecto de investigación es OBS-ns, que fue diseñada y desarrollada sobre la estructura del simulador de redes *ns-2*, y que incluye básicamente un módulo denominado obs-0.9a para extender la funcionalidad de *ns-2* en cuanto al soporte de la tecnología de conmutación óptica de ráfagas, donde su implementación principal está escrita en C++, mientras que la configuración de la red a simular se realiza mediante la versión orientada a objetos del lenguaje de programación tcl conocido como OTcl.

El simulador OBS-ns fue desarrollado como parte de un proyecto de investigación en redes ópticas WDM patrocinado por Alcatel, bajo la supervisión del Prof. Sivalingam, durante el período de diciembre de 2000 a noviembre de 2001 [93]. La estructura del simulador OBS-ns modifica principalmente la implementación del Classifier de *ns-2* para encapsular la clasificación y la funcionalidad de planificación. Los clasificadores rediseñados incluyen los siguientes [93]:

Classifier-edge.- Embebido en el nodo de *edge* para manejar tanto paquetes TCP/ACK (u otros provenientes de la red de acceso) como ráfagas OBS, está diseñado para realizar las siguientes funciones: 1) enrutamiento de las cabeceras y ráfagas de datos; 2) planificación de los canales de datos y de control, introduciendo retardos mediante FDLs si fuera necesario; 3) en el futuro, el nodo de *edge* podría ser diseñado para realizar control de admisión.

Classifier-core.- Similar al *classifier-edge*, con la diferencia de que el clasificador de direcciones en el nodo de *core*, maneja únicamente ráfagas OBS.

Classifier-obs-port[53] (**antes *Portclassifier-edge*).-** Envía los paquetes TCP generados en un nodo de *edge* de ingreso al Agente Integrado "*Integrated Agent*" conectado a ese nodo, mientras que en un nodo de egreso estos paquetes se envían al agente sumidero "*Sink*" conectado al puerto correspondiente.

[53] La implementación de un *Port Classifier* en *ns-2* que permite clasificar únicamente los paquetes cuya dirección coincide con el nodo actual, en sus respectivos puertos destino, no es aplicable a la arquitectura de OBS, puesto que un nodo de *edge* es tanto el origen como el destino de los paquetes TCP/ACK (u otros) provenientes de la red de acceso.

El módulo obs-0.9a original, permitía sólo el uso de paquetes TCP/ACK (extendido posteriormente para el soporte de paquetes UDP, EXP, CBR, PARETO, SELFSIM, etc.) y un esquema de encaminamiento plano *"flat routing"*, limitando con este último el enrutamiento de los paquetes únicamente dentro del dominio OBS desde un nodo de *edge* de ingreso hacia un nodo de *edge* de egreso, por lo tanto, los agentes de tráfico debían estar conectados directamente a los nodos de *edge* y no a otros nodos de acceso fuera de la red OBS, como se requiere para la evaluación de este proyecto de investigación.

Por lo antes indicado, se exploró las funcionalidades que ofrecía la modificación realizada al módulo obs-0.9a en [77], enfocada principalmente a un ambiente de acceso inalámbrico WLAN sobre OBS, donde se extendió la funcionalidad OBS-ns mediante la adaptación de un esquema de enrutamiento jerárquico *"hierarchical routing"* para soportar el encaminamiento de los paquetes desde una estación base BS (*Base Station*) conectada a un nodo de *edge* de ingreso hacia otra estación conectada a un nodo de *edge* de egreso, con lo cual, se llegó a determinar que esta última versión del módulo obs-0.9a, con ciertos cambios, se puede adaptar a los requerimientos que demandan la evaluación de los dos modelos de TCP sobre OBS, objeto de este estudio. En este sentido, se describen a continuación los principales componentes del simulador OBS-ns y sus funcionalidades más relevantes, así como también los cambios realizados en el código fuente.

4.2.1 Enlace WDM (*Wavelength Division Multiplexing*)

La simulación de enlaces *WDM* necesita una simulación de canal explícito, lo cual requiere una nueva implementación de enlace, denominada *obs_simplex-FiberLink*, que incluye un objeto de la clase *OBSFiberLinkDelay* para calcular el tiempo de propagación e introducir el retardo FDL en caso de que sea necesario.

El objeto *obs_SimpleFiberLink* en OBS-ns se utiliza para representar una instancia de un enlace unidireccional que proporciona la funcionalidad de multicanal implementado sobre un solo enlace; característica que es manejada por los clasificadores conectados a el,

cuyos planificadores programan "*n*" transmisiones simultáneas, donde "*n*" denota el número de canales en el enlace. La única modificación al enlace genérico proporcionado por *ns-2*, es la introducción de la propiedad clear channel de un enlace óptico, que se consigue reemplazando el objeto *queue* de la instancia *link* con un simple conector que transfiere paquetes de un extremo a otro con un determinado tiempo de propagación, pero sin ningún tipo de retardo de encolamiento.

El enlace óptico introduce únicamente el retardo de propagación que es uno de los parámetros de configuración, cuyo procedimiento Otcl se describe a continuación:

```
Simulator instproc obs_simplex-FiberLink { n1 n2 bw delay linktype nlambda
args } {
       $self instvar link_

        # create the common simple-link between nodes
        #eval $self simplex-link $n1 $n2 $bw $delay $linktype $args

        set sid [$n1 id]
        set did [$n2 id]

      # use connector to replace queue, in OWNS
      if { $linktype == "ErrorModule" } {
            if { [llength $args] > 0 } {
                  set q [eval new $linktype $args]
                } else {
                        set q [new $linktype Fid]
                }
      } else {
            set q [new Connector]
      }

      # create fiber link
      set link_($sid:$did) [new obs_SimpleFiberLink $n1 $n2 $bw $delay $q]

      # set fiber delay link's wvlen_num_
      [$link_($sid:$did) set link_] set wvlen_num_ $nlambda

        $n1 add-neighbor $n2

#GMG - uncommented the tracing trace and nam trace
         set tracefile [$self get-ns-traceall]
         if {$tracefile != ""} {
                 $self trace-queue $n1 $n2 $tracefile
         }
         set namtracefile [$self get-nam-traceall]
         if {$namtracefile != ""} {
                 $self namtrace-queue $n1 $n2 $namtracefile
```

```
        }
# Register this simplex link in nam link list. Treat it as a duplex link in
nam
        $self register-nam-linkconfig $link_($sid:$did)

                        # Store wavelength information
    set wbw [expr [$self bw_parse $bw] / $nlambda]

# call C++ to register wavelength information - not reqd for OBS
      #[$self get-wassign-logic] register-wvlen $sid $did $nlambda $wbw

# set trace on lightpath classifier of nodes GMG commented out the lines
below, as they are      # inserted above
        # set tracefile [$self get-ns-traceall]
        # set namtracefile [$self get-nam-traceall]

# direct dropped packets from classifiers to link queue's drophead drop-
target is in C++ #command

         set classifier [$n1 set classifier_]
         set droptarget [$classifier drop-target]

# Sigh, there might have more than one link queue which help drop packets,
but we only #select the first one. In nam, it does not make difference, but
in trace file, there might be #confused with the normal dropped packets.
      if { $droptarget == "" } {
            set drophead [$link_($sid:$did) set drophead_]
            eval $classifier drop-target $drophead
      }

}

Class obs_SimpleFiberLink -superclass SimpleLink

obs_SimpleFiberLink instproc init { src dst bw delay q {lltype
"OBSFiberDelayLink"} } {
      $self next $src $dst $bw $delay $q $lltype
      }
```

4.2.2 Integrated Agent

El stack de protocolos incluye la introducción y generación de los paquetes de cabecera BHP (*Burst Header Packet*) y sus correspondientes ráfagas de datos, implementados utilizando un nuevo agente denominado "*Integrated Agent*", derivado de la clase *Agent*, que realiza el proceso de ensamblado y desensamblado de ráfagas. En el primer caso, recolecta los paquetes TCP destinados a un nodo de *edge* común hasta que la suma de los paquetes alcanza un tamaño de

ráfaga predefinido o hasta que expira el temporizador de ensamblado (que fue iniciado al recibir el primer paquete); las cabeceras así como también las ráfagas de datos son generadas y enviadas simultáneamente al *Edge Classifier* que las reenvía hacia su destino. En el segundo caso, separa los paquetes TCP individuales parte de la ráfaga entrante y los reenvía al *Edge Classifier*, el cual a su vez los reenvía a los respectivos agentes sumidero "*Sink*" a través del *Classifier OBS Port*.

4.2.2.1 Formato del paquete

Para manejar tanto las cabeceras como las ráfagas de datos, el *Integrated Agent* define un solo formato de paquete denominado IPKT, que incluye: 1) los campos[54] respectivos que son comunes en su mayor parte para ambos tipos de paquetes; 2) métodos para acceder a los miembros de su estructura; y 3) una función genérica para acceder explícitamente a la estructura del paquete IPKT desde la cabecera común (hdr_cmn).

El paquete IPKT se encuentra embebido en un paquete IP, y utiliza los campos de "*payload*" y "*priority*" para ubicar la ráfaga de datos en el primer caso (por lo que no es necesario un campo de "*payload*" en este nuevo tipo de paquete); y en el segundo caso para diferenciar entre el paquete de un BHP y el de una ráfaga de datos, cuya estructura se presenta a continuación:

```
/* Structure of the Integrated packet header. Note: The Integrated packet
can represent both the burst or the BHP, depending upon the value of the
ipheader's priority bit. If set to zero represents a bhp  or a control
packet. If set to one represents a data-burst or a burst data packet
(bdp).*/
struct hdr_IPKT
{
// fields common to BHP and DB Represents the burst identifier */
      u_long  C_burst_id_;
      /* Represents the burst size */
      u_int  C_burst_size_;
```

[54] Aunque la arquitectura de control de OBS requiere muchos campos, en la implementación de OBS-ns se definió un menor número de campos por temas de eficiencia, que son importantes para las simulaciones [93].

```
      // BHP specific flds only
       /* Number of TCP (or other) packets in the data-burst */
       u_int npkts_;
       // Refer the to BurstManager for these defaults
      /* The guard band value */
      u_int pcntguard_;
      /* The end to end delay */
      double   end_end_delay_;
      /* The offset time */
      double   offset_time_;
      /* The per control packet / BHP processing time */
      double   delta_ ;
      // GMG - added link FDL delay value, so it is easily accessible by
next node
      double FDL_delay_;
      // GMG - added BHP tx delay value, so it is easily accessible by next
node
      double tx_delay_;
/*GMG - added timestamp field; cannot always use the common header
timestamp; for UDP agents, it is multplied by 8000 and truncated, which
leads to roundoff error (by as much as 125 mu s)*/
      double timestamp_;
      //GMG - added sequence number, for tracing the IPKT
      unsigned long int seqno;
/*GMG - added IPKT priority class (note: cannot use ip hdr prio_ field
because that is already used to indicate DB or BHP*/
      int prio_;
/*GMG -- added count of #FDLs DB will have traversed, either globally or
for node, depending on FDL option.  Note that count is maintained in BHP
when setting up path for DB.  We will initialize the field to 0 in both BHP
and DB.*/
      int fdl_count_;
/*GMG -- added indication of whether edge node electronic buffer has been
decremented, as well as pointer to base classifier of ingress edge node.
Needed by link recv method to determine if buffer should be decremented
(done by first link, attached to edge node)*/
      int ebuf_ind;
      BaseClassifier *bc_ingress;

      // accessors methods
      u_long& C_burst_id() { return (C_burst_id_); }
      u_int& C_burst_size() { return(C_burst_size_); }
      u_int& npkts() { return(npkts_); }
      u_int& pcntguard() { return (pcntguard_); }
      double& offset_time() { return(offset_time_); }
      double& end_end_delay() { return(end_end_delay_); }
      /* Needed to access the packet from the provided header */
      static int offset_ipkt_;
      inline static int& offset_ipkt() { return offset_ipkt_; }
      inline static hdr_IPKT* access( Packet *p )
      {
          return (hdr_IPKT*) p->access(offset_ipkt_);
      }
};
```

4.2.2.2 *El método recv*

Es el único punto de entrada de los paquetes hacia el *Integrated Agent* y es invocado por el *Port Clasiffier* para enviar los paquetes TCP y ráfagas OBS a dicho agente. Este método verifica si el paquete recibido es TCP/ACK (u otros provenientes de la red de acceso) o IPKT y según esto llama a los métodos respectivos de ensamblado y desensamblado de ráfagas.

4.2.2.3 *Ensamblado/desensamblado*

Cada *Integrated Agent* mantiene un arreglo de objetos de la clase *BurstManager*[55], cada uno asociado con uno de los nodos de *edge* de la red, a través de la correspondencia de la dirección del nodo de *edge* con el índice del arreglo. Los objetos *BurstManager* mantienen una cola igual al tamaño máximo de ráfaga y recolectan paquetes dirigidos a los nodos de *edge* con los que están asociados. Además, existe un temporizador asociado con cada *BurstManager* que se inicia cuando un paquete es añadido a la cola vacía, es decir cuando se recibe el primer paquete de una ráfaga, la misma que se genera al alcanzar el umbral predefinido del tamaño de ráfaga o cuando expira el temporizador de ensamblado. A continuación se presenta la definición de las clases indicadas:

```
class  BurstManager : public NsObject
{
    public:
/* constructs a new BurstManager object - Note: although the burst manager
can be created via     otcl, we intend to create it in the Integrated agent
block in C++ itself. We just use the tcl interface to configure the
parameters refer BurstManager Class implementation on how-to-configure the
burst-manager parameters */
     BurstManager();
/* Intended to be called by the associated Edge node's IPKT Agent (or
Integrated Agent).
```

[55] Un nodo de *edge* no requiere mantener un objeto *BurstManager* para sí mismo, sin embargo se utiliza este método para reducir la complejidad de la programación [93].

```
destnodeid -- represents the destination node id of the destination
edge router.*/
     void init( IPKTAgent* parent, int destnodeid, int priority );
/* Handles a recv of a packet, when a packet is received it is queued on a
per-destination basis, and a burst is released based on the temporal
burstification algorithm show above. */
     void recv( Packet *p, Handler *h);
/* Timeout method. Intended to be called when a timeout occurs */
     void timeout();
/* Static accessor/modifiers to access static burst parameters
 (i)   max burst size - Maximum burst size; (ii)  pcnt guard; (iii) offset
time; (iv)  burst timeout
 (v)   delta */
      static int& maxburstsize() { return maxburstsize__; }
      static  void  setMaxburstsize(  int  burstSize  )  {  maxburstsize__  =
burstSize; }
      static int& pcntguard() { return pcntguard__; }
      static void setPcntguard( int pcntguard ) { pcntguard__ = pcntguard;
}
      static double& offsettime(int j);
      static void setOffsettime( int j, double offsettime );
      static double& burst_timeout() { return burst_timeout__; }
      static  void  setBursttimeout(  double  timeout  )  {  burst_timeout__  =
timeout; }
      static double& delta() { return delta__; }
      static void setDelta( double delta ) { delta__ = delta; }
      int prio_; //GMG -- added priority class for this BM
      protected:
/* Send a data burst. Makes a call to IPKT agent's send burst method with
the default parameters stored there */
      void sendBurst();
/* Reset the burst queues -- method name may be misleading ... :) */
     void resetBurstParams();
/* TCL command interface. As of now, we just interact all commands via the
associated IPKTAgent/Integrated agent's command() method*/
     int command( int argc, const char*const* argv );
/* Internal use only - Obtains the number of hops between the source and
destination. Employs tcl support method for this, and assumes that the
routes have been computed */
     int nhops( nsaddr_t, nsaddr_t );
     // Holds the packets (only the headers)  while burst are still being
built
     Packet* Burstbuffer_[MAXBURSTSIZE];
     // ref to the burst time
     BurstTimer bt_;
     // ref to the parent iPKT agent
     IPKTAgent* a_;
     // the current burst size maintained by the bm object.
     int currburstsize_;
     // the total number of packet (TCP and ACKs) accumulated
     int npkts_ ;
     // the destination node id bound by this object.
     int destnodeid_;
     // maximum burst size
     static int maxburstsize__;
     // pcnt guard
```

```
     static  int pcntguard__;
     // the offset time to be set in the BHP
     static double offsettime__[nqos_classes]; /*GMG -- made offset time an
array of size equal to
     #QoS classes*/
     // the timeout values
     static double burst_timeout__;
     // delta  (BHP proc time)
     static double delta__;
     // global burst id
     static unsigned long burstid__;
     // generate burst ids
     static unsigned long getburstid()
     {
          if ( burstid__ == MAXBURSTID ) burstid__ = 0L;
             return burstid__++;
     }
};

/* Integrated (or) the IPKT Agent is the source and destination of BHP
(burst header packet or control packets) and BDP (Burst data packets or
data bursts).*/
class IPKTAgent : public Agent
{
    public:
        /* Constructs a new Integrated packet agent */
        IPKTAgent();
        /* Recv method -- receives a packet */
        void recv(Packet*, Handler *);
/* Compute the burst duration double bduration( long pktSize, long chbw );
*/ // GMG --            commented out; apparently not defined anywhere
       /* Send a burst - An independent method intended to be called by the
Burst Manager.
pBurst - The array of packets; bid - the burst identifier; nPkts - the
number of packets in the burst; burstsize - the pre-computed burst size;
destnode - the destination node for the data-burst; pcntgurard - the guard
band value; delta - the computation time*/
void sendBurst( Packet** pBurst, int bid, int npkts, int burstsize, int
destnode, double offsettime, int pcntguard, double delta, int priority );
 /* Deburstify a data burst into its indovidual components. This method is
intended to be called at the receiving edge node (or edge router) p - the
incoming data-burst*/
        void deBurst(Packet* p);
      /* Modified by Isi */
      static int attaching_table__[];
      /* Modified by Isi */
    protected:
        /* TCL interface cmd method */
        int  command( int argc, const char*const* argv );
/* The address of this node, i.e the node to which this IAGENT is already
attached to. Intended to be attached before hand */
        int address_;
//GMG - changed the BM_ array to a 2-dim array (over dest nodes and
priority classes)
/* The burst manager - The ipkt agent maintains one BM per edge node per
QoS class */
```

```
        BurstManager *BM_[nqos_classes];
¡/* Offset to access IPKT header */
        int    off_IPKT_;
// max index (number of sets of BMs (nqos_classes BMs per set))
        int    maxindx_;
//GMG -- added sequence number, for use with tracing the DB and BHP
        unsigned long int seqno_;
};
```

Una vez que se ha generado la ráfaga de datos, los objetos *BurstManager* invocan al método "*sendBurst*" de la clase *IPKTAgent* para: 1) crear e inicializar los paquetes BHP y ráfaga de datos con la dirección del nodo *edge* de destino; 2) establecer el tiempo de *offset*; y 3) insertar el "*payload*" en el paquete de la ráfaga de datos. Por otra parte, el método "*deBurst*" de la clase *IPKTAgent* divide la ráfaga de datos recibida en sus paquetes TCP constitutivos y los envía al *Edge Classifier* que los reenvía al *Integrated Agent* conectado al puerto de destino.

4.2.3 Router de *edge* electrónico

El router de *edge* actúa como una interfaz entre el dominio OBS y la red de acceso; y está constituido principalmente por: 1) una entidad de ensamblado/desensamblado de ráfagas; 2) una unidad de generación de BHPs y sus correspondientes ráfagas de datos; y 3) unidades de enrutamiento y planificación de canal.

4.2.3.1 Diseño del router de edge y sus componentes

Cuando un nodo recibe un paquete, su función es examinar sus campos, generalmente su dirección destino y a veces su dirección origen. En base a la información obtenida, se espera que el nodo mapee los valores a una interfaz de salida (o al siguiente receptor de este paquete); funcionalidad realizada en *ns-2* por el objeto *classifier*. En comparación con un enrutador electrónico convencional, el router de *edge* incluye la funcionalidad de generación y

planificación de ráfagas, que es implementada por el *Integrated Agent* y el nuevo clasificador *Edge Classifier*.

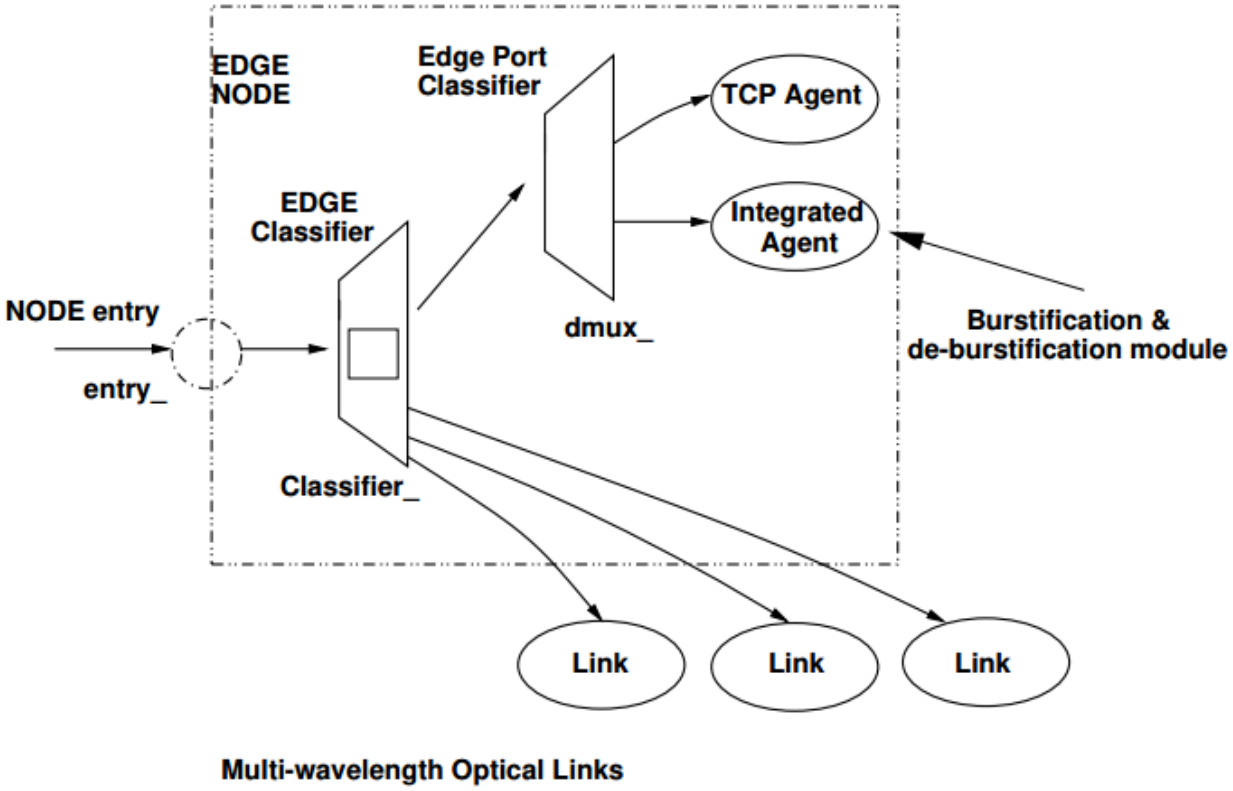


Figura 4.4 Estructura de un nodo de edge unicast [93]

La arquitectura del nodo (ver Figura 4.4) contiene los siguientes elementos:

Agent/IntegratedAgent, que realiza la mayor parte de las funcionalidades asociadas con el nodo de *edge*, incluyendo principalmente los procesos de ensamblado y desensamblado de ráfagas.

Classifier/BaseClassifier/EdgeClassifier, que actúa como un transmisor de paquetes BHP y de ráfagas de datos.

Classifier/OBSPort **(antes** ***Classifier/Port/EdgePortClassifier*****),** que divide el tráfico TCP (u otros provenientes de la red de acceso), transmitido desde los demás nodos y destinado así mismo. En este sentido, los paquetes con su dirección como la dirección destino, son enviados

al puerto destino; mientras que los paquetes no dirigidos a sí mismo son enviados al *Integrated Agent* para el proceso de ensamblado.

Scheduler_group, que mantiene las planificaciones reservadas sobre el enlace. Este objeto está embebido en la clase *Base Classifier*, que invoca el planificador para reservar canales de control para los paquetes BHPs y canales de datos para las ráfagas de datos.

Fiber Delay Lines (FDL), parte del planificador *FdlSchedule* que provee la capacidad del *buffer óptico* para resolver la contención en los enlaces de salida.

Una vez que el BHP y la ráfaga de datos han sido generados y enviados al objeto *EDGE Classifier*, este tiene suficiente información para planificar un canal apropiado para el BHP y la ráfaga de datos.

4.2.3.2 Diseño del Edge Classifier y aspectos de su implementación

En *ns-2*, un clasificador contiene una tabla de objetos de la simulación, indexados por un número de *slot*, y su función es validar la coincidencia de un paquete contra algunos criterios lógicos y recuperar una referencia a otro objeto de la simulación, determinando el número de *slot* asociado con un paquete recibido, para enviarlo al objeto referenciado por ese *slot* en particular [31]. El clasificador se encuentra asociado con un nodo, por lo cual no es llamado directamente a través del script OTcl, y su tarea es ejecutar las siguientes funciones:

- Clasificar los paquetes.
- Introducir el retardo[56] de entrada para los paquetes correspondientes a las ráfagas de datos, a fin de proveer el tiempo suficiente para que el BHP sea procesado y los canales de salida sean reservados para el BHP y la ráfaga de datos, y por consiguiente que se pueda configurar la matriz de conmutación óptica.
- Determinar el puerto de salida (o *slot* con respecto al clasificador) – enrutamiento.

[56] En lugar de introducir realmente un retardo, esto se maneja mediante la planificación de la ráfaga de datos y el BHP en un tiempo Δ después de que arriban al clasificador [93].

- Determinar y reservar los canales de salida para el BHP y la ráfaga de datos – asignación de longitud de onda y planificación.
- Recalcular el tiempo de *offset* entre el paquete del BHP y de la ráfaga de datos.

4.2.3.3 Clasificación de paquetes.

Aunque el *Integrated Agent* realiza el proceso de ensamblado y desensamblado de ráfagas, el *Edge Classifier* tiene que identificar los paquetes para estos propósitos. Es así que, todos los paquetes TCP/ACK (u otros provenientes de la red de acceso) se envían al *OBSPortClassifier* (antes *EdgePortClassifier*), el cual los reenvía a un agente sumidero TCP "*Sink*", si este es su destino o de lo contrario al *Integrated Agent*, para la el proceso de ensamblado.

Los paquetes OBS (tanto BHP y ráfaga de datos) que se generan en este nodo, se enrutan hacia el destino después de su planificación de canal. Los paquetes OBS cuyo destino es el nodo actual, se envían al *OBSPortClassifier* que los reenvía al *Integrated Agent* para el proceso de desensamblado.

```
//    if ( ( ch->ptype() == PT_TCP || ch->ptype() == PT_ACK )
      /* Modified by Isi */
      /* It is necessary to eleminate the source check */
    if ( (ch->ptype() != PT_IPKT) ) {             //GMG -- changed this to
       // && ( hdrip->saddr()== addr() ) ) {   // correspond to the
                                              // receipt of any data packet
(not
                                              //just TCP, ACK, or UDP)
      /* Modified by Isi */
// send it to Port Classifier which will redirect it to the Integrated
Agent or
}
else if( ( ch->ptype() == PT_IPKT ) && ( hdrip->daddr() == addr() ) ) {
   // will send it to PortClassifier
   } else {
      // schedule channel only if not destination
   } }
```

4.2.4 Planificación de canales

La planificación de los canales de datos se basa en el algoritmo del último canal no planificado disponible con relleno de huecos, LAUC-VF (*Latest Available Unscheduled Channel with Void Filling*), mientras que la planificación de los canales de control utiliza el mismo método pero sin relleno de huecos, por temas de eficiencia.

Las clases *Scheduler-group*, *Lauc-scheduler*, *Fdl-scheduler*, incluyen las implementaciones de los planificadores. El primero constituye la clase base del grupo de planificadores que mantiene los planificadores *Lauc Scheduler* (mantenido por defecto) y *Fdl Scheduler*; este último para incluir la opción de FDLs.

4.2.4.1 Arquitectura e inicialización

El conjunto de longitudes de onda disponibles en un enlace utiliza un grupo de planificadores de la clase *Scheduler_group*, donde el planificador mantenido por defecto es de la clase *LaucScheduler*, que administra la planificación de las transmisiones sobre un enlace.

Cada nodo de *edge* cuenta con un grupo de planificadores de la clase *Scheduler_group*, mantenido como un arreglo en la clase *Base Classifier*, cuyo número es igual al grado de libertad del nodo, es decir, existe un grupo de planificadores por cada enlace[57] conectado, identificado unívocamente por un par de direcciones origen-destino que se establecen durante la inicialización de la topología, tal y como se indica a continuación en el procedimiento *createDuplexFiberLink*.

[57] Debido a que los enlaces bidireccionales en *ns-2* se tratan como dos enlaces independientes, los planificadores se crean en ambos extremos [93].

```
Simulator instproc createDuplexFiberLink { node1 node2 bwpc delay ncc ndc
maxch } {
    $self instvar link_

    $self obs_duplex-FiberLink $node1 $node2 $bwpc $delay Null $maxch

    set id1 [$node1 id]
    set id2 [$node2 id]
    if [info exists link_($id1:$id2)] {
      [$node1 set classifier_] install-scheduler $id2 $ncc $ndc $maxch $bwpc
      puts "Adding schedulers between $node1 and $node2"
    }
    if [info exists link_($id2:$id1)] {
      [$node2 set classifier_] install-scheduler $id1 $ncc $ndc $maxch $bwpc
      puts "Adding schedulers between $node2 and $node1"
    }
}
// command method
int BaseClassifier::command( int argc, const char*const* argv )
{
Tcl& tcl = Tcl::instance(); //GMG -- added because referred to in drop
target
    if( argc == 2 )   {
        if( strcmp( argv[1], "display-address" ) == 0 ) {
            char s[100];
            sprintf( s, "Address is %d", address_ );
            //Debug::debug( s );
            return (TCL_OK);
        }
.
.
.
}
   else if( argc == 7 ) {
        /*   $classifier   install-scheduler   $destination   $ncc   $ndc
$maxChannels $bwpc*/
        if( strcmp( argv[1], "install-scheduler" ) == 0 ) {
            LaucScheduler *lsc = new LaucScheduler();
            lsc->alloc( (u_int)atoi( argv[3] ), (u_int)atoi( argv[4] ),
(u_int)atoi( argv[5] ), this ); //GMG –
           added 'this' pointer to base classifier
            (*lsc).destNodeId() = (u_int)atoi( argv[2] );
            //lsc->chbw() = atoi( argv[6] );
            lsc->chbw() = atof( argv[6] ); // swkim - 2004/08/10 changed
atoi to atof
            sg.install( lsc );
            char str[200];
            sprintf( str, "Installed a LaucScheduler at %d for dest: %d",
address_, lsc->destNodeId() );
            //Debug::debug( str );
            // sg.printInfo();
            return TCL_OK;
        }
    }
```

Cuando un BHP, que contiene la dirección final del nodo destino llega al *Edge Classifier*, se debe determinar el siguiente salto en la ruta para seleccionar el planificador correcto, buscando secuencialmente el arreglo de planificadores para encontrar el que tiene el siguiente salto como su ID del nodo destino, y una vez obtenido, se invocan los métodos *"schedControl"* y *"schedData"* para la planificación de los canales de control y ráfaga de datos, respectivamente. La definición del planificador por defecto se presenta a continuación:

```
class LaucScheduler
{
    public:
        /* do nothing constructor */
        LaucScheduler(){}
        /* Construct a new Lauc Scheduler object with the provided control
and data */
        LaucScheduler( u_int ncc, u_int ndc, u_int maxChannels );
        /* alloc space and assign ncc ndc and maxChannels */
        virtual void alloc( u_int ncc, u_int ndc, u_int maxChannels,
BaseClassifier *parent );
        /* Schedule a control channel at the proposed schedule time and
duration */
        virtual Schedule schedControl( double schedTime, double schedDur );
        /* Schedule a data channel at the proposed schedule time and
duration */
        virtual Schedule schedData( double schedTime, double schedDur, int
&fdl_count );
        /// Accesor and modifiers
        u_int& ncc() { return ncc_; }
        u_int& ndc() { return ndc_; }
        u_int& maxChannels() { return maxChannels_; }
        // return the destination Node id
        u_int& destNodeId() { return destNodeId_; }
        // the per channel bw
        //u_int &chbw() { return chbw_; }
        double &chbw() { return chbw_; }
        // Returns the packets duration
        virtual double duration( u_int pktsize );
        /* Diagnostic method */
        virtual void printChInfo( u_int channel );
    protected:
        /* Number of control data and total channels present per-fiber-link
*/
        u_int ncc_, ndc_, maxChannels_;
        /* Unscheduled time, startTime and the endTime */
        double *unschTime_, *startTime_, *endTime_;
        /* per channel bandwidth */
        //u_int chbw_;
        double chbw_;
        /* destination node id of the lauc-scheduler */
        u_int destNodeId_;
```

```
        /* GMG -- added pointer to base classifier whose node this Lauc
scheduler is part of (needed to
        access FDL scheduler)*/
        BaseClassifier *bc_;
        /* search for an appropriate data-schedule */
        Schedule search( double schedTime, double schedDur, int &count );
        /* update channel information */
        void update( u_int channel, double schedTime, double schedDur );
};
```

El planificador mantiene un tiempo de inicio y fin de los últimos huecos de cada canal. Si el tiempo de inicio es mayor que el tiempo actual, se trata de un canal completamente libre. Los canales se buscan secuencialmente hasta encontrar uno que resultará ser el hueco más pequeño después de planificar el BHP y la duración de la ráfaga de datos; y una vez que se ha encontrado un canal libre, se actualiza la tabla de búsqueda. El método de búsqueda *"search"* se presenta a continuación:

```
// search the scheduler and the voids for an appropriate schedule
Schedule LaucScheduler::search( double schedTime, double schedDur, int
&count ) {
    Schedule result;
    int max, fdl;  //GMG -- added local variables
    double FDLdelay = bc_->FS_.fdl_delay_;  //GMG -- added local variable
for FDL delay
    double   diffTime = HUGE_VAL;
    double time0 = HUGE_VAL;
    int j0;
    int savecount;  //needed to restore count if can't schedule FDL
    .
    .
    .
    .
    for( u_int i = ncc_; i < maxChannels_; i++ )
       {
           // try to schedule in a void
           if( schedTime >= startTime_[i] )
           if( ( endTime_[i] - schedTime ) >= schedDur )
           if( ( schedTime - startTime_[i] ) < diffTime )
           {
               diffTime = schedTime - startTime_[i];
               result.channel() = i;
               result.startTime() = schedTime;
           }
           // try to schedule after the void
           if( schedTime >= unschTime_[i] )
           if( ( schedTime - unschTime_[i] ) < diffTime )
           {
```

```
            diffTime = schedTime - unschTime_[i];
            result.channel() = i;
            result.startTime() = schedTime;
        }
    }
```

El método *"search"* se invoca por primera vez para programar la ráfaga de datos y devuelve un objeto de la clase *Schedule* que almacena el ID de canal planificado y el tiempo de inicio. Dicho objeto que es devuelto por la función *"schedData"*, se utiliza luego para programar el BHP al mismo tiempo que la ráfaga de datos, y se inserta en una tabla *hash* indexada según el ID único de ráfaga generado para cada una. Finalmente, cuando arriba la ráfaga de datos correspondiente, se recupera este objeto desde dicha tabla.

4.2.5 Líneas de retardo de fibra FDLs (*Fiber Delay Lines*)

Se utilizan para resolver la contención en los canales de salida cuando no existen canales libres cuando el BHP solicita reservaciones sobre ellos. En tal caso, la ráfaga de datos o el BHP pueden ser retrasados desviándolos hacia *buffers* ópticos provistos por FDLs. Cada canal tiene un conjunto de unidades de retardo que corresponden a valores discretos de B WDM FDLs, existiendo por tanto un límite en el retardo que se puede introducir, con el i-ésimo FDL capaz de establecer un retardo dentro del intervalo Q_i, $1 \leq i \leq B$, donde $Q_1 < Q_2 < ... < Q_B$.

En el caso de que todos los canales estén planificados en el tiempo requerido, el método *"search"* de la subsección previa trata de encontrar un canal no planificado introduciendo retardos Q_1, Q_2, ... < Q_B en secuencia, hasta encontrar un canal libre. Si no existen canales disponibles incluso después de insertar el máximo retardo posible Q_B, simplemente se descartan tanto el BHP como la ráfaga de datos. Dicho concepto se ha incluido modificando el método de búsqueda anterior, el cual se indica a continuación:

```
/ search the scheduler and the voids for an appropriate schedule
FdlSchedule FdlScheduler::search( double schedTime, double schedDur )
{
```

```
    FdlSchedule result;
    double   diffTime = HUGE_VAL;

    for( u_int i = 0; i < nfdl_; i++ ) {
        // try to schedule in a void
        if( schedTime >= startTime_[i] )
           if( ( endTime_[i] - schedTime ) >= schedDur )
              if( ( schedTime - startTime_[i] ) < diffTime )
              {
                 diffTime = schedTime - startTime_[i];
                 result.fdl() = i;
                 result.startTime() = schedTime;
              }
        // try to schedule after the void
        if( schedTime >= unschTime_[i] )
           if( ( schedTime - unschTime_[i] ) < diffTime )
           {
              diffTime = schedTime - unschTime_[i];
              result.fdl() = i;
              result.startTime() = schedTime;
           }
    }
    return (result);
}
```

4.2.6 Ajuste del tiempo de *offset*

El tiempo de *offset* representa el intervalo entre la llegada del BHP y la ráfaga de datos asociada a un nodo, el mismo que no es constante ya que la configuración del conmutador, la planificación y los retardos para resolver las contenciones varían en cada nodo.

Dependiendo de la duración que puedan tomar dichas operaciones, el tiempo de *offset* se actualiza en el BHP, antes de salir del nodo. A continuación se muestran los cálculos que están implícitos en la implementación, y cuyo tiempo de *offset* es utilizado en el siguiente salto para cálculos similares:

X = Switch Configuration time.

O_t = *Offset* time.

$BHP_{request\_time} = BHP_{arrival\_time} + X$

$Burst_{request\_time} = BHP_{arrival\_time} + O_t$ + Input delay

$BHP_{start} = BHP_{request\_time} + BHP_{FDLdelay}$

$Burst_{start} = Burst_{request\_time} + Burst_{FDLdelay}$

New $O_t = Burst_{start} - BHP_{start}$

4.2.7 Diseño del *OBSPortClassifier* (antes *EdgePortClassifier*) y aspectos de su implementación

El clasificador de direcciones clasifica los paquetes en base a la dirección destino y los reenvía al enlace apropiado. Si un nodo es el destino de un paquete, lo envía al *OBSPortClassifier*, que lee el puerto destino y reenvía el paquete al agente conectado a dicho puerto. El clasificador de puertos en el nodo de *edge*, fue modificado ya que maneja también paquetes TCP/ACK (u otros provenientes de la red de acceso) no destinados a sí mismo.

Si el nodo actual es el destino de un paquete TCP/ACK (u otros provenientes de la red de acceso), se lo reenvía al puerto destino al que está conectado un agente sumidero TCP "*Sink*". Si este es el nodo origen para ese paquete, se lo envía al puerto al que está conectado el *Integrated Agent* para el proceso de ensamblado.

Sólo los paquetes de ráfaga de datos cuyo destino corresponde al nodo actual, se envían mediante el clasificador de direcciones hacia el *OBSPortClassifier*, y no directamente al *Integrated Agent* para el proceso de desensamblado; la implementación de estas operaciones se presentan a continuación, junto con las rutas que siguen los paquetes TCP y las ráfagas de datos dentro de un nodo de *edge* de ingreso y un nodo de *edge* de egreso, ilustradas en las Figuras 4.5 y 4.6.

```
void OBSPortClassifier::recv( Packet *p, Handler * ) {
char s[200];
    if( ( iAgent_ == NULL ) || ( p == NULL ) ){
        //Debug::debug( __FILE__, __LINE__, "Error Iagent is unitialized or
provided packet is NULL" );
        return;
    }
.
.
```

```
if( ( ch->ptype() == PT_IPKT ) && ( Address::instance().hier_addr(des_addr,
1) == address_ ) )
        // Send to Integrated Agent Port
        iAgent_->recv( p );
    else {
      /* Modified by Isi */
        //if( src_addr == address_ )  {
        //    iAgent_->recv( p );
        //}
        //else
      /* Modified  by Isi */
      if( Address::instance().hier_addr(des_addr, 1) == address_ ) {
            //Debug::debug(  "Portclassifier:  received  a  tcp  (or)  ack
packet" );
            slot_[iph->dport()]->recv( p ); // Send to destination port
        }
        else {
            //Debug::debug(     "Edge-port-classifier:     error     unknown
destination" );
            /* Modified by Isi */
          iAgent_->recv( p ); // Send to Integrated Agent Port
            /* Modified by Isi */
            // return;
        }
    }
```

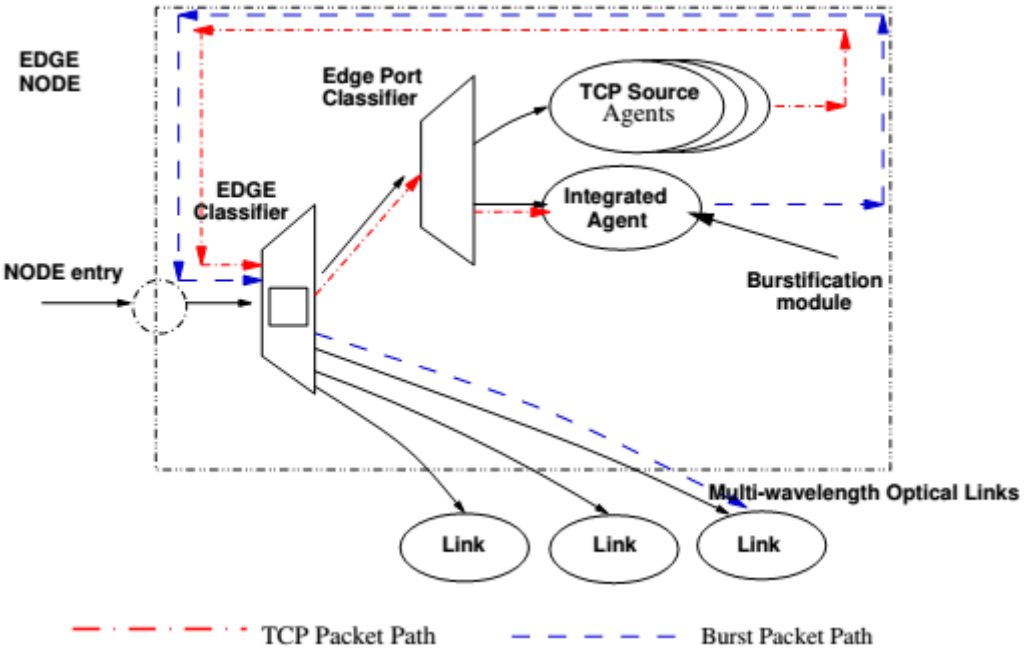


Figura 4.5 Ruta de los paquetes TCP y ráfagas al ingreso del nodo de edge [93]

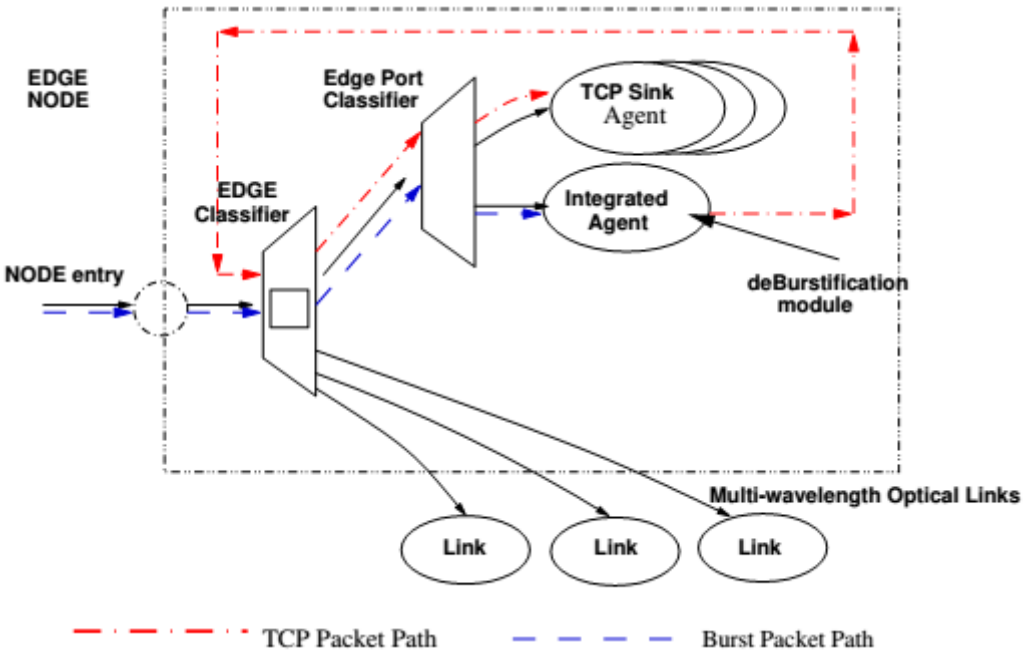


Figura 4.6 Ruta de los paquetes TCP y ráfagas a la salida del nodo de edge [93]

4.2.8 Router óptico de *core*

Está constituido principalmente por: 1) una unidad de control electrónico SCU (*Switch Control Unit*) que procesa los paquetes de control BHPs para llevar a cabo las tareas de enrutamiento y planificación de canales; 2) unidades de enrutamiento y planificación de canales que realizan este tipo de tareas; 3) una matriz de conmutación óptica OSM (*Optical Switching Matrix*) que se configura previamente para permitir que la ráfaga entrante atraviese el nodo con un retardo mínimo; 4) líneas de retardo FDL (*Fiber Delay Lines*) que trabajan en conjunto con la unidad de planificación con el fin de introducir un retardo para resolver la contención en los canales de salida, en caso de no encontrar uno disponible en un determinado instante.

El router de *core* es similar al router de *edge*, y a diferencia de este, su función es planificar únicamente las ráfagas que pasan a través de el, por lo cual no requiere conectar agentes y el *Port Classifier* no necesita modificarse por la misma razón; el único cambio requerido es en el clasificador de direcciones para introducir la planificación de canales, que utiliza el mismo

grupo de planificadores de la clase *Scheduler_group*, manteniendo por defecto el de la clase *LaucScheduler*.

4.2.9 Simulación de la capa de transporte

Los componentes utilizados como parte de la capa de transporte son principalmente agentes TCP o UDP que están conectados a los nodos de acceso que se interconectan a los nodos de *edge*, a través de interfaces hacia la red de acceso. Puesto que OBS-ns simula principalmente el desempeño del *backbone* de una red de *core* óptica, a través del cual podrían cruzar miles de conexiones TCP y/o UDP, desde el punto de vista de la red OBS esto se visualiza como tráfico entre nodos de *edge* generado por sus respectivas interfaces, que actúan como fuentes y sumideros de tráfico, donde los agentes conectados a dichos nodos emulan dichas interfaces [93].

Para la generación de tráfico, se utilizó el generador *"my-expoo"* (versión modificada del generador *Expoonential* de *ns-2*) implementado para el simulador OBS-ns original, y que fue adaptado para modelar un flujo continuo de datos con una alta tasa de tráfico, debido a que los generadores de tráfico implementados en *ns-2* son periódicos con ciclos de flujos de tráfico seguidos por tiempos de inactividad, los mismos que no son apropiados para la evaluación de la arquitectura de OBS.

4.2.10 Recolección de los datos de la simulación

La clase *StatCollector* recolecta estadísticas de la simulación y envía los datos a un archivo al final de su ejecución, pudiéndose extender fácilmente agregando más etiquetas a la lista de datos predefinida *baseInfoList[]*.

Utilizando la clase *StatEntry*, el objeto *StatCollector* define una interfaz simple para recolectar varios tipos de datos durante la ejecución de la simulación, que opcionalmente podrían ser

escritos en un archivo. La interfaz definida como genérica a través del uso de etiquetas, que indican el tipo de dato junto con un valor *string* y uno flotante, se invoca cada vez que se recolecta un dato utilizando los métodos *"addEntry"*, *"getStatEntry"* y *"updateEntry"*; cuya definición se muestra a continuación:

```
typedef enum {
    TCPSND,     // TCP Send
    TCPBYTESSND, // TCP bytes Send  -- added by GMG
    TCPRCV,     // TCP recv
.
.
.
} statType;

class StatEntry
{
    public:
        /* Construct a new StatEntry object with the provided information
         * statType t - represents the basic statistical type
         * s - the informational string
         * v - the value which will be stored as a double data-type */
        explicit StatEntry( statType t = OTHER, string s = "", double v =
0.0 ) {
                type_ = t;
                infoStr_ = s;
                value_ = v;
        }
        /* Returns a reference to the statType */
        statType& type() { return type_; }
        /* Returns a reference to the informational string */
        string& infoStr() { return infoStr_; }
        /* Returns a reference to the recorded value */
        double& value() { return value_; }
        /* Add an Entry to the list of entries maintained */
        static bool addStatEntry( StatEntry se );
        /* Add an Entry to the list of entries maintained */
        static bool addStatEntry( statType st, string s, double value );
        /* Add an Entry to the list of entries maintained */
        static bool addStatEntry( string s, double value );
        /* Get an Entry from the list of entries */
        static StatEntry& getStatEntry( string keyStr );
        /* Diagnostic display -content method */
        static void displayEntry( StatEntry se );
        /* Diagnostic display-all method */
        static void displayEntry();
        /* Base info list is essentially a string representation of the 11
base statistics */
        static string baseInfoList[44];
    protected:
        /* stat type */
```

```
        statType    type_;
        /* informational string */
        string      infoStr_;
        /* Actual value stored as a double */
        double      value_;
        /* Vector of different statistical Entries */
        static map<string, StatEntry> mapList__;
};

/* The StatCollector object provides an interface to the StatEntry class
both via Tcl and via C++. Its like a common interface to add and remove
simulation Entries in C++ and Tcl */
class StatCollector : public NsObject
{
    public:
        /* Returns back the reference to this object */
        static StatCollector& instance();
        /*  Returns  back  a  pointer  of  the  instance..  Need  for  Tcl  via
instantiation */
        static StatCollector* getInstance();
       /* add an entry to the stat table */
        virtual void addEntry( statType type, string entry, double value );
        /* add a custom entry to the stat table */
        virtual void addEntry( string entry, double value );
        /* Interface methods to the StatEntry object */
        virtual void updateEntry( string entry, double value );
        /* Get the value for the specific entry */
        virtual double getValue( string entry );
    protected:
        /* Constructs a new StatCollector object */
        StatCollector();
        /* Interface recv method */
        virtual void recv( Packet *p, Handler *h = 0 )  {}
        /* tcl command */
        virtual int command( int argc, const char*const* argv );
        /* self referential static instance */
        static StatCollector *instance__;
};
```

4.3 Cambios realizados en el código fuente de OBS-ns

Particularmente para el desarrollo del ambiente de simulación, se instaló la versión 2.33 del simulador *ns-2*, en un computador portátil *core* i5 con 8G de memoria RAM bajo el sistema operativo Ubuntu 10.04, junto con el módulo obs-0.9a, el cual requirió de ciertos cambios para su compatibilidad con el compilador gcc-4.1/g++-4.1, necesario para esta versión del simulador, así como también para la adición de ciertas funcionalidades no incluidas dentro de la arquitectura original del simulador, requeridas para la evaluación de este proyecto de tesis.

Las modificaciones antes mencionados, realizadas sobre el código fuente del simulador *ns-2* y el módulo obs-0.9a se señalan brevemente a continuación:

4.3.1 Cambios realizados en el código C++

- **Clases *BurstManager*, *IPKTAgent* y *StatCollector* definidas en los archivos integrated_agent.{h,cc} y stat-collector.{cc,h} respectivamente**
  - Impresión en pantalla de las ráfagas ensambladas con su respectivo tiempo de simulación, ID, número de paquetes y tamaño.
  - Opción a ser seleccionada por parte del usuario desde el script Otcl, para habilitar o deshabilitar la funcionalidad del ensamblador de ráfagas, para un determinado tipo de paquete (ACK por defecto); esto para adaptar la simulación al modelo descrito en la subsección 3.4.1, e ilustrado en la Figura 3.12, para lo cual se implementó la función denominada *"burstless"* dentro de la clase *BurstManager* que recibe uno o dos parámetros, con el valor de cero (0) y el tipo de paquete (p.ej. PT_ACK) para deshabilitar el ensamblador, y el valor de uno (1) para habilitarlo, respectivamente.
  - Contabilización de las ráfagas enviadas y recibidas sobre un enlace OBS, es decir, desde un nodo de *edge* de ingreso hacia un nodo de *edge* de egreso, dado que las estadísticas recolectadas por la clase *StatCollector* incluyen la información consolidada de todos agentes integrados en los diferentes nodos del dominio OBS y no información específica sobre un determinado enlace OBS entre un par de nodos de *edge*. Por esta razón se agregaron dos entradas de estado, *BURSTFROMTORCV* y *BURSTFROMTOSND*, en lista de la base de información de estadísticas *baseInfoList[]*.
  - Rastreo de los valores correspondientes al *timeout* de retransmisión de TCP (RTO) con el fin de poder calcular las estimaciones teóricas del *throughput* de TCP, definidas en las ecuaciones presentadas en la subsección 3.4.2.

- **Clases *EdgeClassifier* definida en los archivos classifier-base.{cc,h}**
  - Modificación del enrutamiento jerárquico de la función *"void EdgeClassifier::recv(Packet *pkt, Handler *h = 0)"* para obtener una referencia hacia el nodo de acceso con dirección X.0.Y al cual va destinado el tráfico, en lugar del comportamiento por defecto que consiste en enrutar los paquetes que salen del nodo *edge* de egreso, hacia la BS (*Base Station*) conectada a dicho nodo con dirección X.0.1.

4.3.2 Cambios efectuados en el código Otcl

- **Script ns-obs-lib-hierarchical.tcl**
  - Agregación del procedimiento *"Simulator instproc createSimplexFiberLink"* para crear un enlace óptico WDM unidireccional que es necesario para adaptar un módulo de error, a fin de simular la probabilidad de pérdidas de ráfagas requerida para las simulaciones a evaluar.

4.4 Modelos de simulación y resultados obtenidos

Los escenarios implementados para la validación de los modelos analíticos descritos en el capítulo anterior, se basan en el esquema mostrado en la Figura 3.12, del mismo que se derivan: 1) escenario simplificado con un sólo flujo TCP sin considerar el proceso de ensamblado de ráfagas para los paquetes ACK para la evaluación de TCP Reno; y 2) escenario básico similar al anterior, con la diferencia de que no elimina el proceso de ensamblado para la evaluación de TCP Reno, Newreno y SACK; los mismos que se describen con mayor detalle a continuación:

4.4.1 Escenario simplificado

Escenario desarrollado para evaluar el desempeño de TCP Reno y comparar sus resultados contra el modelo analítico basado en flujos TCP lentos y rápidos, que incluye un solo emisor TCP y un receptor TCP conectados a un nodo OBS de *edge* cada uno, por medio de un enlace de acceso sin pérdidas, con los nodos OBS conectados a través de un enlace DWDM que experimenta una probabilidad de pérdida p_b con distribución de Bernoulli, únicamente en el sentido desde el origen TCP (servidor) hacia el destino TCP (cliente) y sin proceso de ensamblado de ráfagas en el sentido opuesto.

A continuación se presenta el modelo a simular bajo este escenario (ver Figura 4.7), así como también los parámetros de configuración utilizados que se definen en la Tabla 4.1.

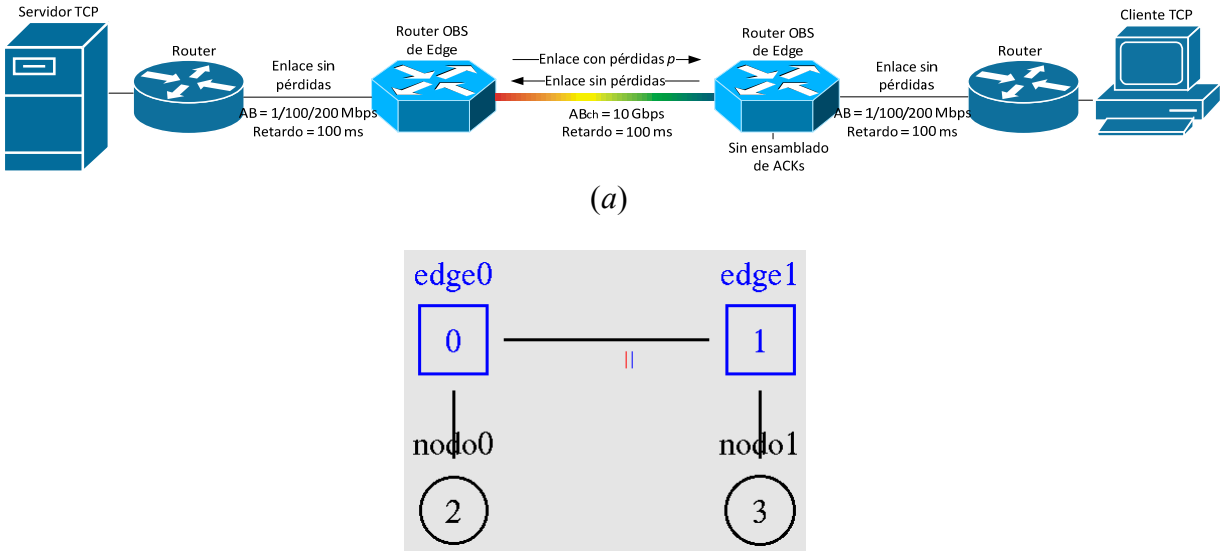


Figura 4.7 Modelo simplificado para la evaluación de TCP Reno; a) diagrama esquemático; y b) topología creada en ns-2

Tabla 4.1 Parámetros para la simulación

Parámetro	Valor
Ancho de banda red de acceso	1M, 100M y 200M
Retardo red de acceso	100 ms
Tamaño máximo del segmento	552 bytes
Ventana máxima de transmisión	70656 bytes (128 segmentos)
Versión de TCP	Reno
Tipo de tráfico	Exponencial
Número de canales de control	2
Número de canales de datos	8
Ancho de banda por canal	1 Gbps
Tiempo de propagación	100 ms
Tiempo de ensamblado	3 ms
Tamaño máximo de ráfaga	100000 bytes
Probabilidad de pérdida de ráfagas	10^{-4} – 0.2
Tiempo de simulación	1000 seg por muestra

A continuación se presentan los resultados obtenidos al simular este escenario.

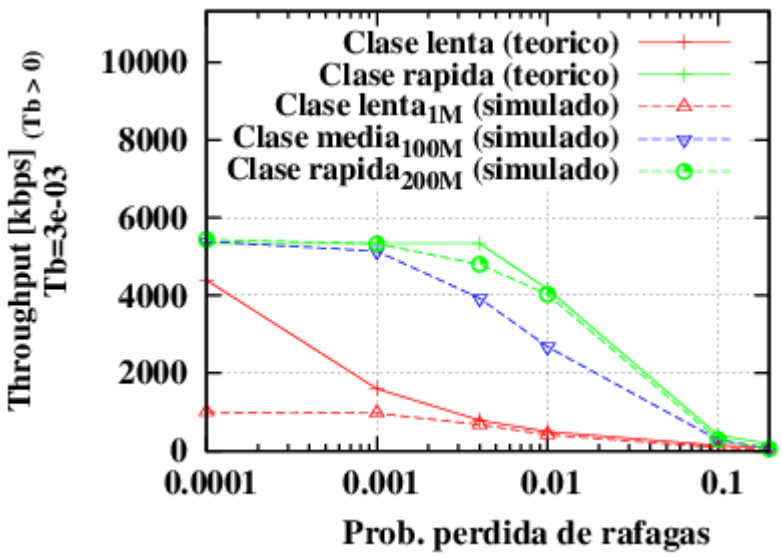


Figura 4.8 Throughput de TCP Reno vs probabilidad de pérdida de ráfagas para fuentes de 1 Mbps (clase lenta); 100 Mbps (clase media); y 200 Mbps (clase rápida), con RTT=603 ms, W_{max}=128, L=512 bytes, y T_b=3 ms; sin ACK retardado

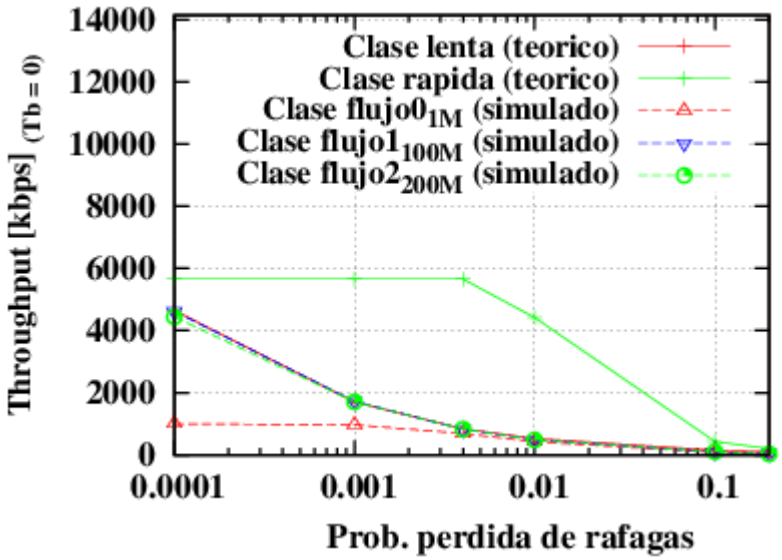


Figura 4.9 Throughput de TCP Reno vs probabilidad de pérdida de ráfagas para fuentes de 1 Mbps (clase lenta); 100 Mbps (clase media); y 200Mbps (clase rápida), con RTT=603 ms, W_{max}=128, L=512 bytes, y T_b=0 ms; sin ACK retardado

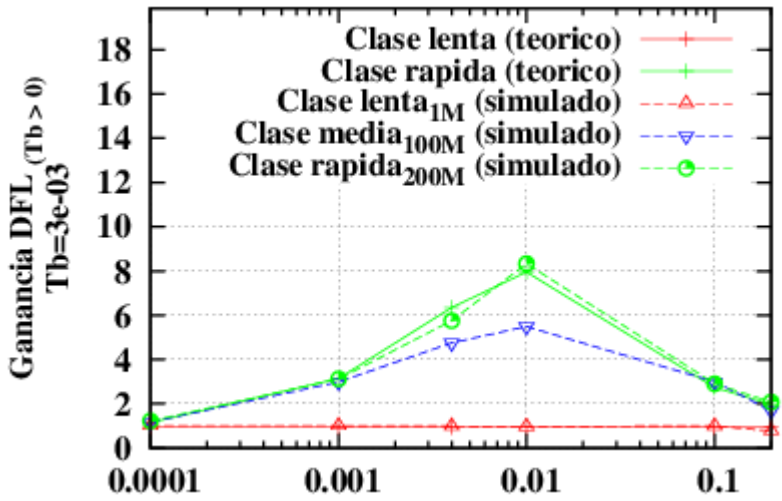


Figura 4.10 Ganancia DFL de TCP Reno vs probabilidad de pérdida de ráfagas para fuentes de 1 Mbps (clase lenta); 100 Mbps (clase media); y 200Mbps (clase rápida), con RTT=603 ms, W_{max}=128, L=512 bytes y T_b=3 ms; sin ACK retardado

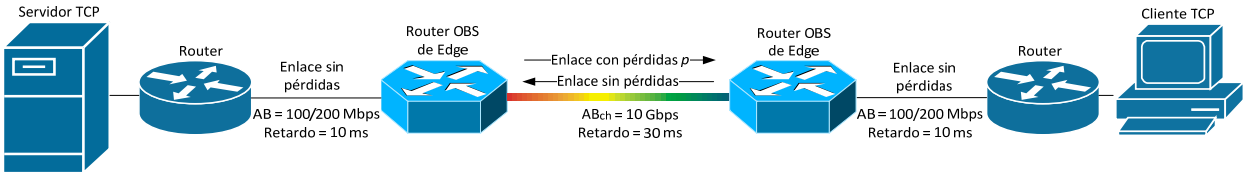
El tiempo de simulación para este caso fue de aproximadamente 3 horas, donde los resultados muestran que el *throughput* de TCP Reno sobre redes OBS varía entre dos funciones límite que definen los valores mínimo y máximo, en base al modelo analítico de fuentes lentas y fuentes rápidas, respectivamente, y en función de la probabilidad de pérdida de ráfagas p_b y el tiempo de ensamblado de ráfagas. En los resultados obtenidos se puede evidenciar que el modelo analítico se aproxima en gran medida al comportamiento obtenido por simulación para fuentes lentas y rápidas, y que para fuentes medias, el *throughput* se encuentra entre estos dos límites. Por otro lado, se puede observar también que al comparar el comportamiento de TCP Reno sobre una red de conmutación de paquetes (ver Figura 4.9) y una red OBS (para una misma condición de probabilidad de pérdida), en esta última se experimenta un efecto de ganancia en el *throughput* (ver Figura 4.10) que varía en función de la cantidad de segmentos agregados en una ráfaga y la probabilidad de pérdida experimentada, llegando a un valor máximo cuando p_b se aproxima a $1/W_m$, que para este caso estaría en un valor cercano a $p_b \approx 0.008$, y que se podría entender como el resultado de un mayor número de segmentos transmitidos, por la propia naturaleza de una red OBS al ensamblar datos en una entidad de mayor tamaño, en comparación con una red tradicional de conmutación de paquetes bajo la misma condición de

pérdidas[58]. Debido a que este escenario pretende evaluar el comportamiento de TCP Reno en función del modelo simplificado basado en la Figura 3.12, que no sería realista por la ausencia del ensamblador de ráfagas en el sentido de la transmisión de ACKs, no se profundiza más en el análisis de este modelo.

4.4.2 Escenario básico

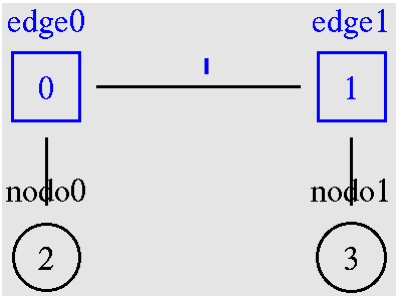
Escenario desarrollado para evaluar el desempeño de las variantes de TCP basadas en pérdidas y comparar sus resultados contra los modelos analíticos basados en la teoría de renovación de Markov, tanto el caso de flujos lentos y rápidos como para fuentes de velocidad genérica, que al igual que el caso anterior incluye un solo emisor TCP y un receptor TCP conectados a un nodo OBS de *edge* cada uno, por medio de un enlace de acceso sin pérdidas, con los nodos OBS conectados a través de un enlace DWDM que experimenta una probabilidad de pérdida p_b con distribución de Bernoulli únicamente en el sentido desde el origen TCP (servidor) hacia el destino TCP (cliente) y sin eliminar el proceso de ensamblado.

A continuación se presenta el modelo a simular bajo este escenario (ver Figura 4.11), así como también los parámetros de configuración utilizados que se definen en la Tabla 4.2.



(*a*)

[58] Obviamente la ganancia DFL presenta su nivel más alto en una determinada condición de probabilidad de pérdida que no causa mayor reducción en cuanto al número de segmentos transmitidos, con relación al impacto ocasionado al disminuir significativamente el número de segmentos transmitidos en una red de conmutación de paquetes bajo la misma condición.



(b)

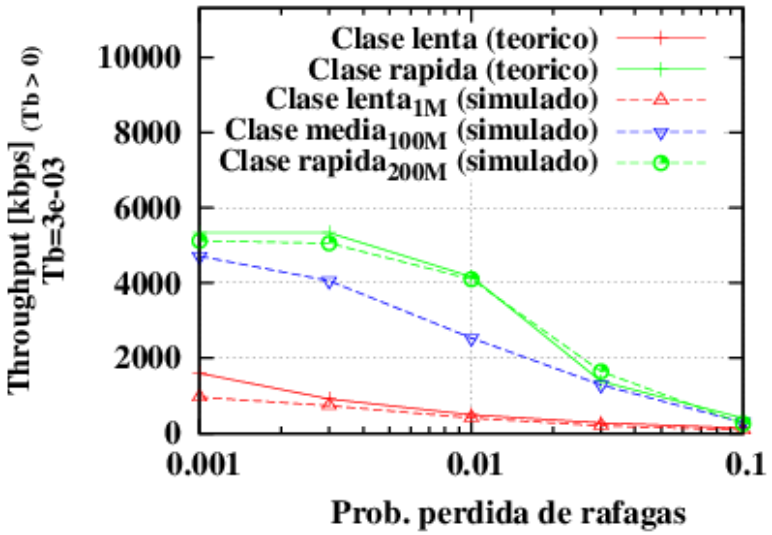
Figura 4.11 Modelo básico para la evaluación de TCP Reno, Newreno y SACK; a) diagrama esquemático; y b) topología creada en ns-2

Tabla 4.2 Parámetros para la simulación

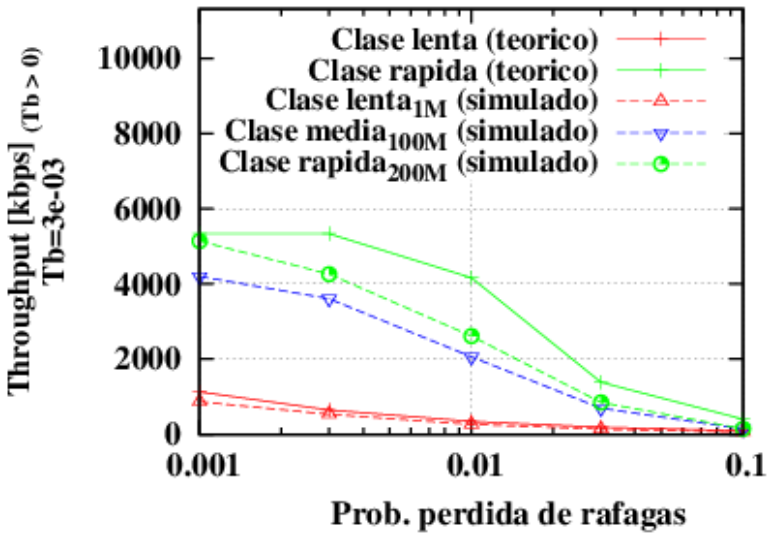
Parámetro	**Valor**
Ancho de banda red de acceso	1M, 100M y 200M
Retardo red de acceso	10 ms
Tamaño máximo del segmento	552 bytes
Ventana máxima de transmisión	70656 bytes (128 segmentos)
Versión de TCP	Reno, Newreno y Sack
Tipo de tráfico	Exponencial
Número de canales de control	2
Número de canales de datos	2 y 8
Ancho de banda por canal	1 Gbps
Tiempo de propagación	30 ms
Tiempo de ensamblado	0.1 - 100 ms
Tamaño máximo de ráfaga	100000 bytes
Probabilidad de pérdida de ráfagas	10^{-3} – 10^{-1}
Tiempo de simulación	1000 seg por muestra

4.4.2.1 *Análisis del modelo de flujos lentos y rápidos*

El tiempo de simulación para la evaluación de TCP Reno utilizando el modelo de procesos regenerativos de Markov basado en fuentes lentas y rápidas, fue de aproximadamente 8 horas (sin ACK retardado) y 9 horas (con ACK retardado) con los siguientes resultados.

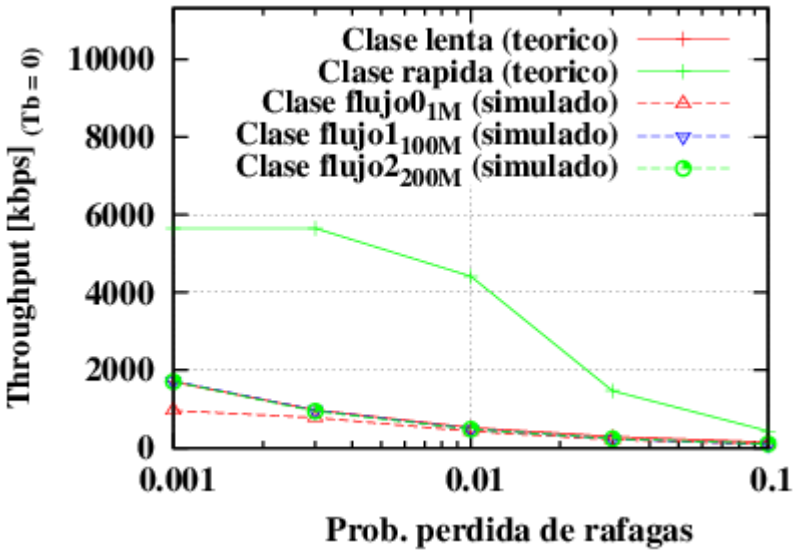


(*a*)

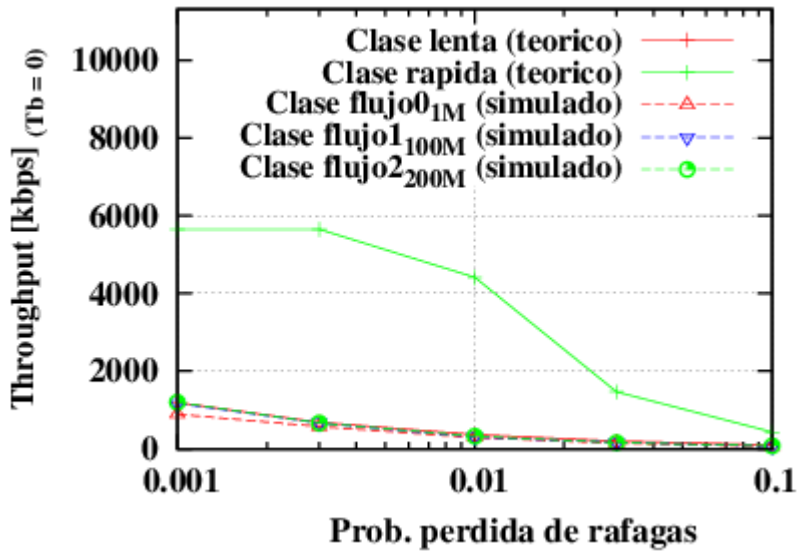


(*b*)

Figura 4.12 Throughput de TCP Reno vs probabilidad de pérdida de ráfagas para fuentes de 1 Mbps (clase lenta); 100 Mbps (clase media); y 200Mbps (clase rápida), con RTT=106 ms, W_{max}=128, L=512 bytes, y T_b=3 ms; sin ACK retardado (a) y con ACK retardado (b)

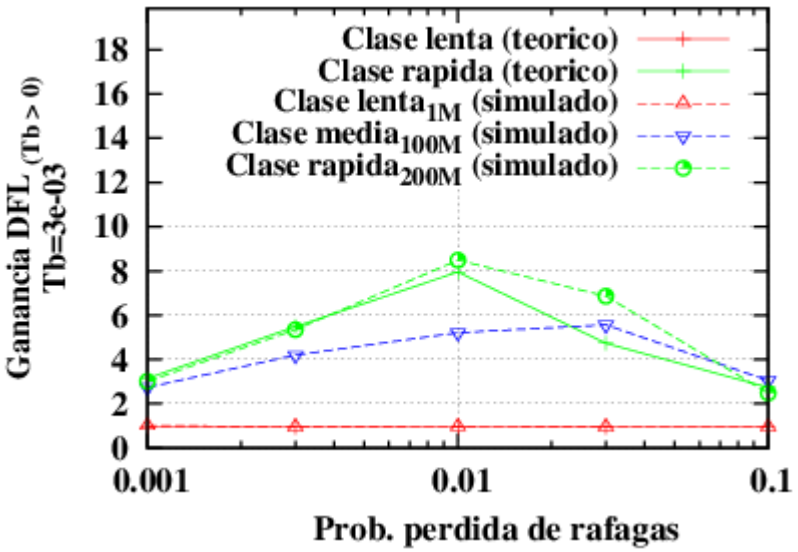


(*a*)

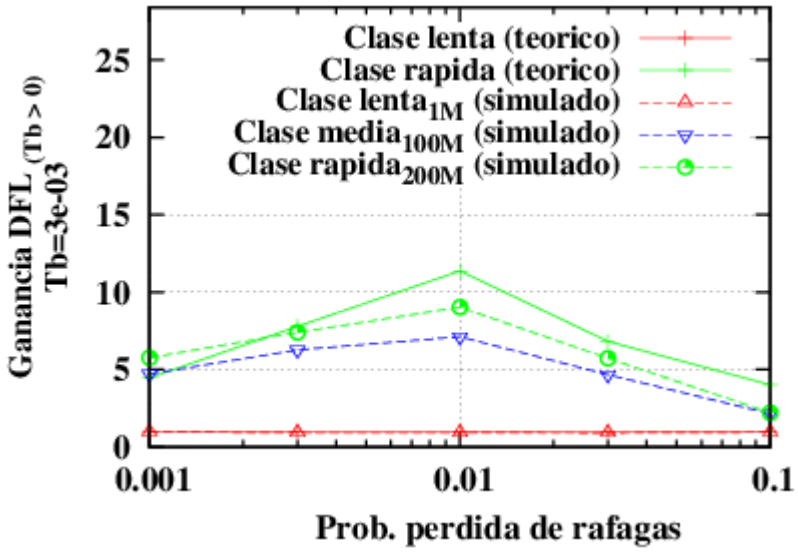


(*b*)

Figura 4.13 Throughput de TCP Reno vs probabilidad de pérdida de ráfagas para fuentes de 1 Mbps (clase lenta); 100 Mbps (clase media); y 200Mbps (clase rápida), con RTT=106 ms, W_{max}=128, L=512 bytes, y T_b=0 ms; sin ACK retardado (a) y con ACK retardado (b)

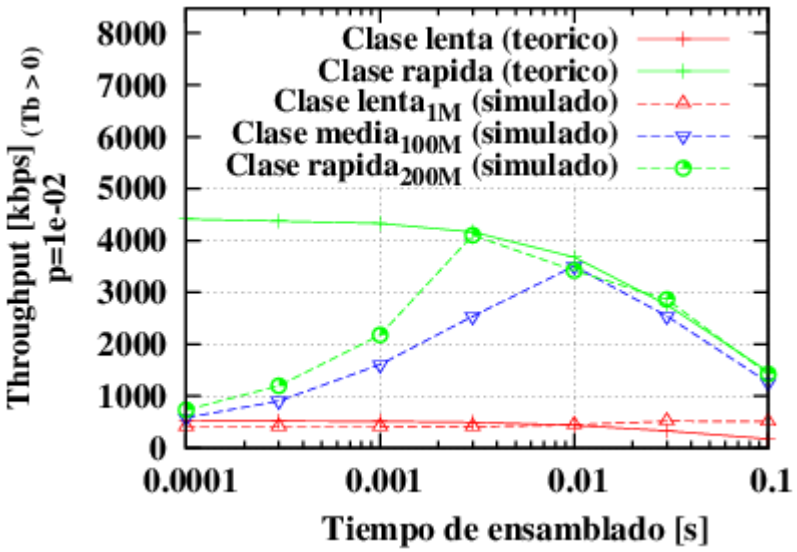


(*a*)

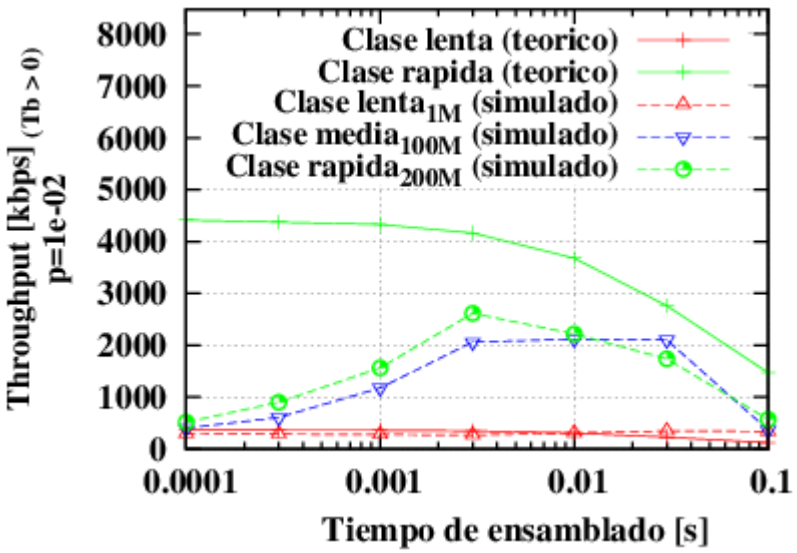


(*b*)

Figura 4.14 Ganancia DFL de TCP Reno vs probabilidad de pérdida de ráfagas para fuentes de 1 Mbps (clase lenta); 100 Mbps (clase media); y 200Mbps (clase rápida), con RTT=106 ms, W_{max}=128, L=512 bytes y T_b=3 ms; sin ACK retardado (a) y con ACK retardado (b)

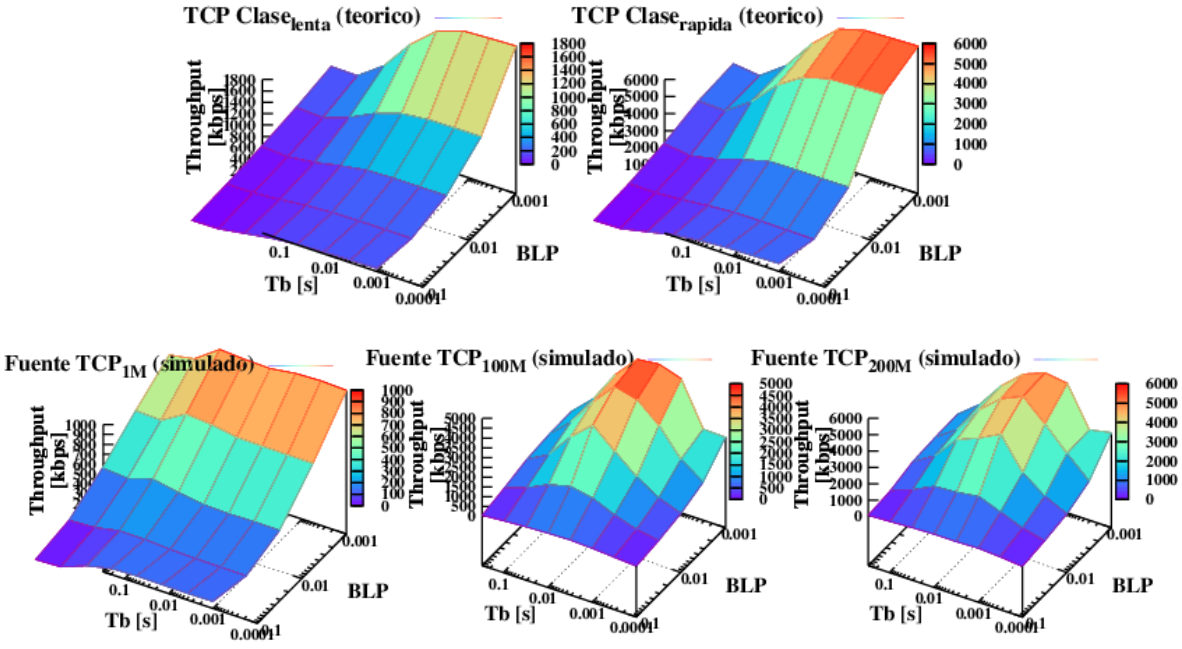


(*a*)

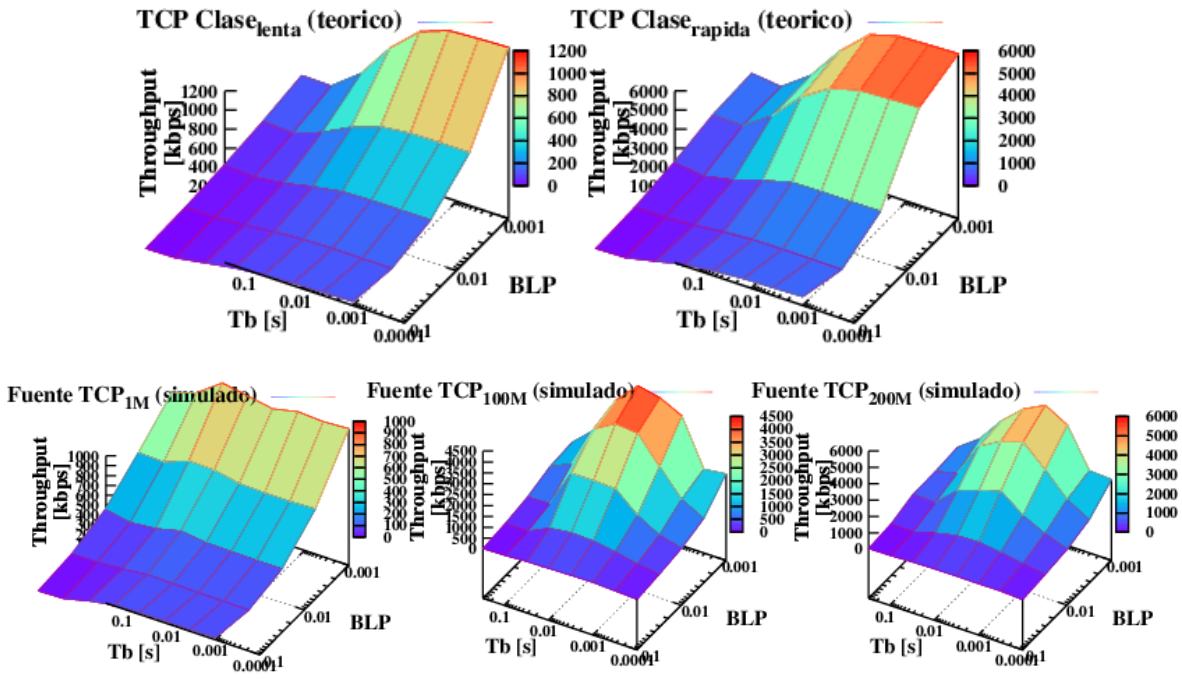


(*b*)

Figura 4.15 Throughput de TCP Reno vs tiempo de ensamblado para fuentes de 1 Mbps (clase lenta); 100 Mbps (clase media); y 200Mbps (clase rápida), con RTT=106 ms, W_{max}=128, L=512 bytes y p_b=0.01; sin ACK retardado (a) y con ACK retardado (b)

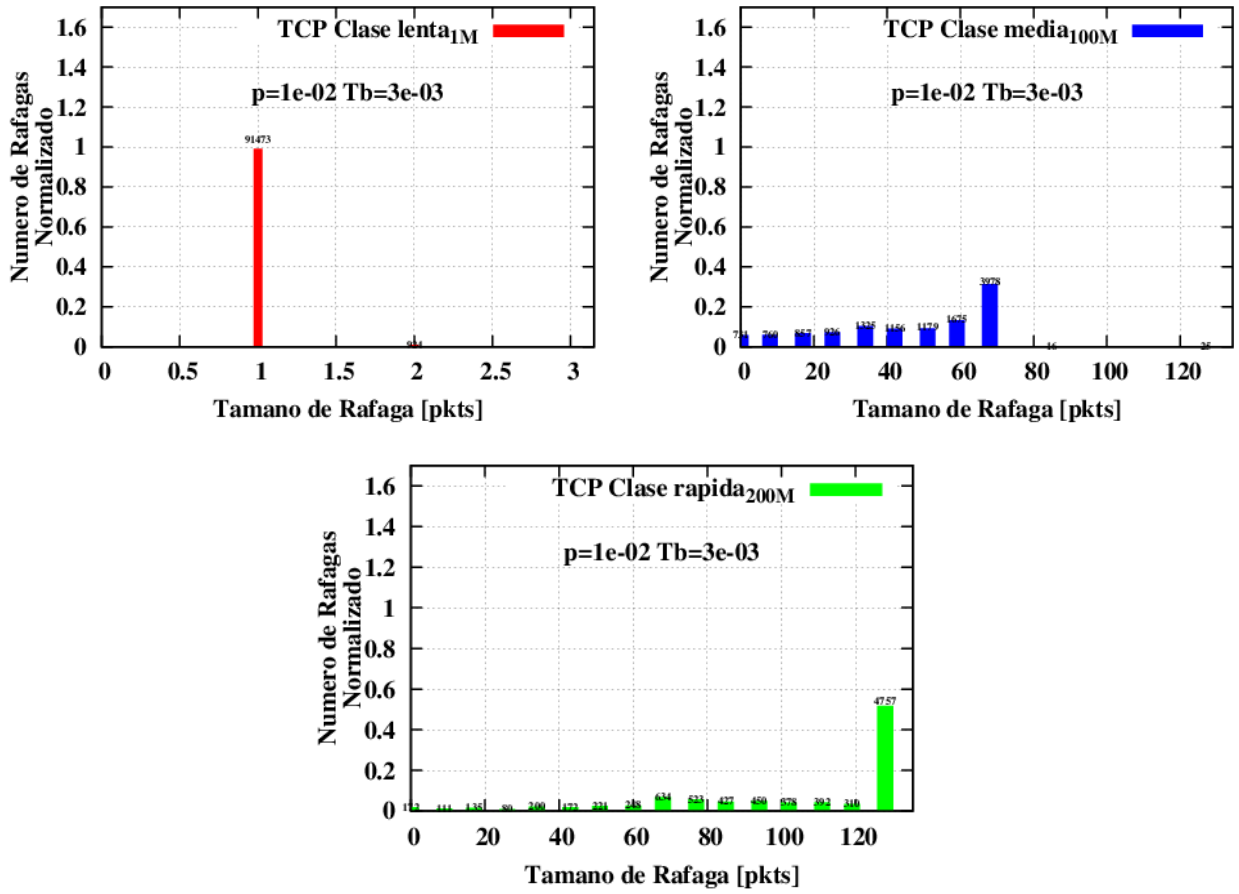


(*a*)

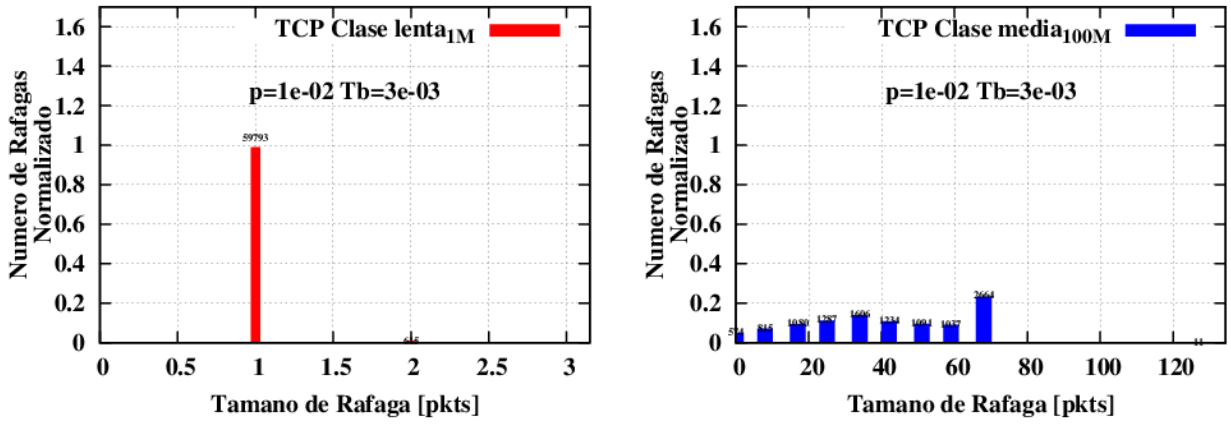


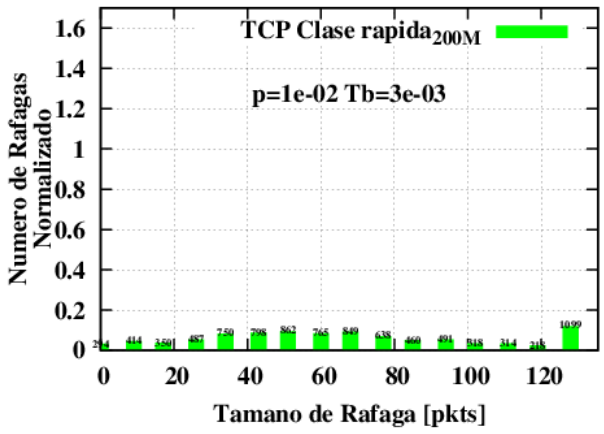
(*b*)

Figura 4.16 Throughput de TCP Reno con resultados analíticos y simulados, para diferentes valores de probabilidad de pérdida de ráfagas y tiempos de ensamblado; sin ACK retardado (a) y con ACK retardado (b)


TCP Clase lenta1M
p=1e-02 Tb=3e-03
Numero de Rafagas Normalizado
Tamano de Rafaga [pkts]
91473
TCP Clase media100M
p=1e-02 Tb=3e-03
Numero de Rafagas Normalizado
Tamano de Rafaga [pkts]
TCP Clase rapida200M
p=1e-02 Tb=3e-03
Numero de Rafagas Normalizado
Tamano de Rafaga [pkts]
4757

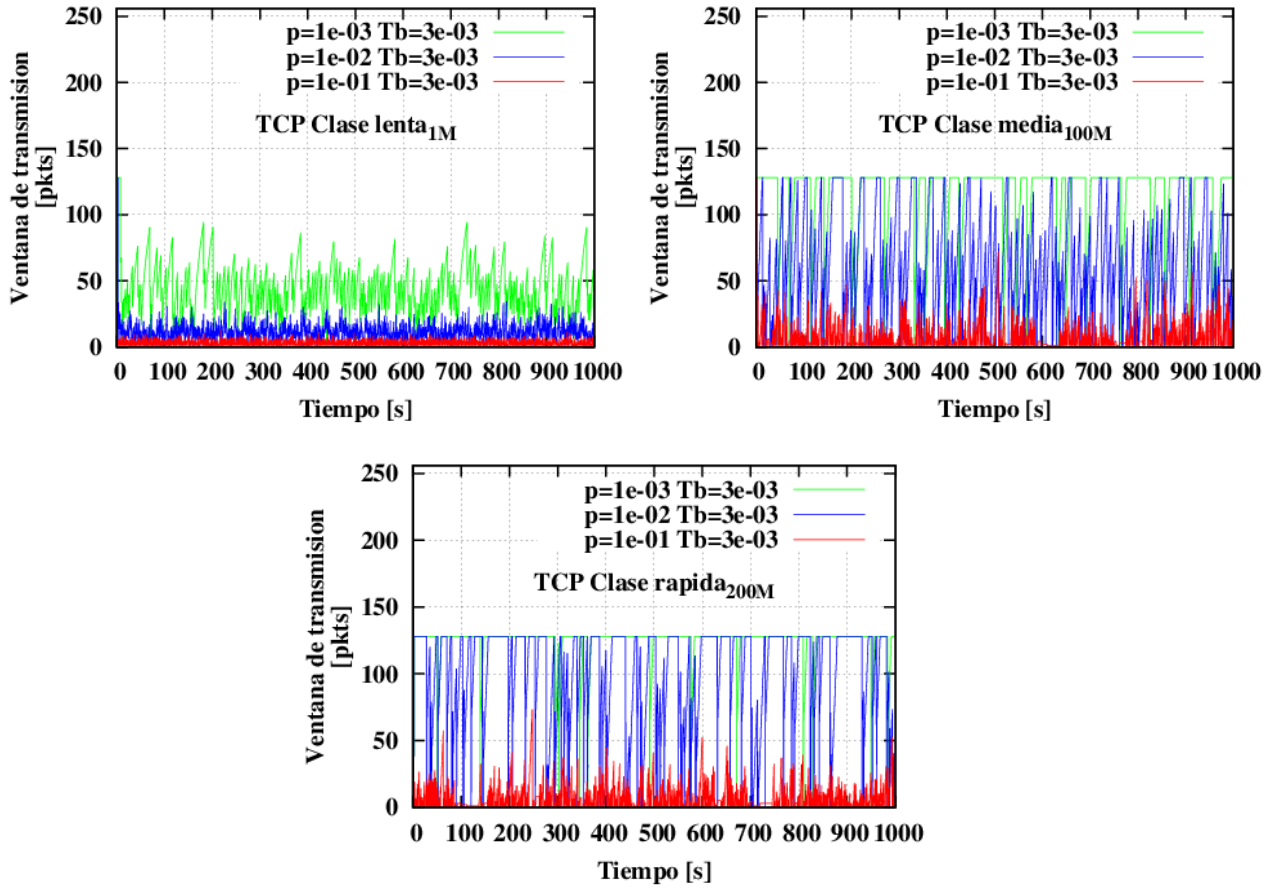
(*a*)


TCP Clase lenta1M
p=1e-02 Tb=3e-03
Numero de Rafagas Normalizado
Tamano de Rafaga [pkts]
59793
TCP Clase media100M
p=1e-02 Tb=3e-03
Numero de Rafagas Normalizado
Tamano de Rafaga [pkts]
2664

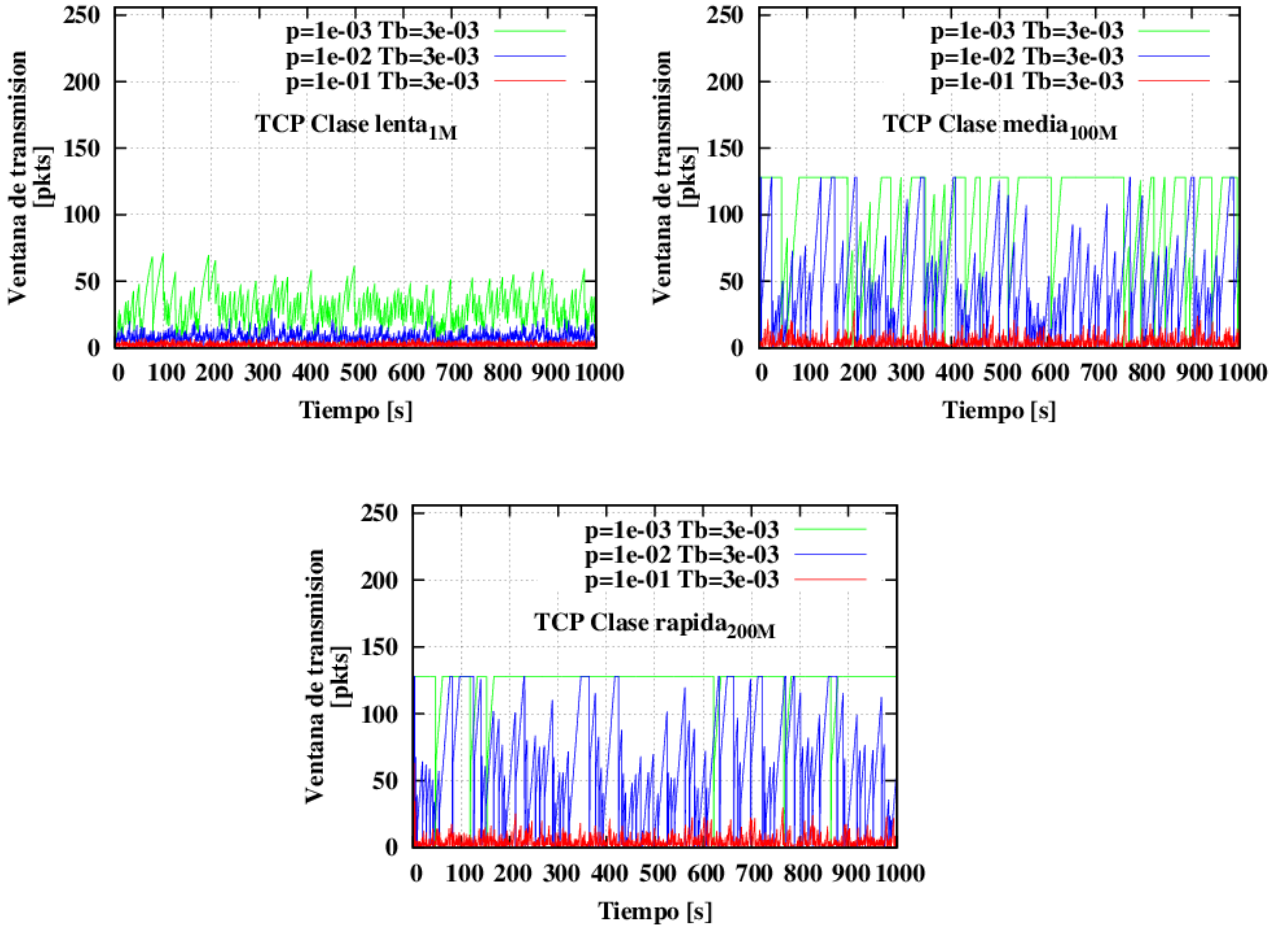


(*b*)

Figura 4.17 Distribución del tamaño de ráfaga de TCP Reno para para fuentes de 1 Mbps (clase lenta); 100 Mbps (clase media); y 200Mbps (clase rápida), con RTT= 106 ms, W_{max}=128, L=512 bytes, p_b=0.01 y T_b=3ms; sin ACK retardado (a) y con ACK retardado (b)



(*a*)



(*b*)

Figura 4.18 Evolución del tamaño de la ventana de transmisión de TCP Reno para fuentes de 1 Mbps (clase lenta); 100 Mbps (clase media); y 200Mbps (clase rápida), con RTT= 106 ms, W_{max}=128, T_b=3ms y diferentes probabilidades de pérdida; sin ACK retardado (a) y con ACK retardado (a) y con ACK retardado (b)

Los datos obtenidos considerando el ensamblador de ráfagas en el sentido de los paquetes ACK muestran que el *throughput* de TCP Reno sobre redes OBS, al igual que el caso anterior, varía entre dos funciones límite que definen sus valores mínimo y máximo; y el modelo analítico se aproxima en gran medida al comportamiento obtenido por simulación cuando no se activa la opción de ACK retardado (ver Figuras 4.12a y 4.16a); de lo contrario el valor del *throughput* se reduce en aproximadamente un 40%, como se puede apreciar en las Figuras 4.12b y 4.16b. De manera similar que en el escenario anterior, se puede observar también que al comparar el comportamiento de TCP Reno sobre una red de conmutación de paquetes (ver

Figura 4.13) y una red OBS, para una misma condición de probabilidad de pérdida, en esta última se evidencia el efecto de ganancia DFL (ver Figura 4.14) que se experimenta en mayor medida para fuentes de clase rápida. Además, en la Figura 4.15 se puede apreciar que existe un rango de tiempos de ensamblado entre 3 ms y 30 ms donde el *throughput* de TCP presenta sus mayores valores para el caso de fuentes de clase media y rápida, mientras que fuera de ese rango, su valor disminuye considerablemente ya que aunque el ancho de banda de acceso no cambie, el tiempo de ensamblado de ráfagas puede conducir a los casos de fuentes de clase baja o media.

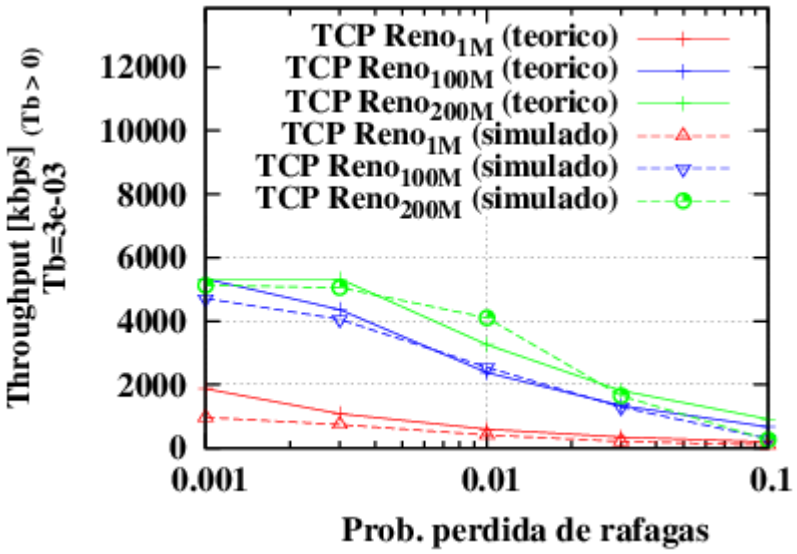
Por otro lado, del resultado obtenido en la Figura 4.17 se puede apreciar que cuando no se emplea la opción de ACK retardado, 1) la distribución del número de ráfagas de datos transmitidas para el caso de fuentes lentas se concentra en su mayor parte en un valor constante de un paquete; 2) para fuentes de clase media se distribuye en valores inferiores a 70 paquetes, con su máximo en este valor; y 3) para el caso de fuentes rápidas se distribuye en su mayor parte al valor máximo del límite de la ventana de TCP (W_{max}=128). Por el contrario, cuando se utiliza la opción de ACK retardado, como el *throughput* se reduce, la distribución del número de ráfagas de datos transmitidas generalmente hablando, presenta un comportamiento similar pero con valores inferiores; con una mayor diferencia para el caso de fuentes rápidas donde se observa una distribución en mayor grado sobre algunos puntos dentro del rango del tamaño de cada ráfaga hasta (W_{max}=128), con su valor máximo en 128 paquetes, seguido de los valores para 70 y 50 paquetes.

Finalmente en la Figura 4.18 se puede apreciar la evolución de la ventana de transmisión de TCP considerando distintos niveles de pérdida y con el mismo tiempo de ensamblado T_b=3 ms, donde se puede evidenciar claramente un mayor *throughput* a menor probabilidad de pérdidas y una mayor transmisión de segmentos cuando no se emplea la opción de ACK retardado.

4.4.2.2 Análisis del modelo basado en la teoría de renovación de Markov

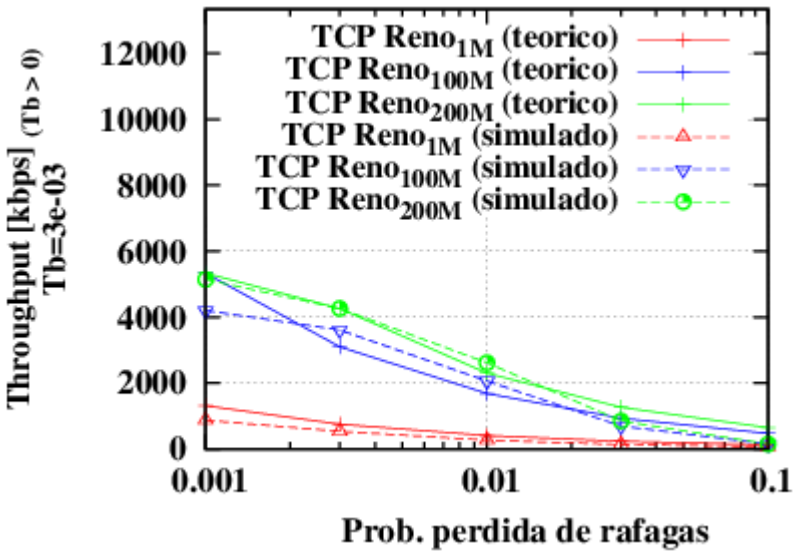
El tiempo de simulación para el caso de la evaluación de las variantes de TCP Reno, Newreno y SACK utilizando el modelo de la teoría de renovación de Markov basada en una fuente de velocidad genérica, fue de aproximadamente 23 horas (sin ACK retardado) y 17 horas (con ACK retardado). Cabe mencionar que en el análisis para la evaluación del modelo analítico que se va a contrastar mediante simulación por ordenador, se hará énfasis principalmente en cuanto a fuentes de clase media y rápida, así como también en el rango de tiempos de ensamblado de entre 1 ms y 30 ms[59], y una probabilidad de pérdida de ráfagas de 0.01 que de manera general supone un valor más realista, al resto de valores que en un ambiente en producción basado en una red DWDM, podría ser muy poco probable.

1) Resultados para TCP Reno



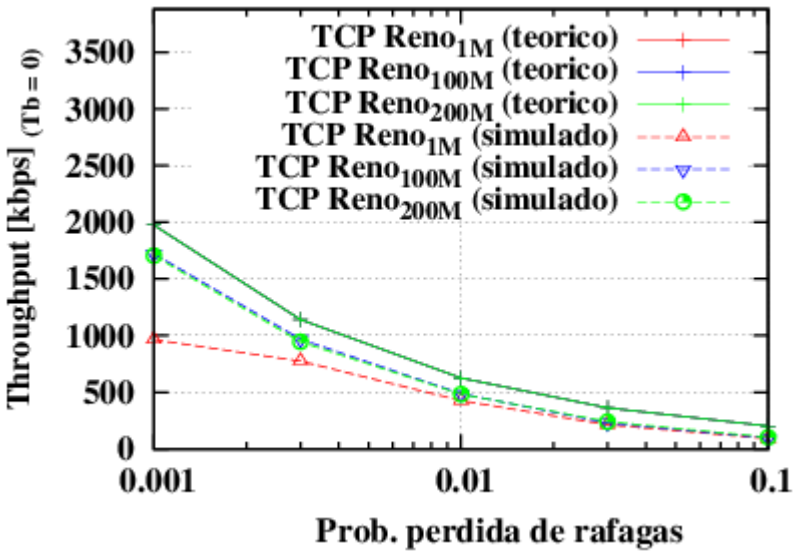
(*a*)

[59] Para valores fuera de este rango, el *throughput* de TCP disminuye considerablemente ya que aunque se mantenga el mismo ancho de banda de acceso, la variación del tiempo de ensamblado puede conducir a los casos de fuentes de clase baja o media.

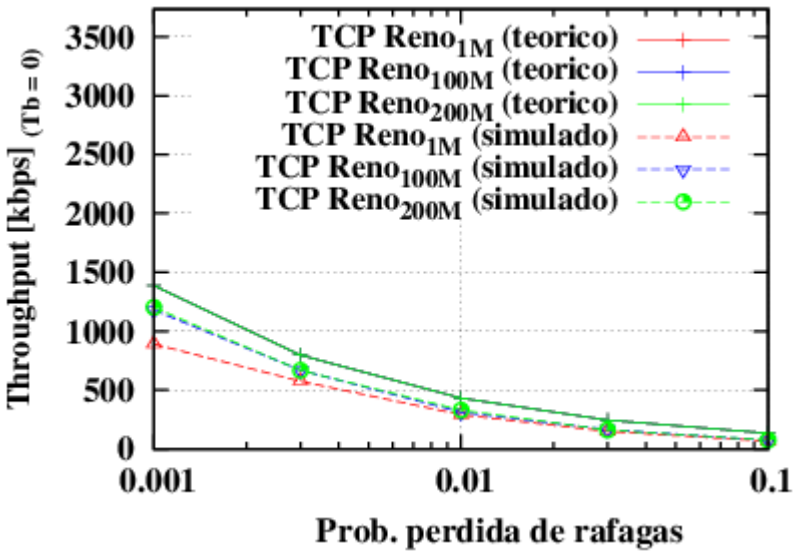


(*b*)

Figura 4.19 Throughput de TCP Reno vs probabilidad de pérdida de ráfagas para fuentes de 1 Mbps (clase lenta); 100 Mbps (clase media); y 200Mbps (clase rápida), con RTT=106 ms, W_{max}=128, L=512 bytes, y T_b=3 ms; sin ACK retardado (a) y con ACK retardado (b)

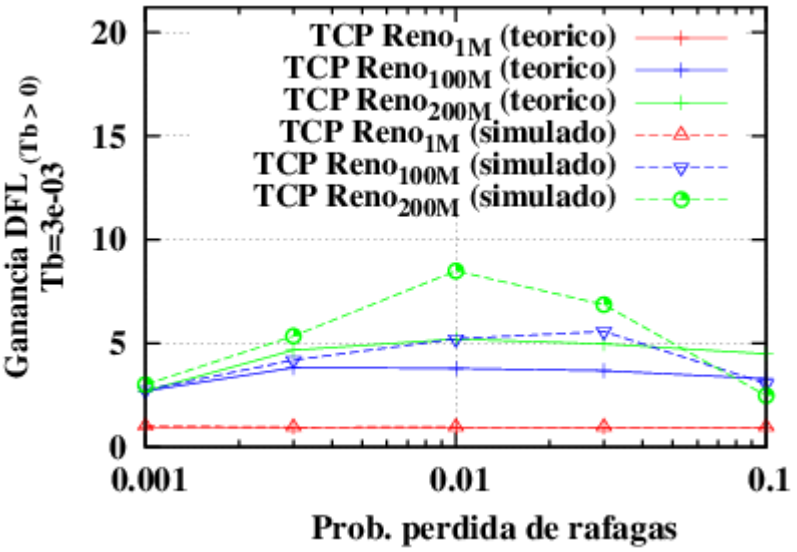


(*a*)

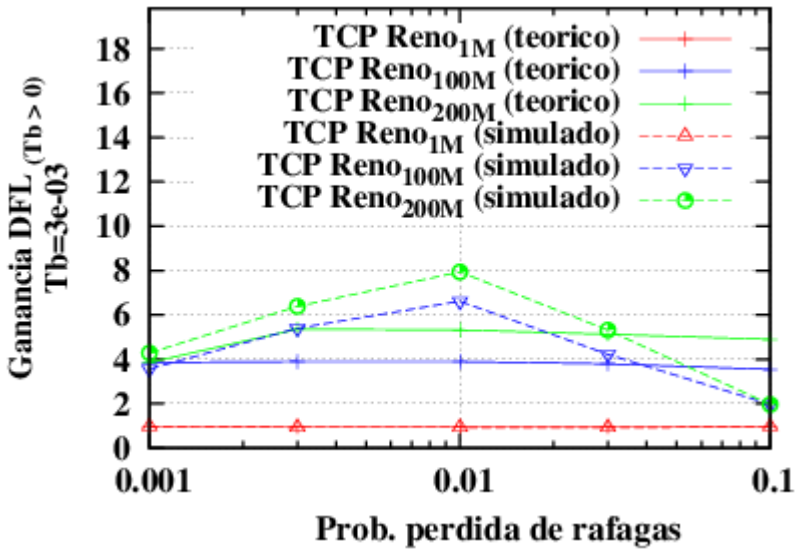


(*b*)

Figura 4.20 Throughput de TCP Reno vs probabilidad de pérdida de ráfagas para fuentes de 1 Mbps (clase lenta); 100 Mbps (clase media); y 200Mbps (clase rápida), con RTT=106 ms, W_{max}=128, L=512 bytes, y T_b=0 ms; sin ACK retardado (a) y con ACK retardado (b)

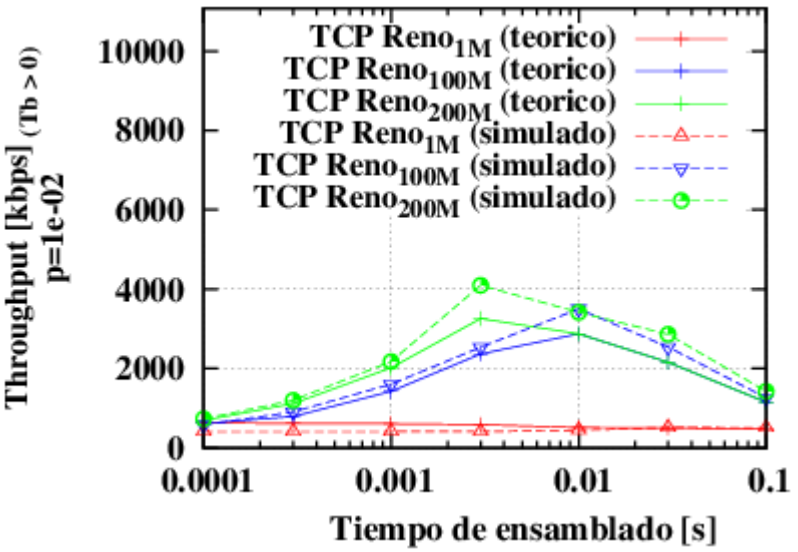


(*a*)

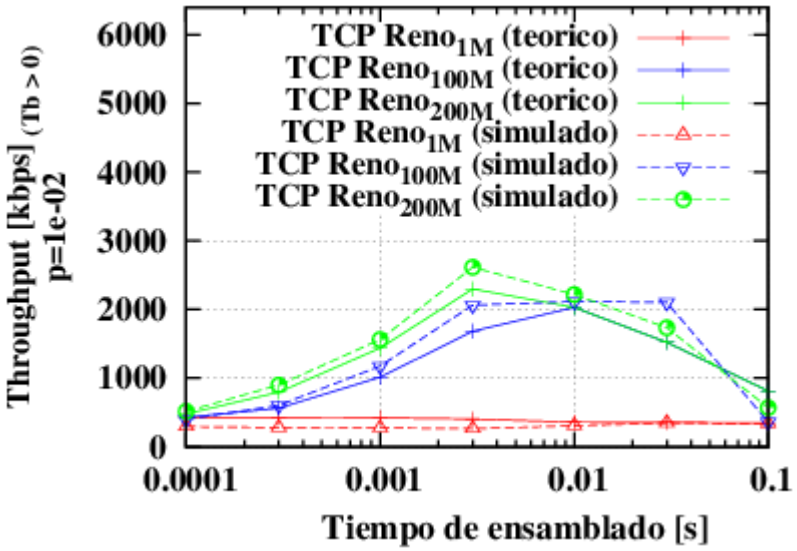


(*b*)

Figura 4.21 Ganancia DFL de TCP Reno vs probabilidad de pérdida de ráfagas para fuentes de 1 Mbps (clase lenta); 100 Mbps (clase media); y 200Mbps (clase rápida), con RTT=106 ms, W_{max}=128, L=512 bytes y T_b=3 ms; sin ACK retardado (a) y con ACK retardado (b)

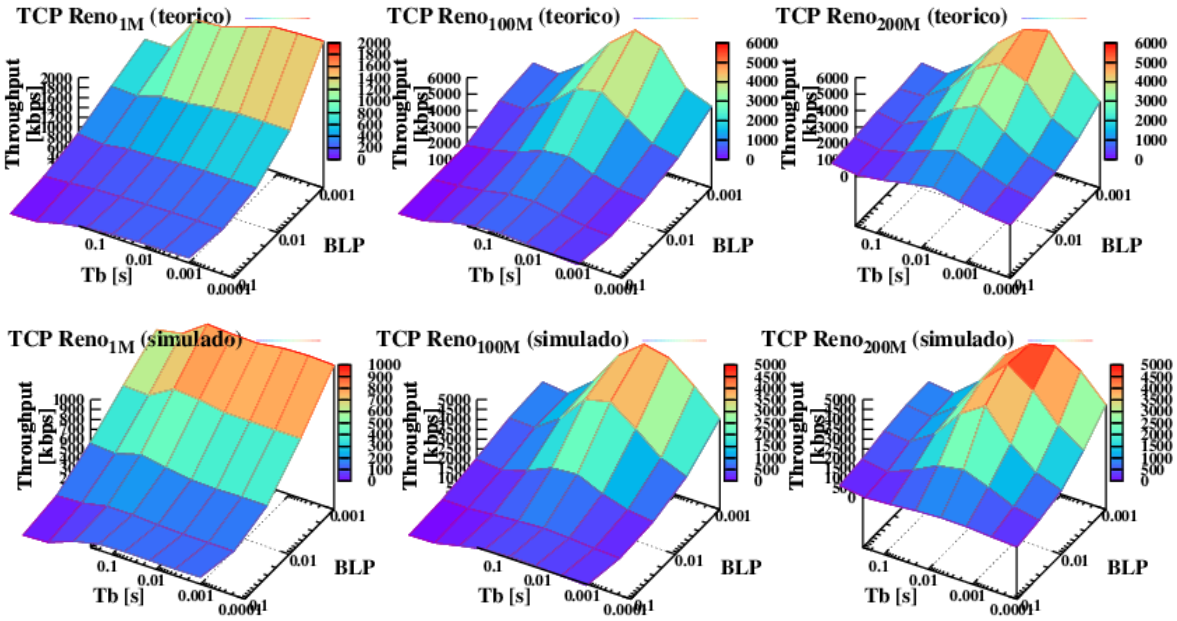


(*a*)

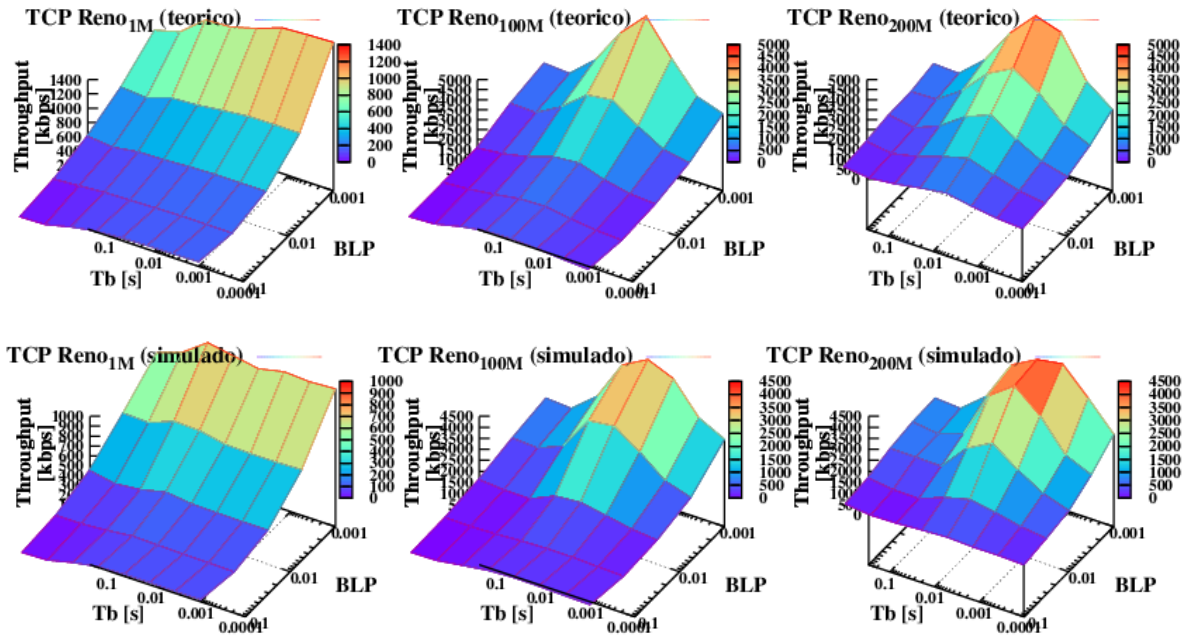


(*b*)

Figura 4.22 Throughput de TCP Reno vs tiempo de ensamblado para fuentes de 1 Mbps (clase lenta); 100 Mbps (clase media); y 200Mbps (clase rápida), con RTT=106 ms, W_{max}=128, L=512 bytes y p_b=0.01; sin ACK retardado (a) y con ACK retardado (b)

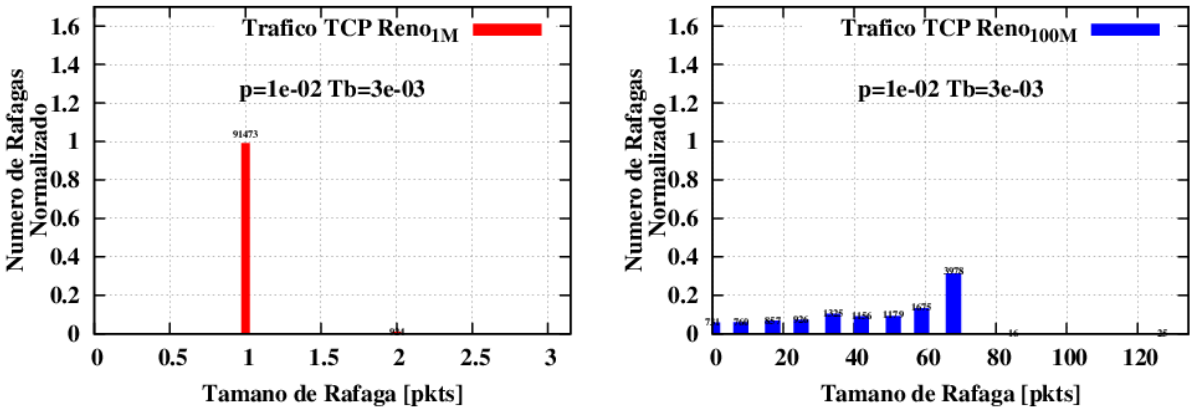


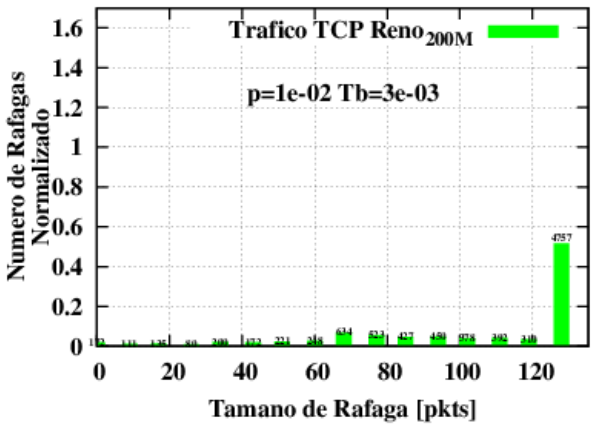
(*a*)



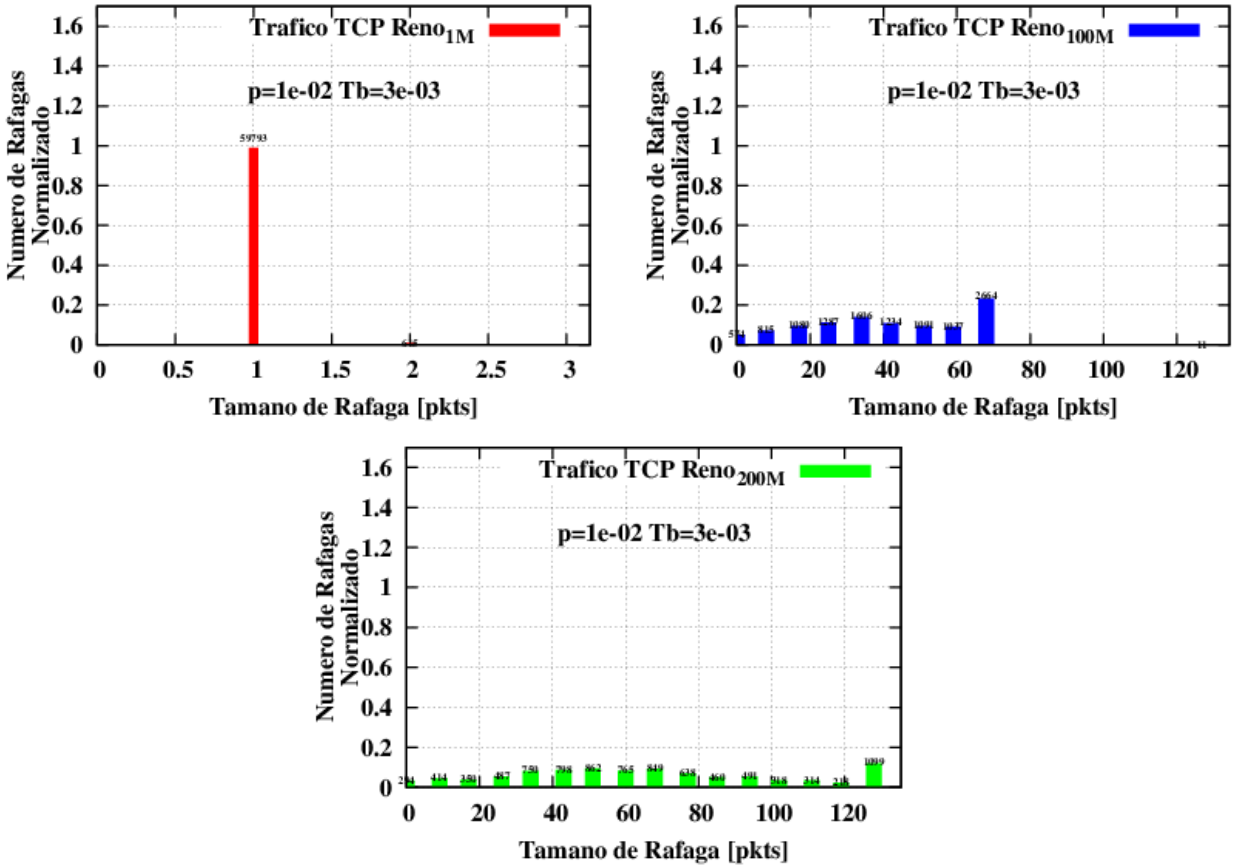
(*b*)

Figura 4.23 Throughput de TCP Reno con resultados analíticos y simulados, para diferentes valores de probabilidad de pérdida de ráfagas y tiempos de ensamblado; sin ACK retardado (a) y con ACK retardado (b)



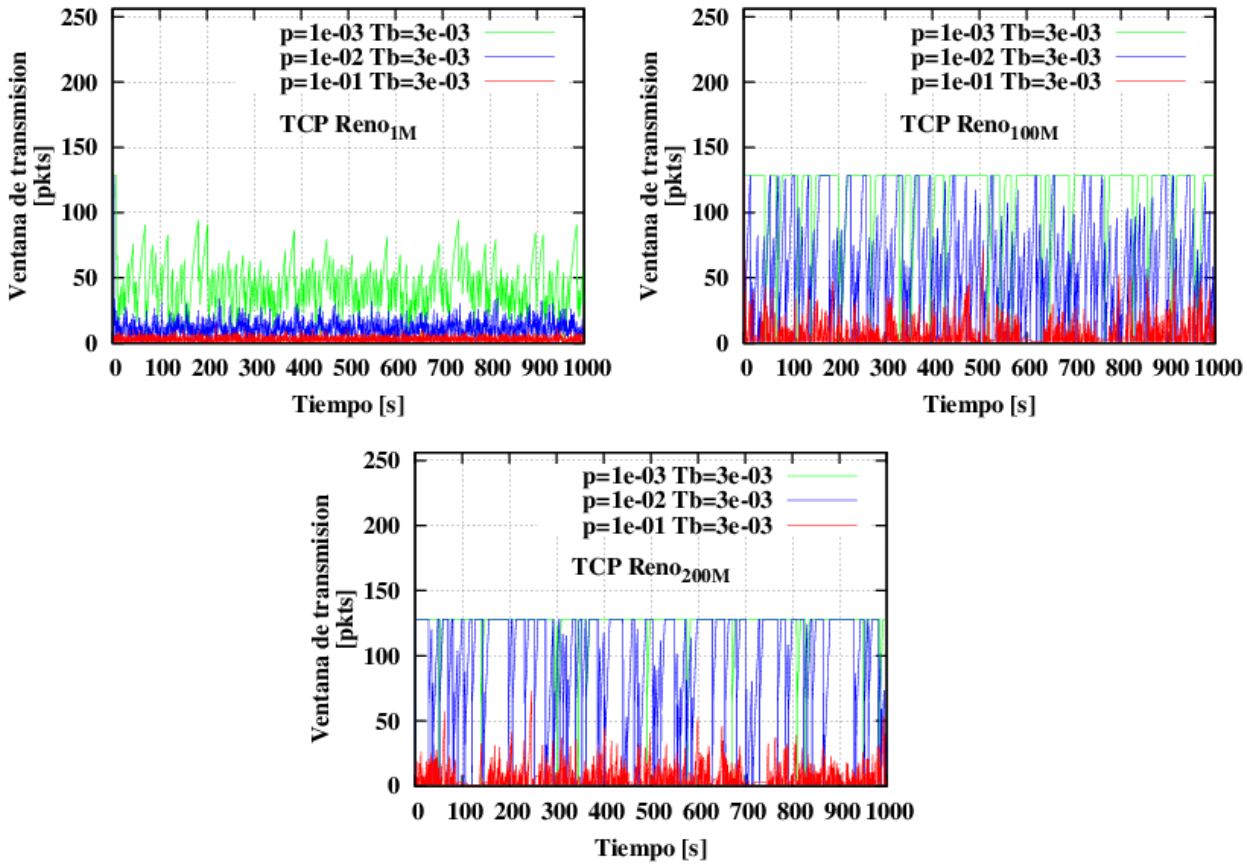


(*a*)

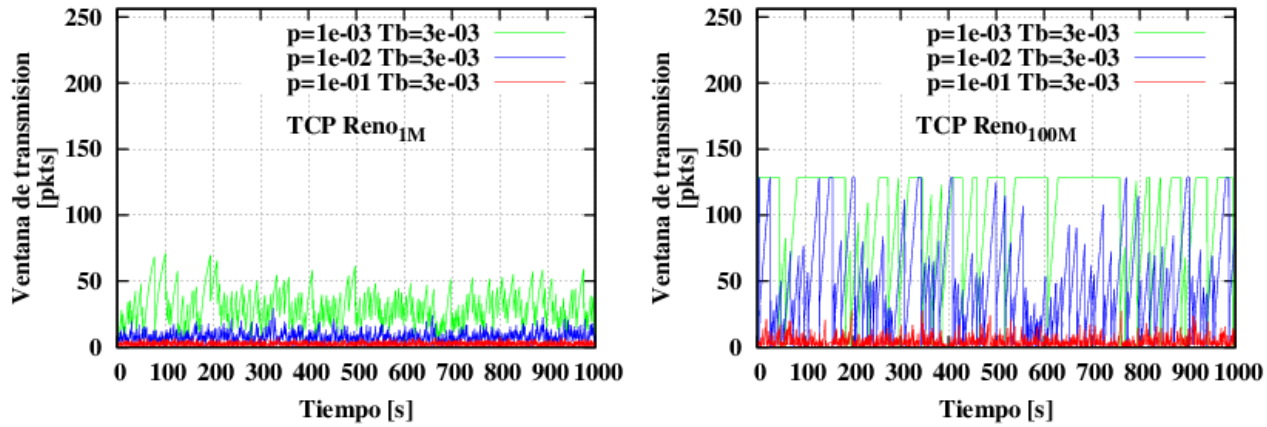


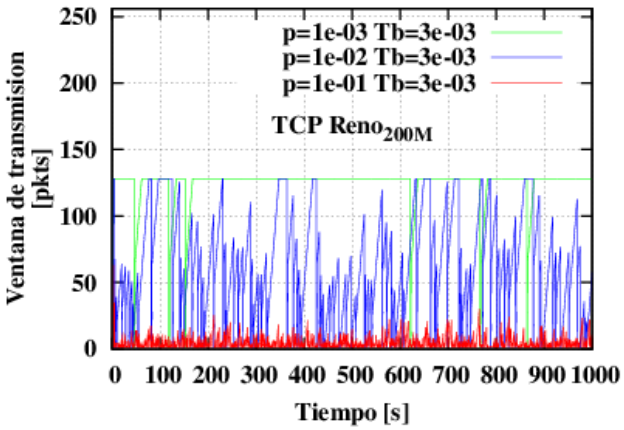
(*b*)

Figura 4.24 Distribución del tamaño de ráfaga de TCP Reno para para fuentes de 1 Mbps (clase lenta); 100 Mbps (clase media); y 200Mbps (clase rápida), con RTT= 106 ms, W_{max}=128, L=512 bytes, p_b=0.01 y T_b=3ms; sin ACK retardado (a) y con ACK retardado (b)


p=1e-03 Tb=3e-03
p=1e-02 Tb=3e-03
p=1e-01 Tb=3e-03
TCP Reno1M
TCP Reno100M
TCP Reno200M
Ventana de transmision [pkts]
Tiempo [s]
0
50
100
150
200
250
0 100 200 300 400 500 600 700 800 900 1000

(*a*)


p=1e-03 Tb=3e-03
p=1e-02 Tb=3e-03
p=1e-01 Tb=3e-03
TCP Reno1M
TCP Reno100M
Ventana de transmision [pkts]
Tiempo [s]
0
50
100
150
200
250
0 100 200 300 400 500 600 700 800 900 1000



(*b*)

Figura 4.25 Evolución del tamaño de la ventana de transmisión de TCP Reno para fuentes de 1 Mbps (clase lenta); 100 Mbps (clase media); y 200Mbps (clase rápida), con RTT= 106 ms, W_{max}=128, T_b=3ms y diferentes probabilidades de pérdida; sin ACK retardado (a) y con ACK retardo (b)

Los resultados obtenidos para el caso de TCP Reno, teniendo en cuenta los parámetros definidos anteriormente para el análisis, muestran que el modelo analítico del *throughput* para esta variante de TCP sobre redes OBS, presenta un buen nivel de aproximación (ver Figuras 4.19 y 4.23) considerando el tiempo empleado para la simulación[60], con un error experimental inferior al 28% y 33% para los casos con y sin ACK retardado, respectivamente (ver Figuras 4.26 y 4.27), obtenido de los archivos de traza de la simulación; con una reducción en este último en el *throughput* de aproximadamente el 36% y 19% para el caso de fuentes medias y rápidas, respectivamente, al comparar ambos casos como se puede apreciar en la Tabla 4.3.

[60] Mientras mayor sea el tiempo de la simulación, se obtendrán mejores resultados, a expensas de incrementar significativamente el tiempo de ejecución, que en la mayoría de los casos sería prácticamente inaceptable si se utilizan computadores convencionales.

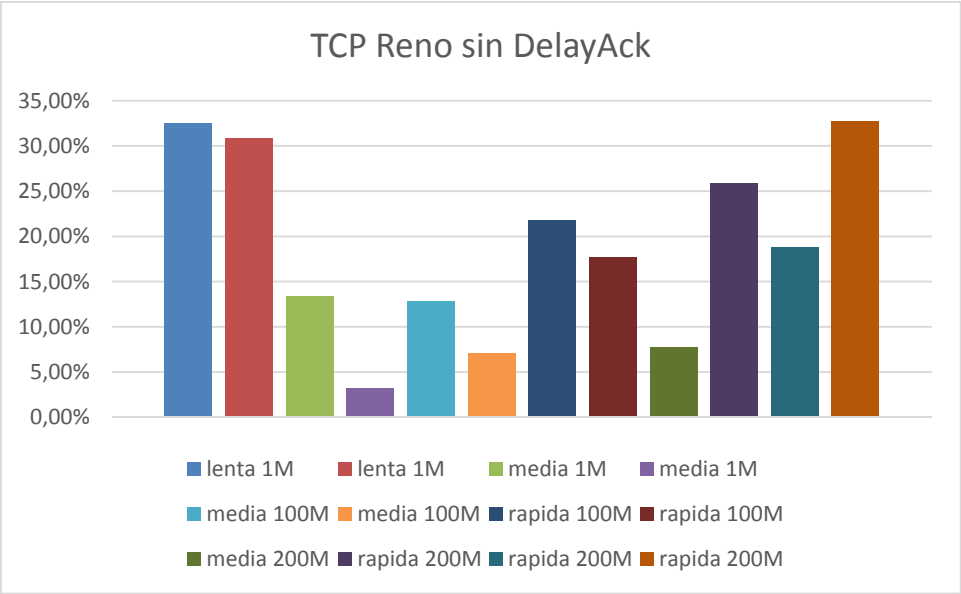


Figura 4.26 Error experimental para TCP Reno sin ACK retardado, para T_b=1 ms - 30ms y p_b=0.01

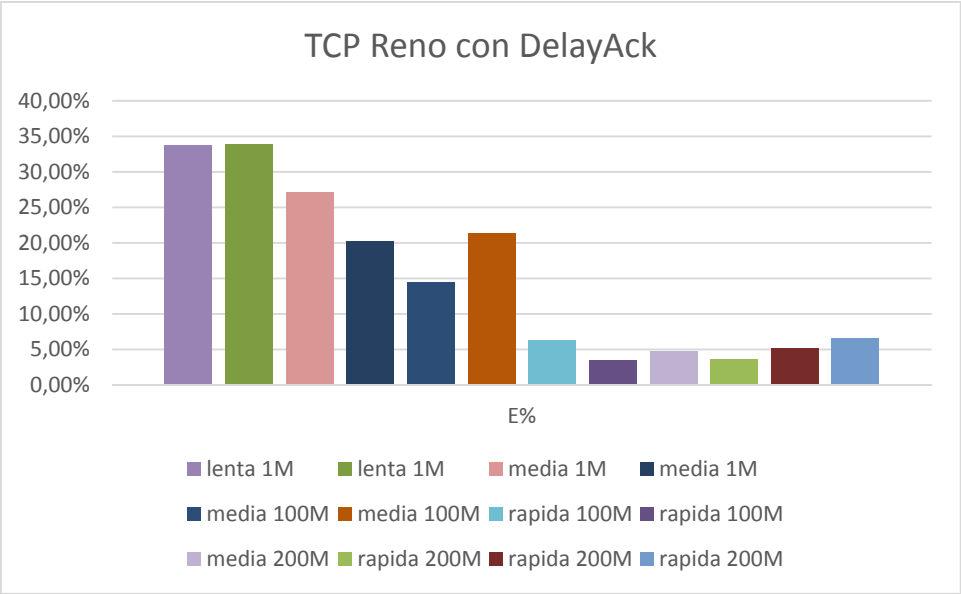


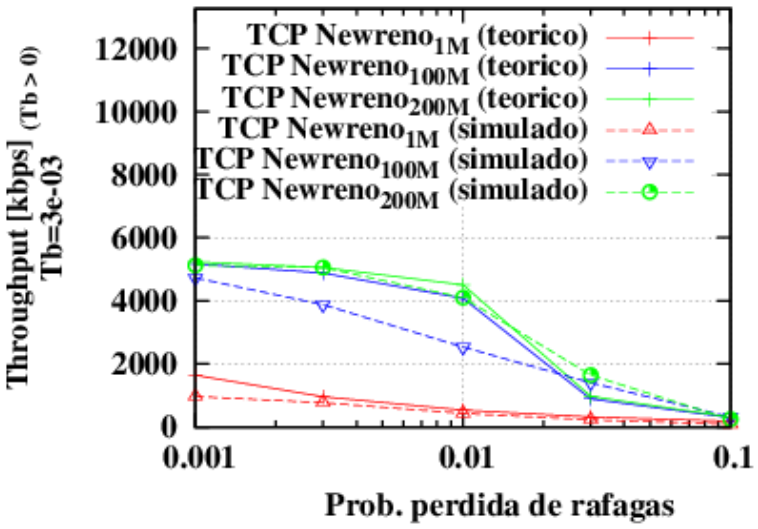
Figura 4.27 Error experimental para TCP Reno con ACK retardado, para T_b=1 ms - 30ms y p_b=0.01

Tabla 4.3 Comparativa del throughput de TCP Reno con y sin ACK retardado para Tb=0.003 y p_b=0.01

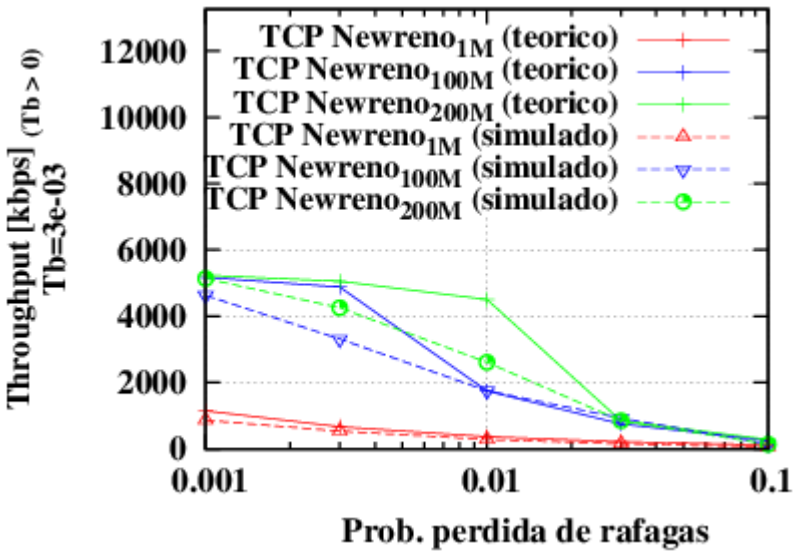
Variante de TCP	T_b	p_b	*Throughput* teórico [kbps]	*Throughput* simulado [kbps]	Clase	Ancho de banda de acceso [Mbps]	Error experimental%
Reno	0,003	0,01	2.370,24	2.539,496	media	100	7,14%
Reno	0,003	0,01	3.257,136	4.100,404	rápida	200	25,89%
Reno DelayAck	0,003	0,01	1.682,774	2.061,927	media	100	22,53%
Reno DelayAck	0,003	0,01	2.299,095	2.613,715	rápida	200	13,68%

De manera similar que en el caso anterior, se puede evidenciar el efecto de ganancia DFL (ver Figura 4.21), que se experimenta en mayor medida para fuentes de clase rápida, al comparar el *throughput* de TCP sobre OBS con una red de conmutación de paquetes (ver Figura 4.20). Por otro lado, se puede apreciar también que el *throughput* de TCP presenta sus mayores valores dentro del rango de tiempos de ensamblado entre 3 ms y 30 ms (ver Figura 4.22) para el caso de fuentes de clase media y rápida. Finalmente, el resultado de la distribución del número de ráfagas de datos y la evolución de la ventana de transmisión de TCP (ver Figuras 4.24 y 4.25) es similar al caso anterior, al tratarse de la misma variante de este protocolo.

2) Resultados para TCP Newreno

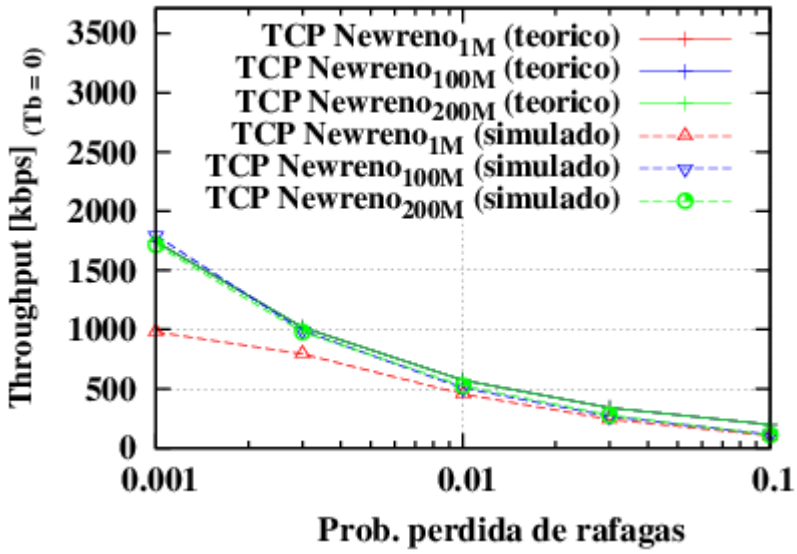


(*a*)

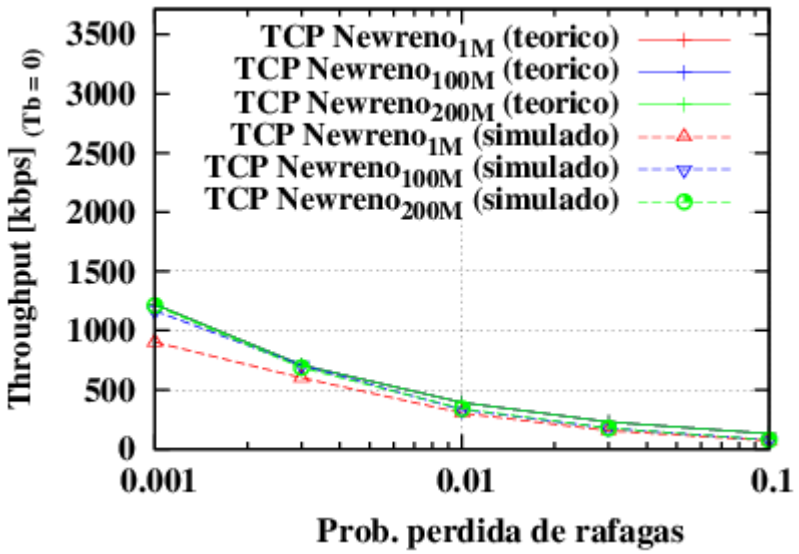


(*b*)

Figura 4.28 Throughput de TCP Newreno vs probabilidad de pérdida de ráfagas para fuentes de 1 Mbps (clase lenta); 100 Mbps (clase media); y 200Mbps (clase rápida), con RTT=106 ms, W_{max}=128, L=512 bytes, y T_b=3 ms; sin ACK retardado (a) y con ACK retardado (b)

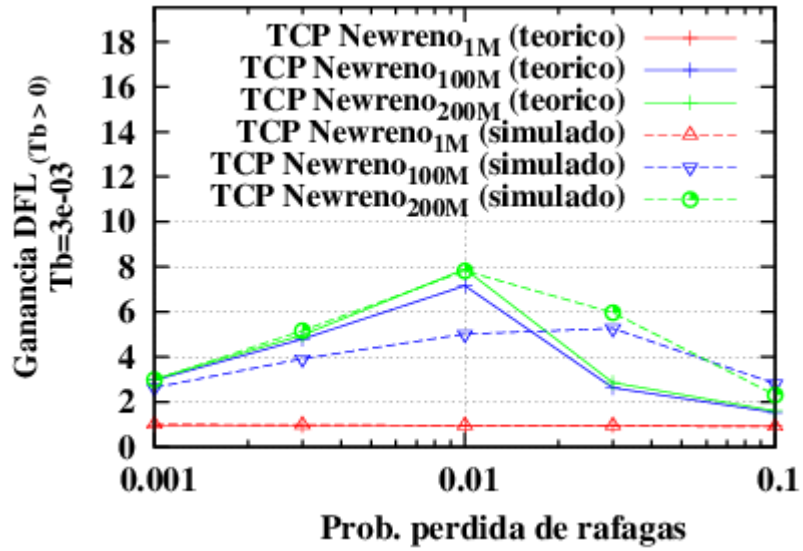


(*a*)

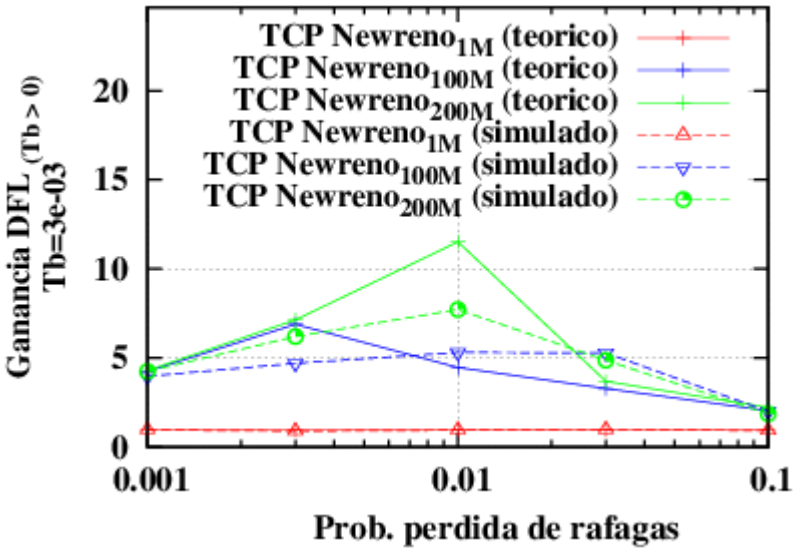


(*b*)

Figura 4.29 Throughput de TCP Newreno vs probabilidad de pérdida de ráfagas para fuentes de 1 Mbps (clase lenta); 100 Mbps (clase media); y 200Mbps (clase rápida), con RTT=106 ms, W_{max}=128, L=512 bytes, y T_b=0 ms; sin ACK retardado (a) y con ACK retardado (b)

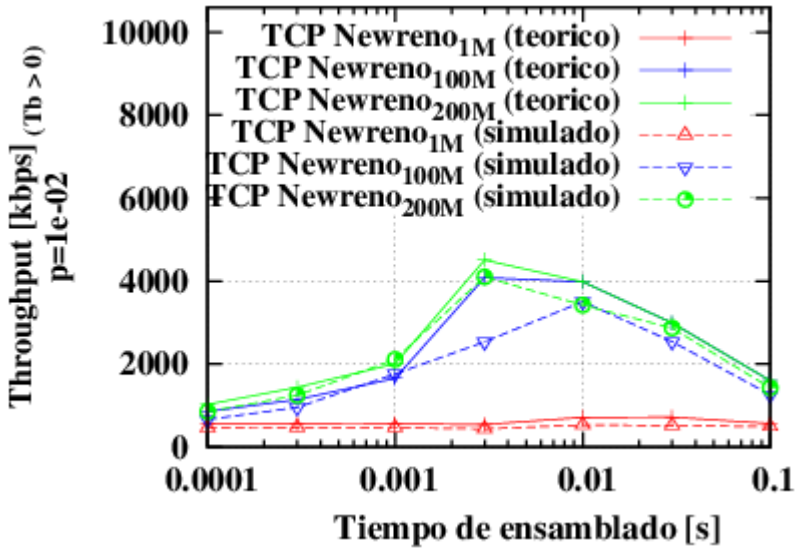


(*a*)

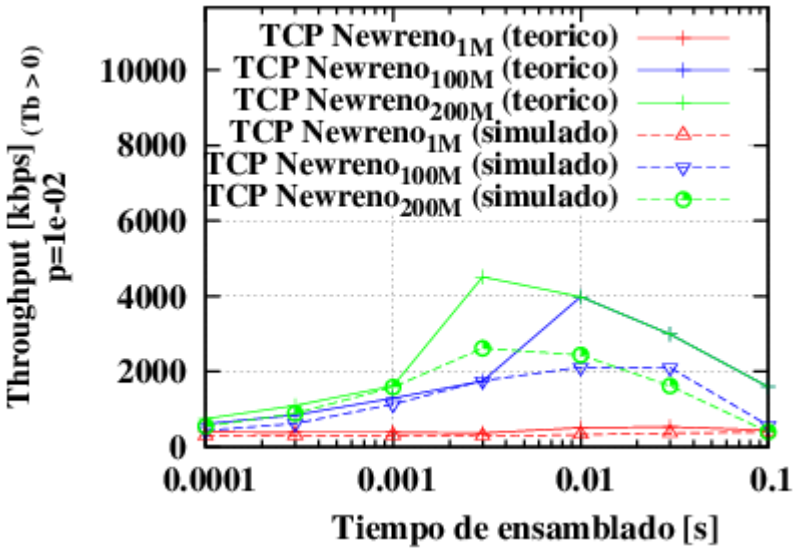


(*b*)

Figura 4.30 Ganancia DFL de TCP Newreno vs probabilidad de pérdida de ráfagas para fuentes de 1 Mbps (clase lenta); 100 Mbps (clase media); y 200Mbps (clase rápida), con RTT=106 ms, W_{max}=128, L=512 bytes y T_b=3 ms; sin ACK retardado (a) y con ACK retardado (b)

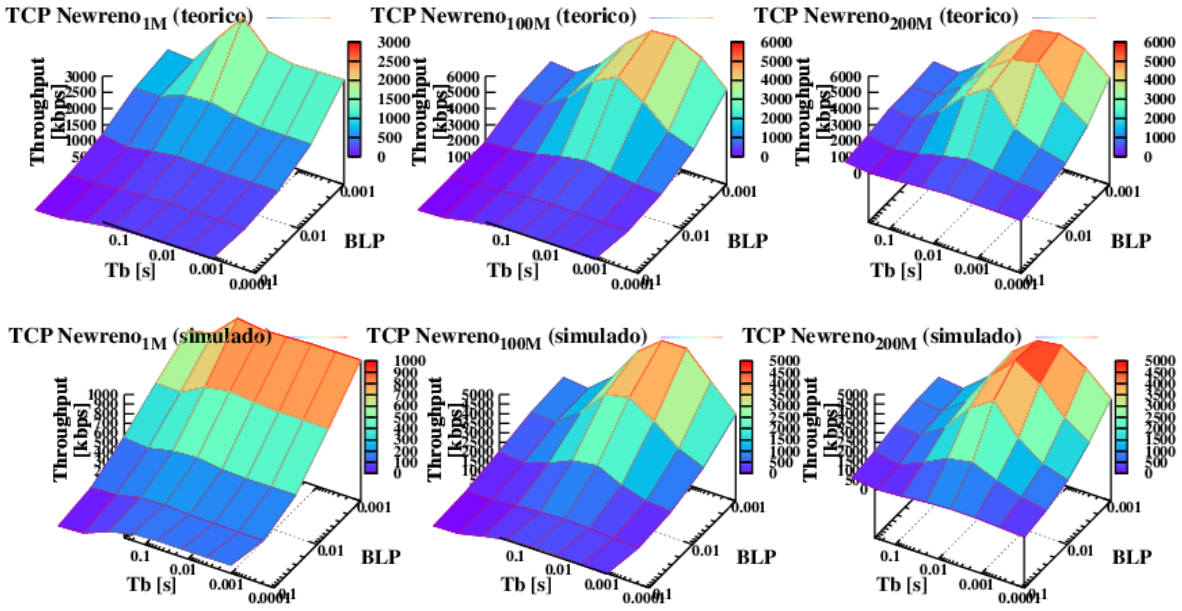


(*a*)

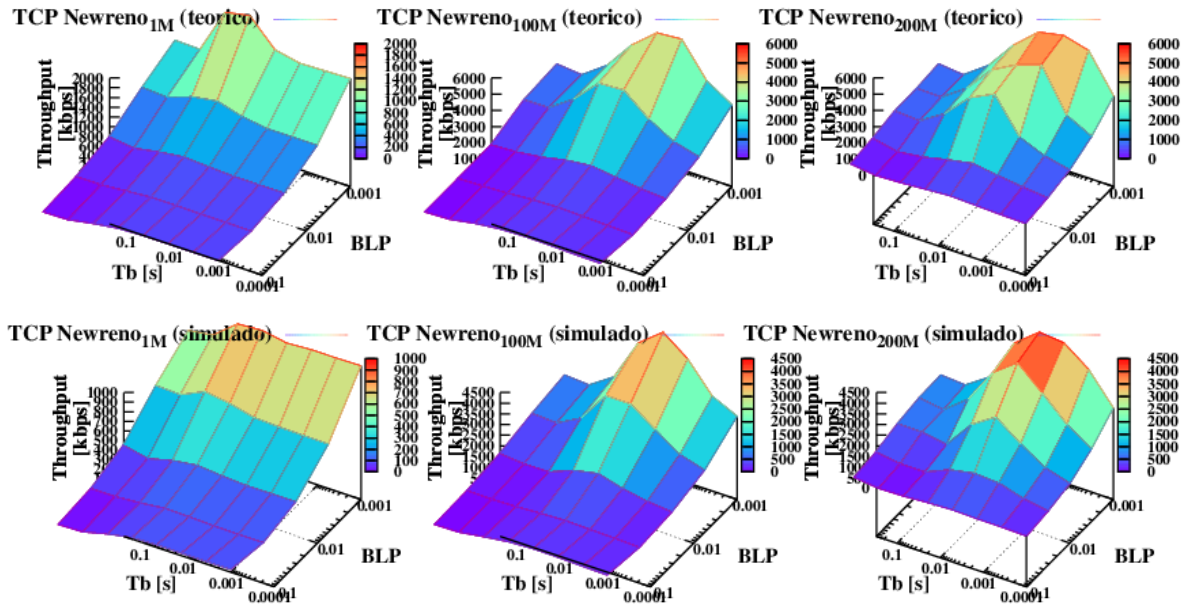


(*b*)

Figura 4.31 Throughput de TCP Newreno vs tiempo de ensamblado para fuentes de 1 Mbps (clase lenta); 100 Mbps (clase media); y 200Mbps (clase rápida), con RTT=106 ms, W_{max}=128, L=512 bytes y p_b=0.01; sin ACK retardado (a) y con ACK retardado (b)

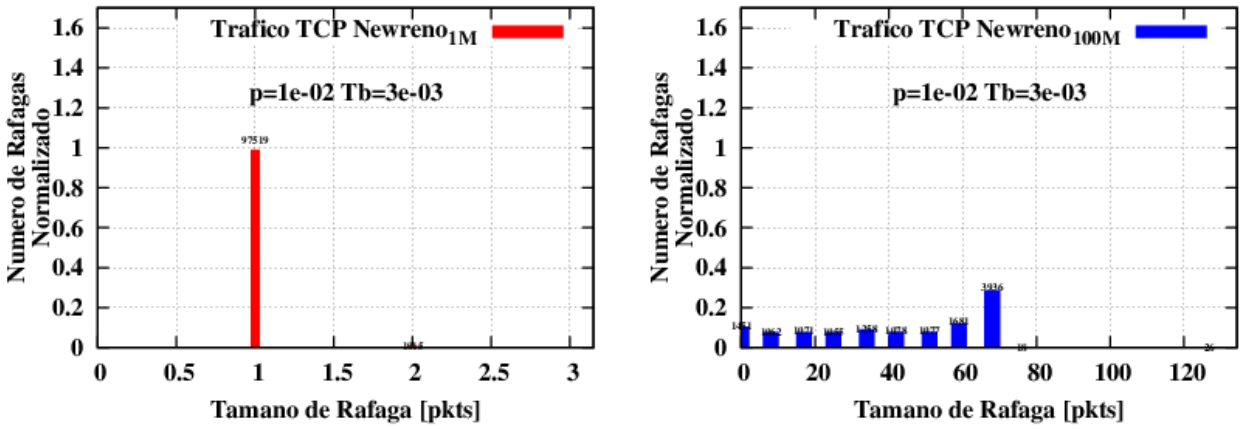


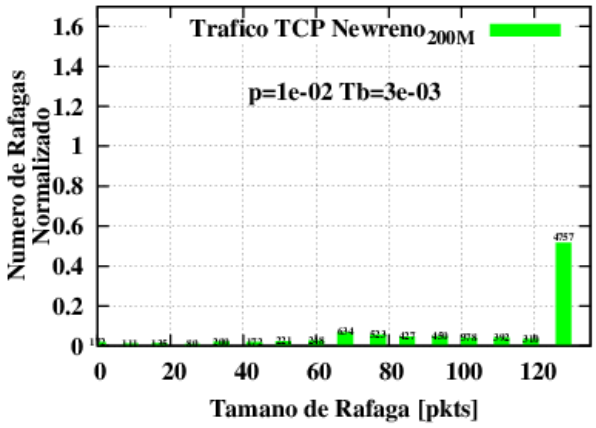
(*a*)



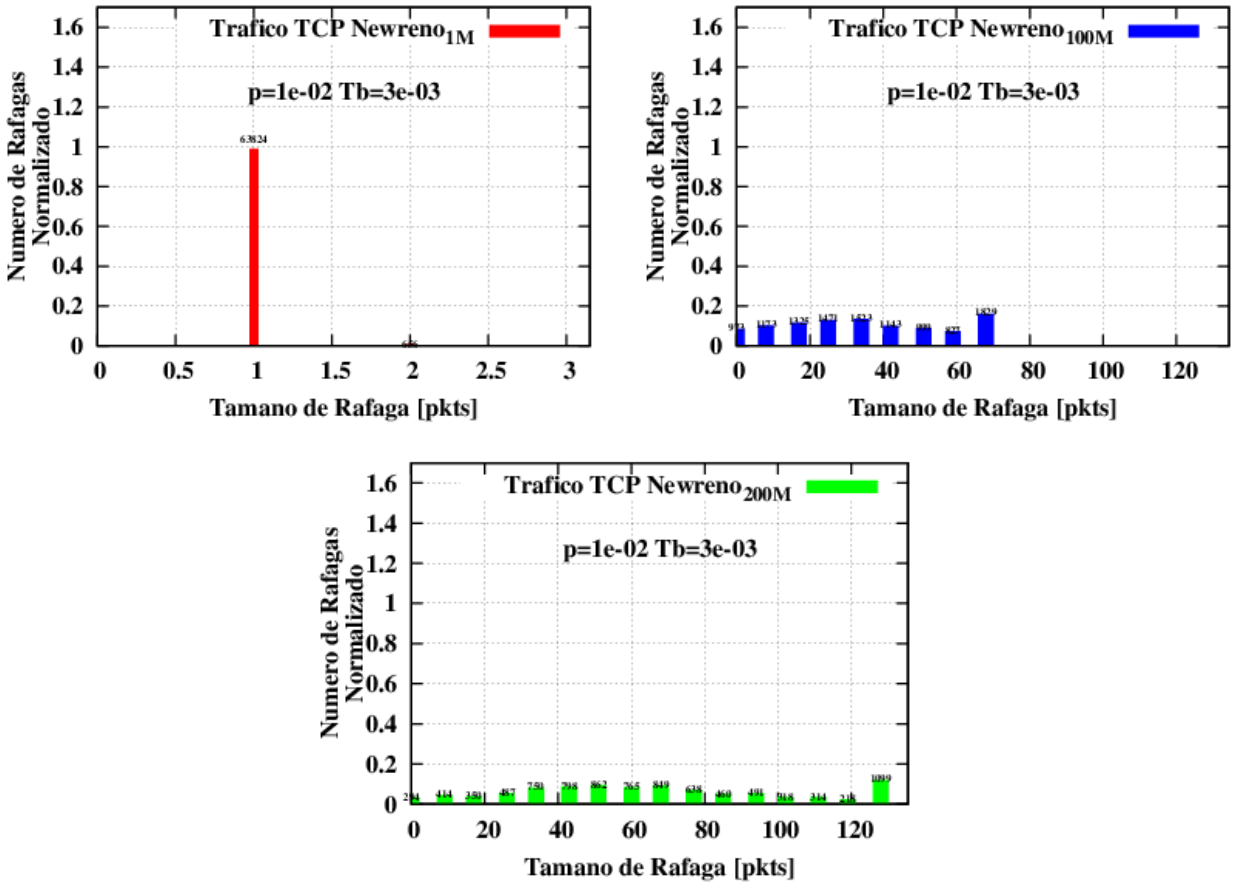
(*b*)

Figura 4.32 Throughput de TCP Newreno con resultados analíticos y simulados, para diferentes valores de probabilidad de pérdida de ráfagas y tiempos de ensamblado; sin ACK retardado (a) y con ACK retardado (b)



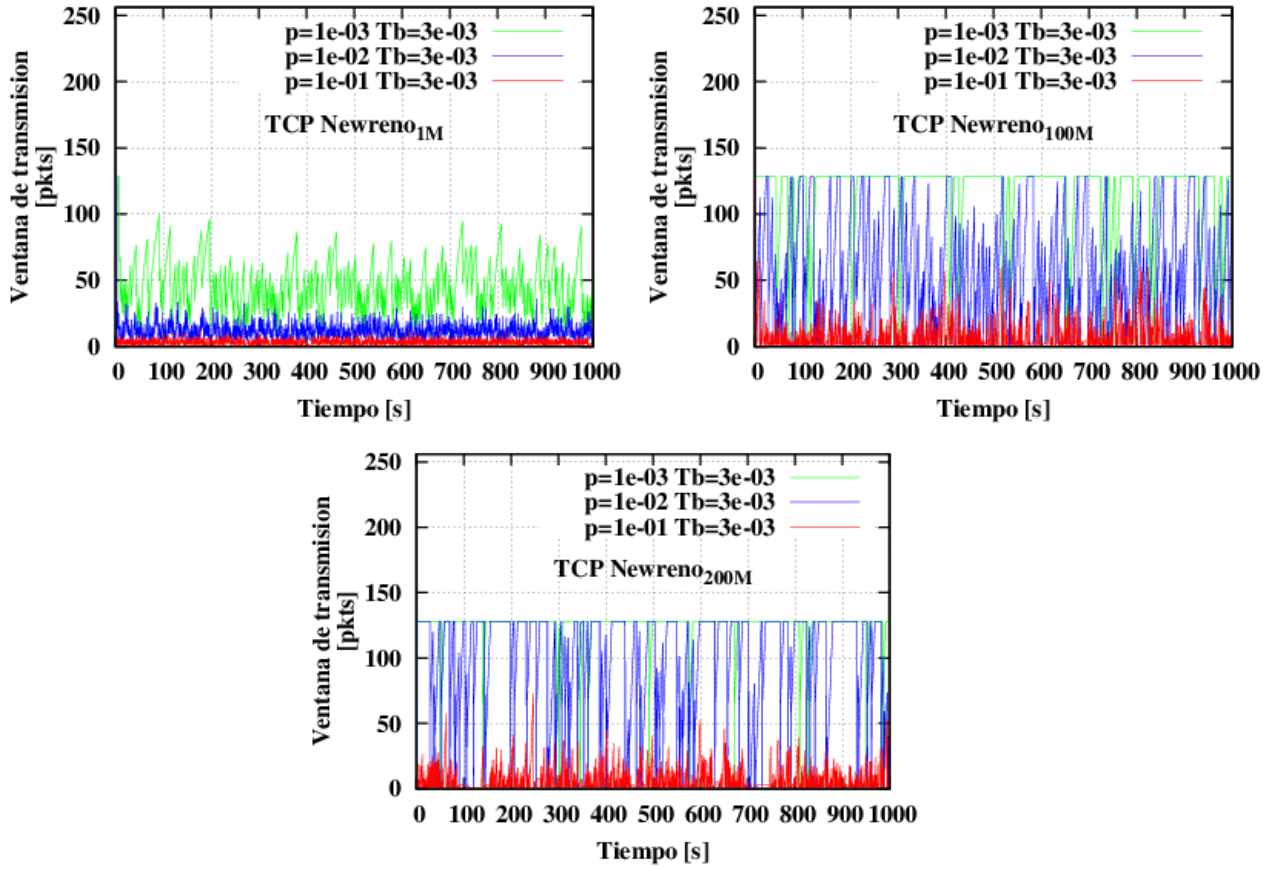


(*a*)

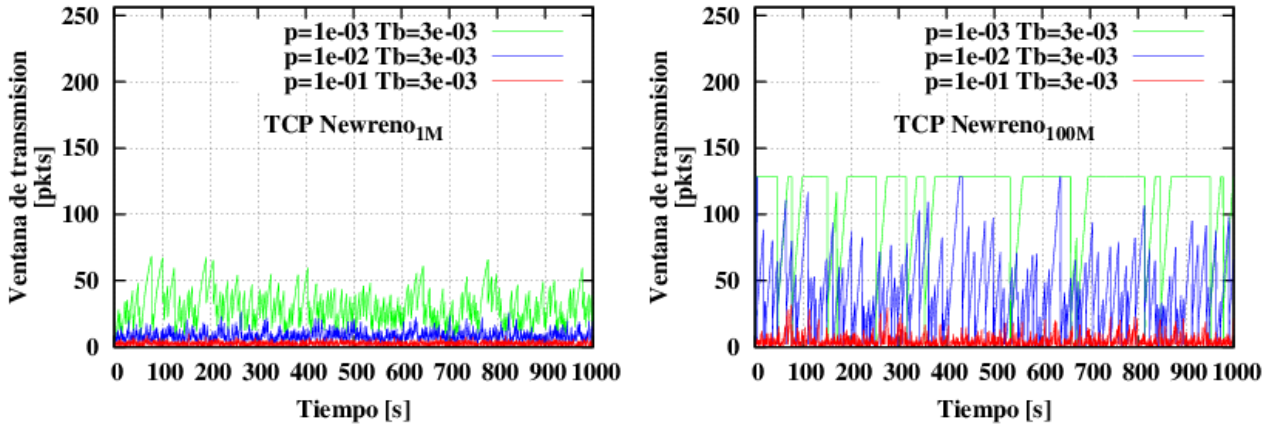


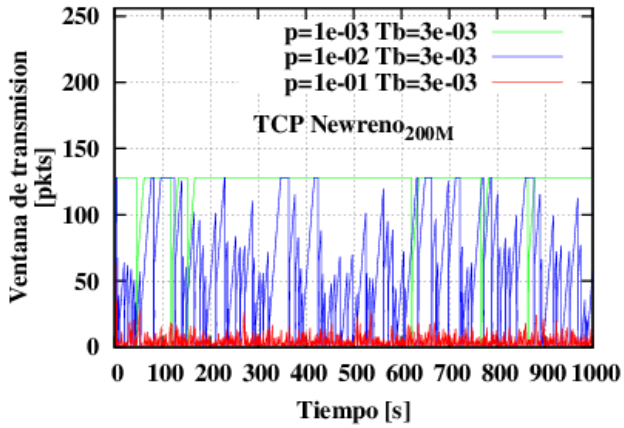
(*b*)

Figura 4.33 Distribución del tamaño de ráfaga de TCP Newreno para para fuentes de 1 Mbps (clase lenta); 100 Mbps (clase media); y 200Mbps (clase rápida), con RTT= 106 ms, W_{max}=128, L=512 bytes, p_b=0.01 y T_b=3ms; sin ACK retardado (a) y con ACK retardado (b)


p=1e-03 Tb=3e-03
p=1e-02 Tb=3e-03
p=1e-01 Tb=3e-03
TCP Newreno1M
TCP Newreno100M
TCP Newreno200M
Ventana de transmision [pkts]
Tiempo [s]
0
50
100
150
200
250
0 100 200 300 400 500 600 700 800 900 1000

(*a*)


p=1e-03 Tb=3e-03
p=1e-02 Tb=3e-03
p=1e-01 Tb=3e-03
TCP Newreno1M
TCP Newreno100M
Ventana de transmision [pkts]
Tiempo [s]
0
50
100
150
200
250
0 100 200 300 400 500 600 700 800 900 1000



(*b*)

Figura 4.34 Evolución del tamaño de la ventana de transmisión de TCP Newreno para fuentes de 1 Mbps (clase lenta); 100 Mbps (clase media); y 200Mbps (clase rápida), con RTT= 106 ms, W_{max}=128, T_b=3ms y diferentes probabilidades de pérdida; sin ACK retardado (a) y con ACK retardado (b)

Los resultados obtenidos para el caso de TCP Newreno, teniendo en cuenta los parámetros definidos para el análisis, muestran que el modelo analítico del *throughput* para esta variante de TCP sobre redes OBS, presenta un buen nivel de aproximación (ver Figuras 4.28 y 4.32) para el caso sin ACK retardado, considerando el tiempo empleado para la simulación, con un error experimental inferior al 15% en la mayoría de los casos (ver Figura 4.35), obtenido de los archivos de traza de la simulación; a diferencia de la alternativa con ACK retardado que experimenta un error experimental inferior al 40% en la mayoría de los casos (ver Figura 4.36); con una reducción en el *throughput* de aproximadamente el 31% y 36% para el caso de fuentes medias y rápidas, respectivamente, al comparar ambos casos como se puede apreciar en la Tabla 4.4. Con respecto al incremento significativo del error experimental para el caso con ACK retardado, se debe indicar que el modelo analítico sobreestima el *throughput* de TCP para el caso de fuentes de clase media y rápida bajo condiciones de pérdidas en el rango entre 0.01 y 0.02, lo cual se puede evidenciar claramente en la Figura 4.28b.

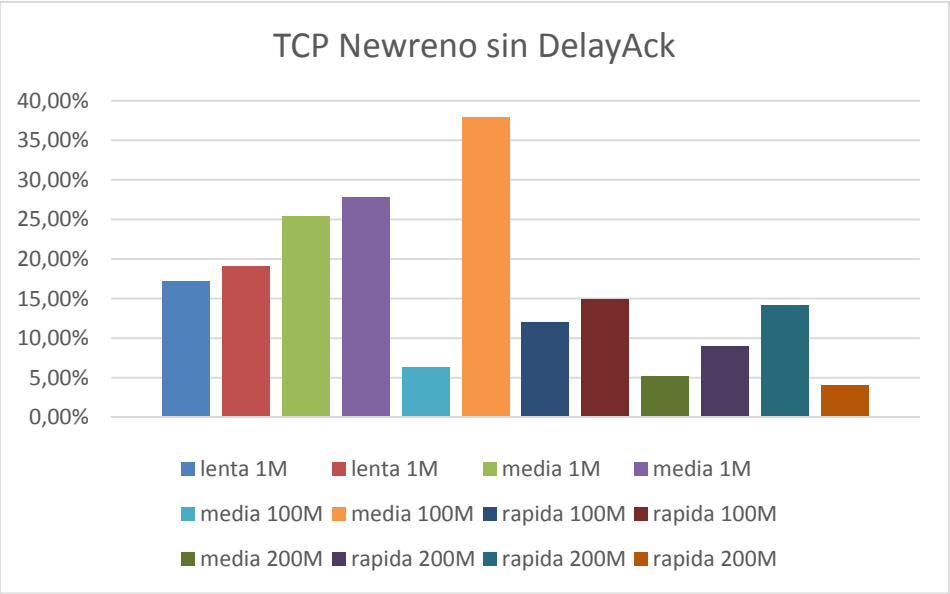


Figura 4.35 Error experimental para TCP Newreno sin ACK retardado, para T_b=1 ms - 30ms y p_b=0.01

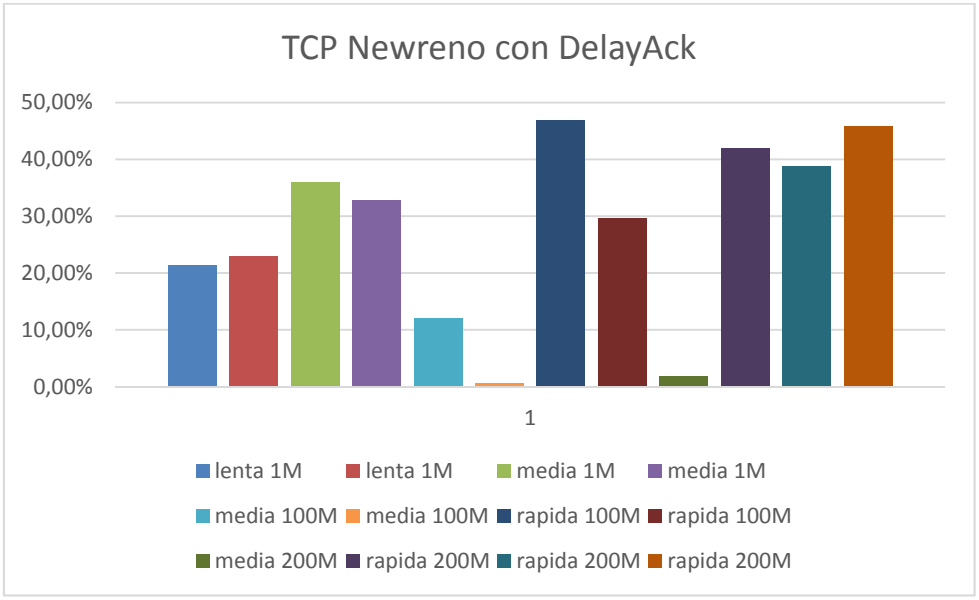


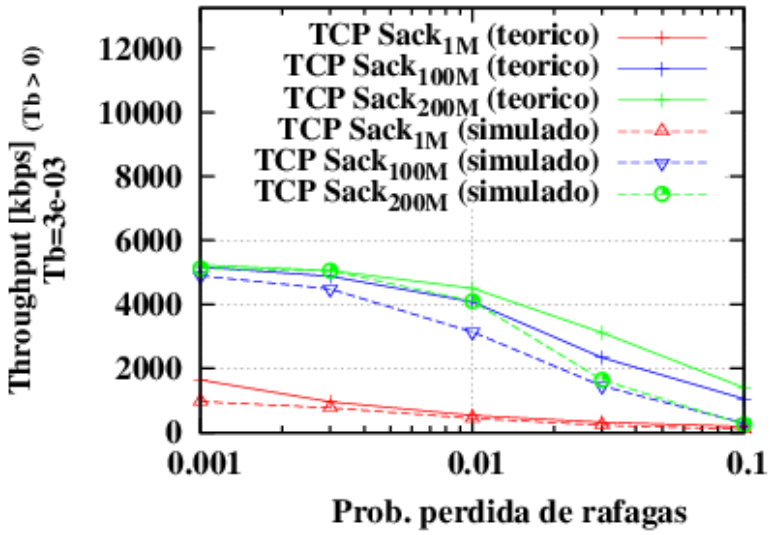
Figura 4.36 Error experimental para TCP Newreno con ACK retardado, para T_b=1 ms - 30ms y p_b=0.01

Tabla 4.4 Comparativa del throughput de TCP Newreno con y sin ACK retardado, para T_b=3 ms y p_b=0.01

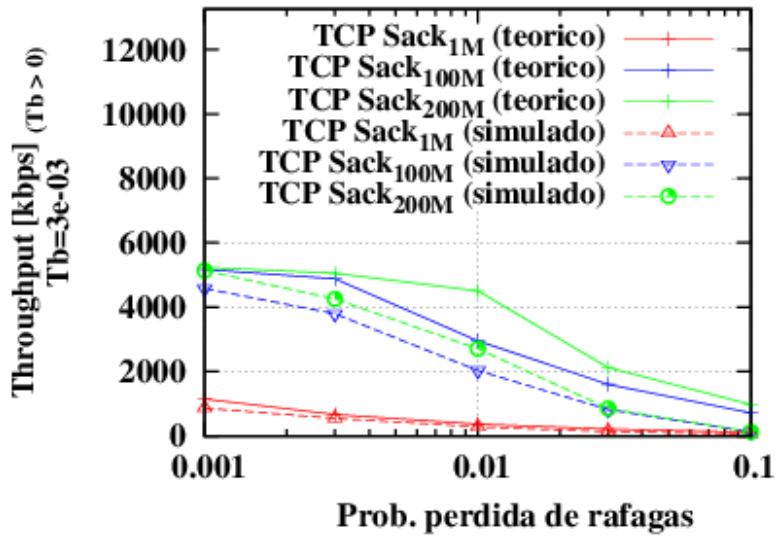
Variante de TCP	T_b	p_b	*Throughput* teórico [kbps]	*Throughput* simulado [kbps]	Clase	Ancho de banda de acceso [Mbps]	Error experimental%
Newreno	0,003	0,01	4.082,95	2.533,697	media	100	37,94%
Newreno	0,003	0,01	4.507,438	4.100,404	rápida	200	9,03%
Newreno DelayAck	0,003	0,01	1.745,559	1.756,512	media	100	0,63%
Newreno DelayAck	0,003	0,01	4.506,823	2.613,715	rápida	200	42,01%

De manera similar que en el caso anterior, se puede evidenciar el efecto de ganancia DFL (ver Figura 4.30), que se experimenta en mayor medida para fuentes de clase rápida, al comparar el *throughput* de TCP sobre OBS con una red de conmutación de paquetes (ver Figura 4.29). Por otro lado, se puede apreciar también que el *throughput* de TCP presenta sus mayores valores dentro del rango de tiempos de ensamblado entre 3 ms y 30 ms (ver Figura 4.31) para el caso de fuentes de clase media y rápida. Finalmente, el resultado de la distribución del número de ráfagas de datos y la evolución de la ventana de transmisión de TCP (ver Figuras 4.33 y 4.34) presenta un comportamiento similar al caso de TCP Reno, pero con valores superiores.

3) Resultados para TCP SACK

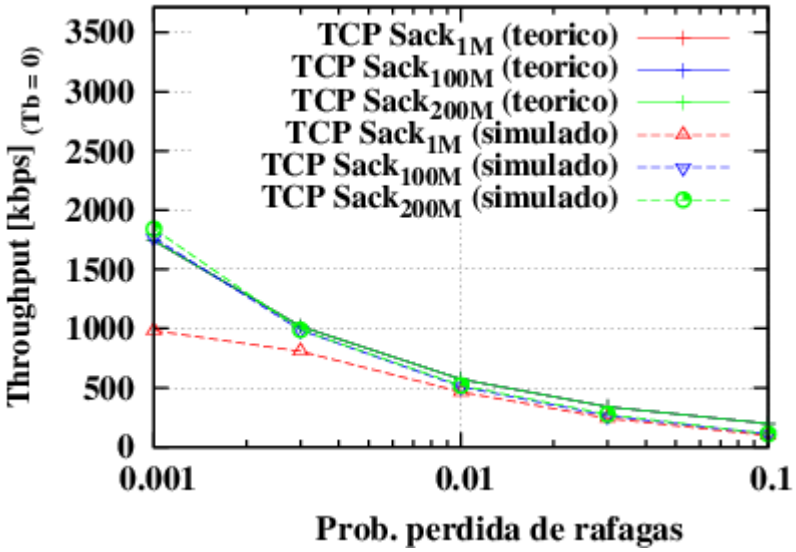


(*a*)

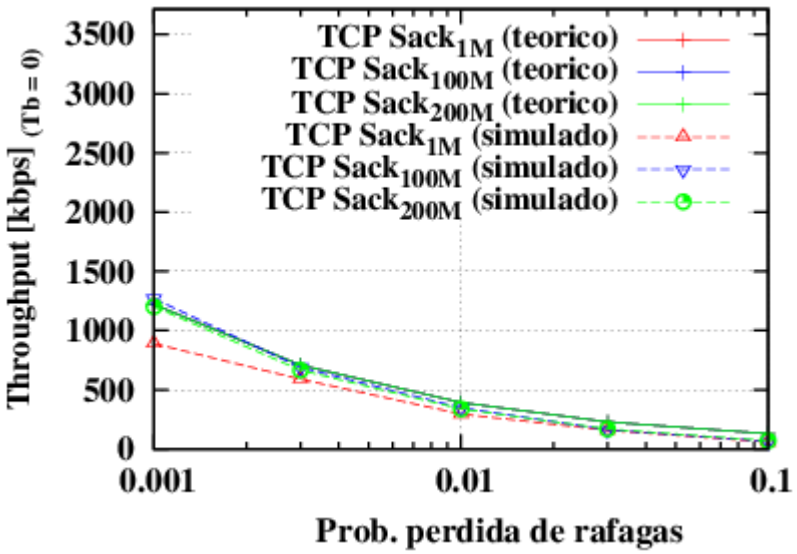


(*b*)

Figura 4.37 Throughput de TCP SACK vs probabilidad de pérdida de ráfagas para fuentes de 1 Mbps (clase lenta); 100 Mbps (clase media); y 200Mbps (clase rápida), con RTT=106 ms, W_{max}=128, L=512 bytes, y T_b=3 ms; sin ACK retardado (a) y con ACK retardado (b)

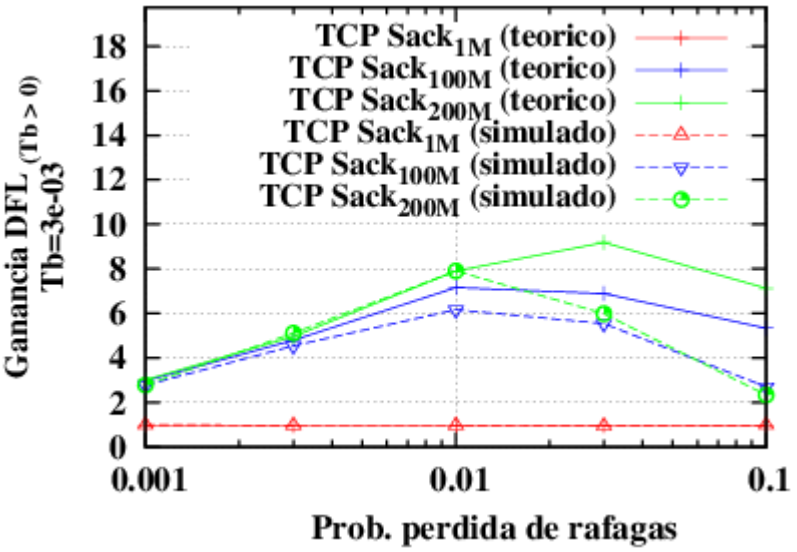


(*a*)

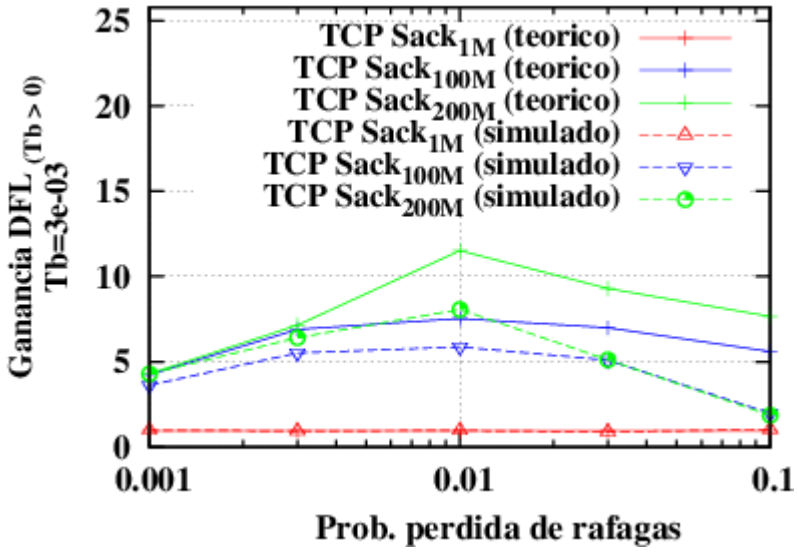


(*b*)

Figura 4.38 Throughput de TCP SACK vs probabilidad de pérdida de ráfagas para fuentes de 1 Mbps (clase lenta); 100 Mbps (clase media); y 200Mbps (clase rápida), con RTT=106 ms, W_{max}=128, L=512 bytes, y T_b=0 ms; sin ACK retardado (a) y con ACK retardado (b)

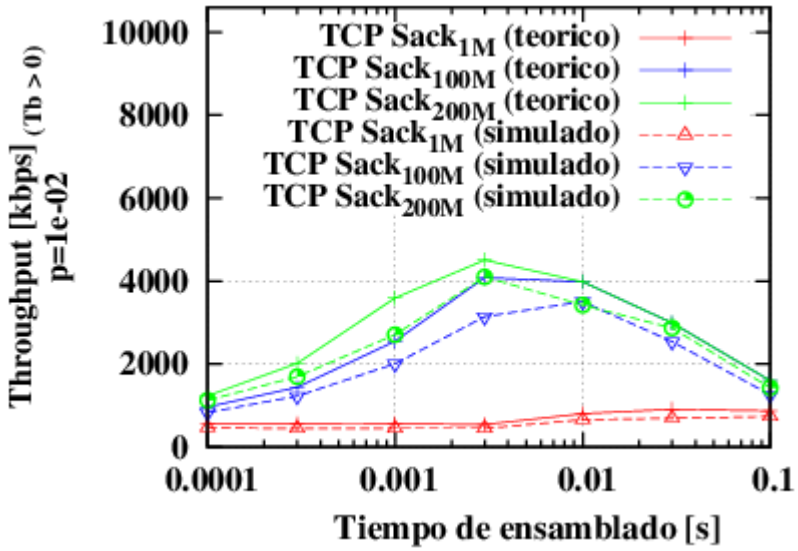


(*a*)

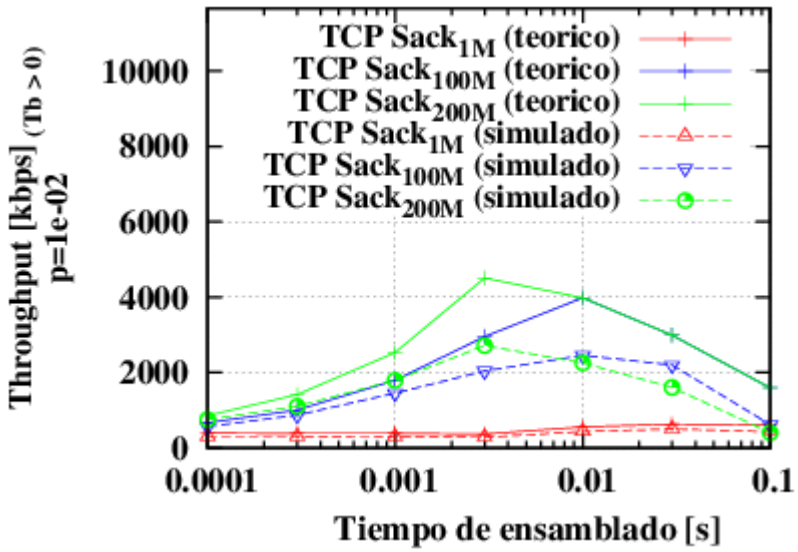


(*b*)

Figura 4.39 Ganancia DFL de TCP SACK vs probabilidad de pérdida de ráfagas para fuentes de 1 Mbps (clase lenta); 100 Mbps (clase media); y 200Mbps (clase rápida), con RTT=106 ms, W_{max}=128, L=512 bytes y T_b=3 ms; sin ACK retardado (a) y con ACK retardado (b)

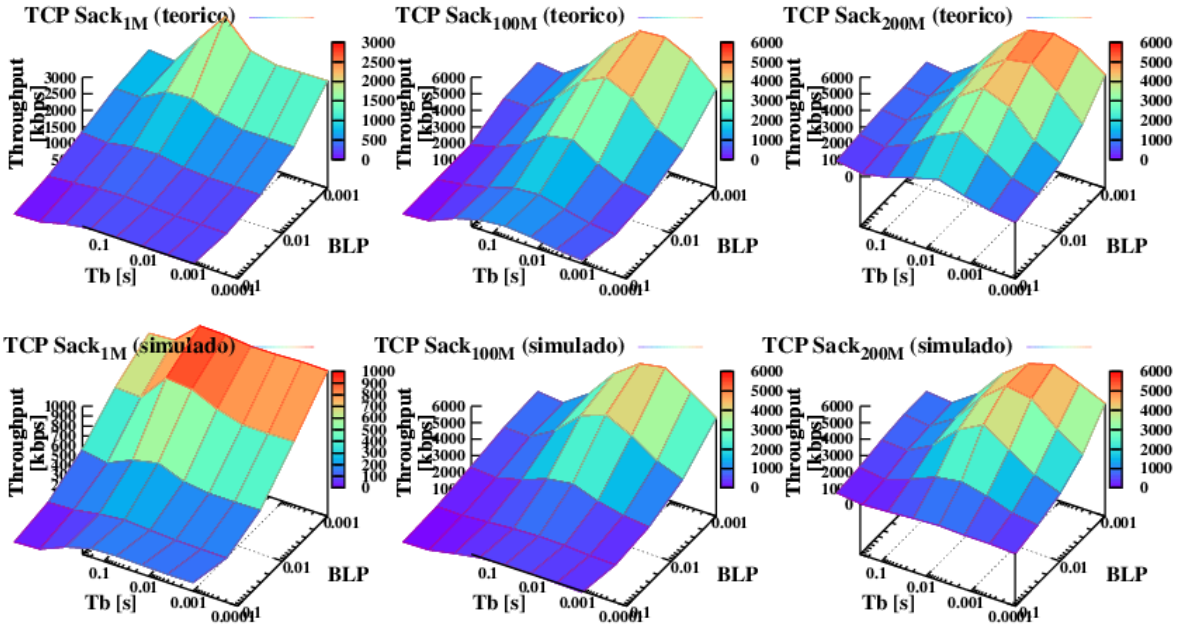


(*a*)

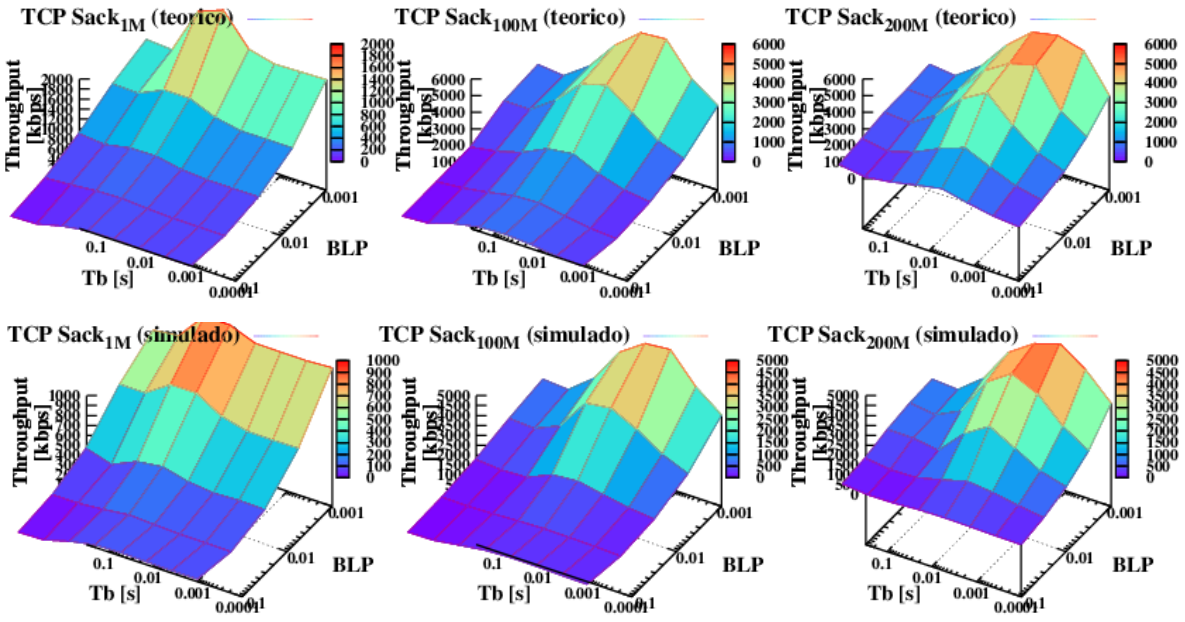


(*b*)

Figura 4.40 Throughput de TCP SACK vs tiempo de ensamblado para fuentes de 1 Mbps (clase lenta); 100 Mbps (clase media); y 200Mbps (clase rápida), con RTT=106 ms, W_{max}=128, L=512 bytes y p_b=0.01; sin ACK retardado (a) y con ACK retardado (b)

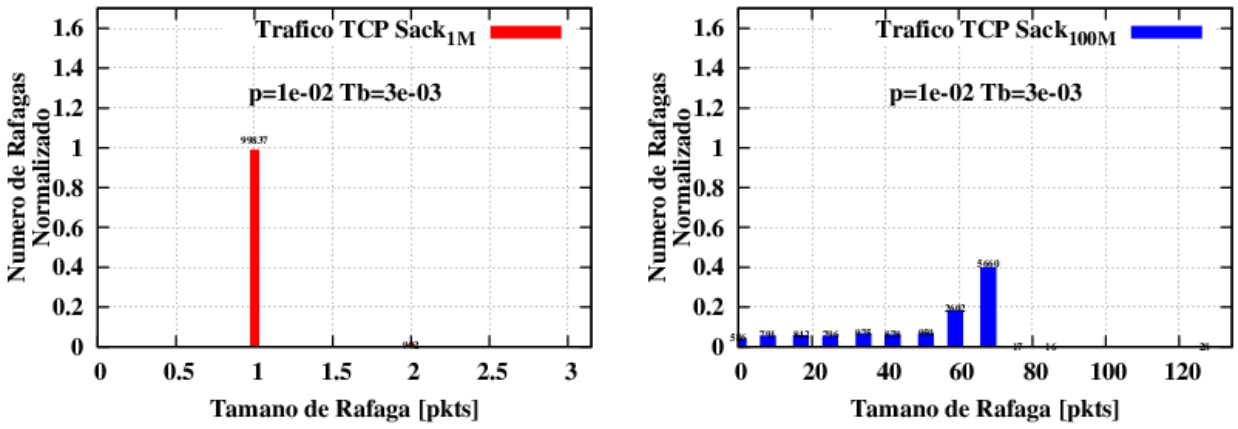


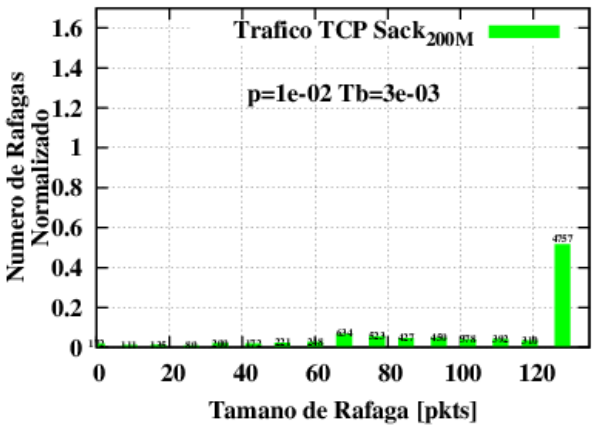
(*a*)



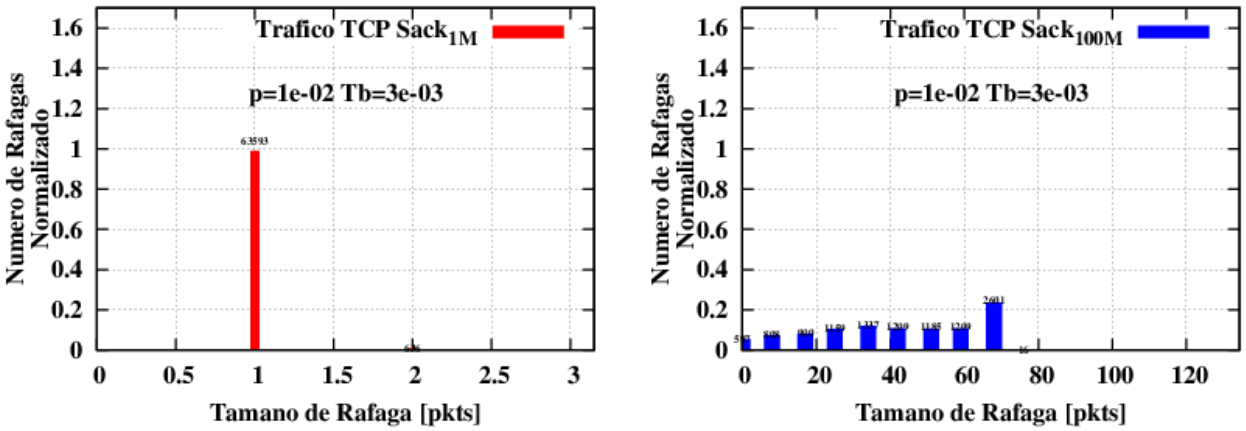
(*b*)

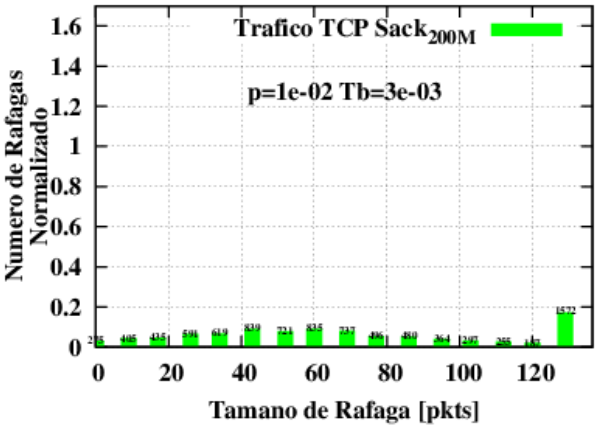
Figura 4.41 Throughput de TCP SACK con resultados analíticos y simulados, para diferentes valores de probabilidad de pérdida de ráfagas y tiempos de ensamblado; sin ACK retardado (a) y con ACK retardado (b)





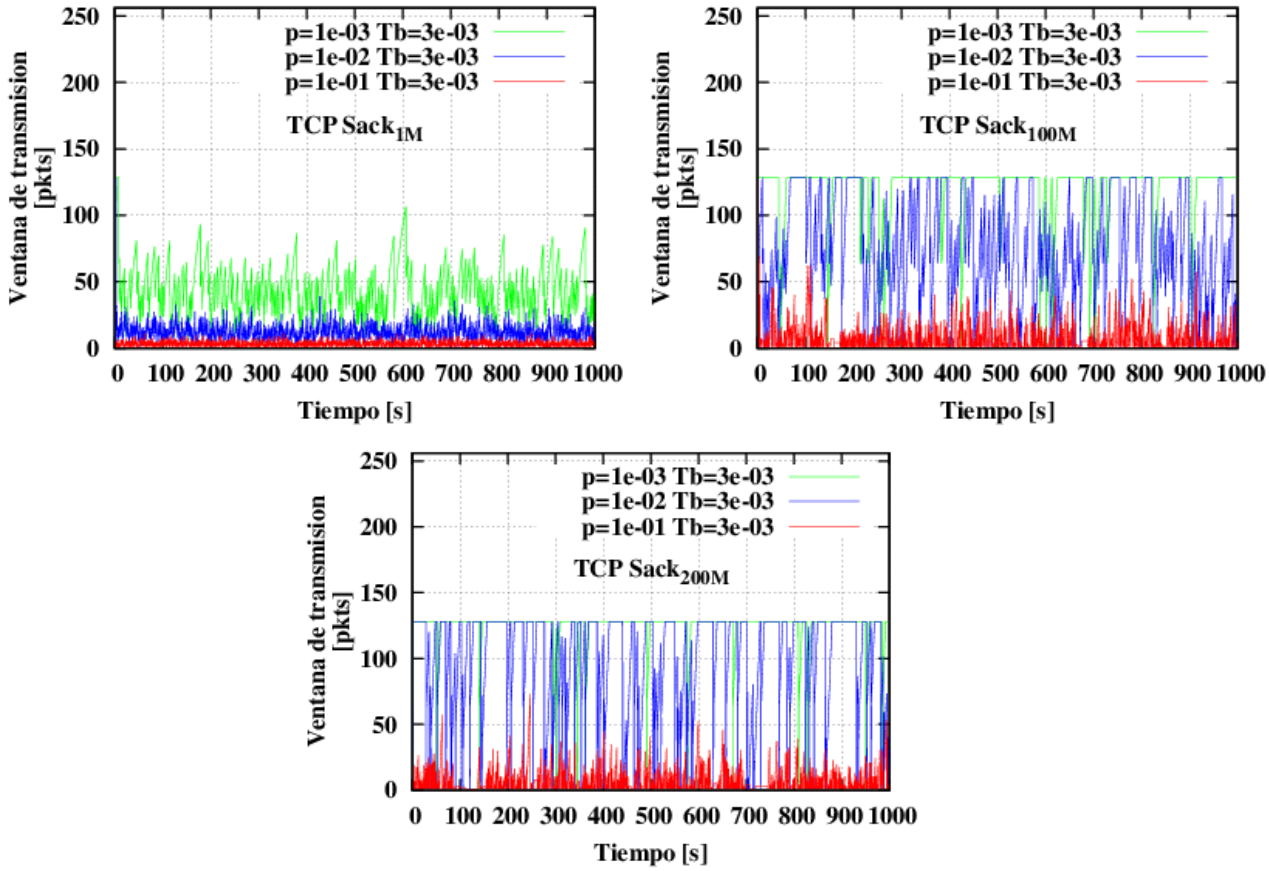
(*a*)



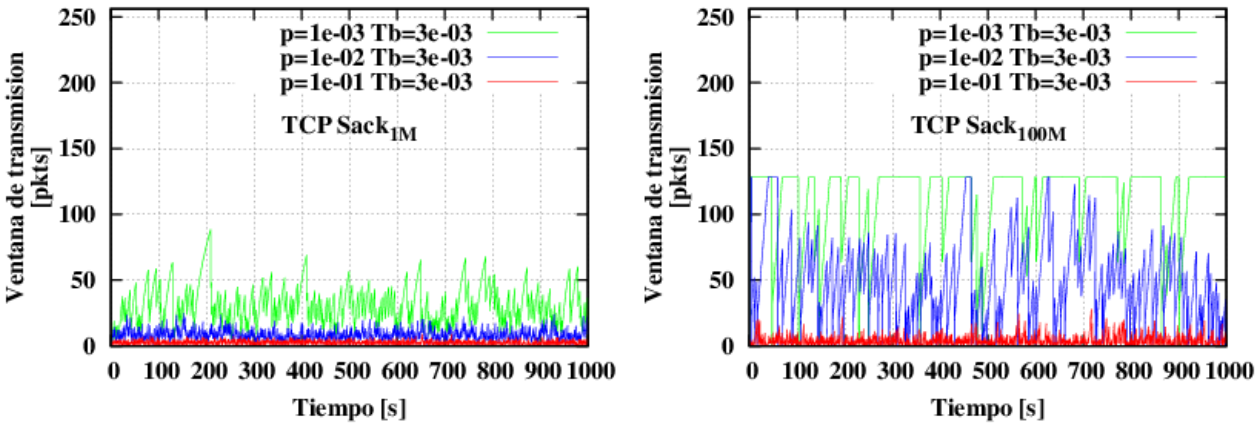


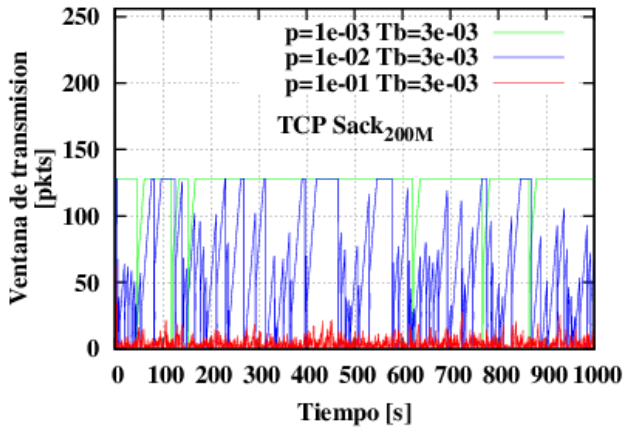
(*b*)

Figura 4.42 Distribución del tamaño de ráfaga de TCP SACK para para fuentes de 1 Mbps (clase lenta); 100 Mbps (clase media); y 200Mbps (clase rápida), con RTT= 106 ms, W_{max}=128, L=512 bytes, p_b=0.01 y T_b=3ms; sin ACK retardado (a) y con ACK retardado (b)


p=1e-03 Tb=3e-03
p=1e-02 Tb=3e-03
p=1e-01 Tb=3e-03
TCP Sack1M
TCP Sack100M
TCP Sack200M
Ventana de transmision [pkts]
Tiempo [s]
0
50
100
150
200
250
0 100 200 300 400 500 600 700 800 900 1000

(*a*)


p=1e-03 Tb=3e-03
p=1e-02 Tb=3e-03
p=1e-01 Tb=3e-03
TCP Sack1M
TCP Sack100M
Ventana de transmision [pkts]
Tiempo [s]
0
50
100
150
200
250
0 100 200 300 400 500 600 700 800 900 1000



(*b*)

Figura 4.43 Evolución del tamaño de la ventana de transmisión de TCP SACK para fuentes de 1 Mbps (clase lenta); 100 Mbps (clase media); y 200Mbps (clase rápida), con RTT= 106 ms, W_{max}=128, T_b=3ms y diferentes probabilidades de pérdida; sin ACK retardado (a) y con ACK retardado (b)

Los resultados obtenidos para el caso de TCP SACK, teniendo en cuenta los parámetros definidos anteriormente para el análisis, muestran que el modelo analítico del *throughput* para esta variante de TCP sobre redes OBS, presenta un buen nivel de aproximación (ver Figuras 4.37 y 4.41) para el caso sin ACK retardado, considerando el tiempo empleado para la simulación, con un error experimental inferior al 22% en la mayoría de los casos; a diferencia de la alternativa con ACK retardado que experimenta un error experimental inferior al 60% en la mayoría de los casos; con una reducción en el *throughput* de aproximadamente el 51% y 59% para el caso de fuentes medias y rápidas, respectivamente, al comparar ambos casos como se puede apreciar en la Tabla 4.5. Con respecto al incremento significativo del error experimental para el caso con ACK retardado, se debe indicar que el modelo analítico sobreestima el *throughput* de TCP para el caso de fuentes de clase media y rápida bajo condiciones de pérdidas mayores a 0.01, lo cual se puede evidenciar claramente en la Figura 4.37b.

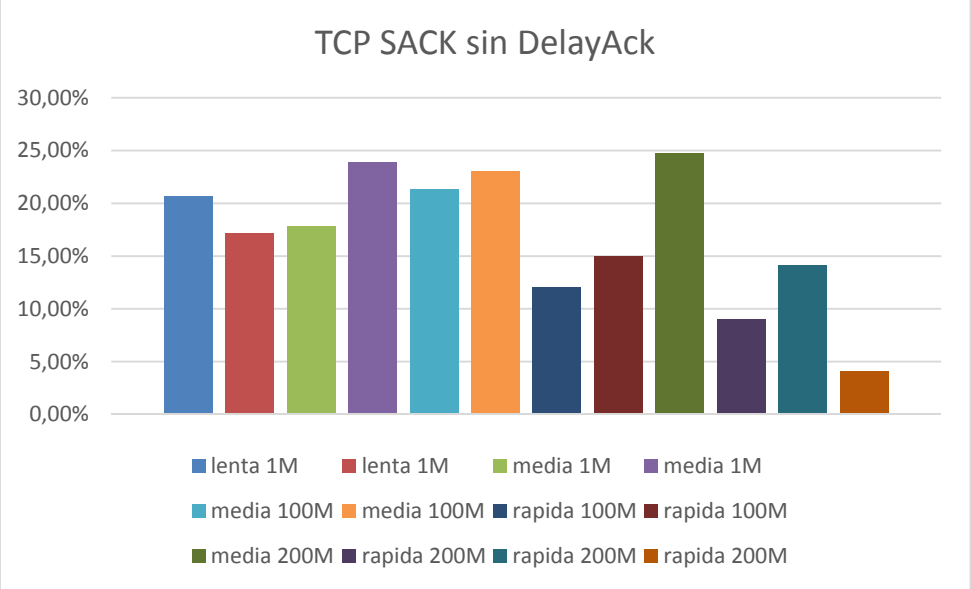


Figura 4.44 Error experimental para TCP SACK sin ACK retardado, para T_b=1 ms - 30ms y p_b=0.01

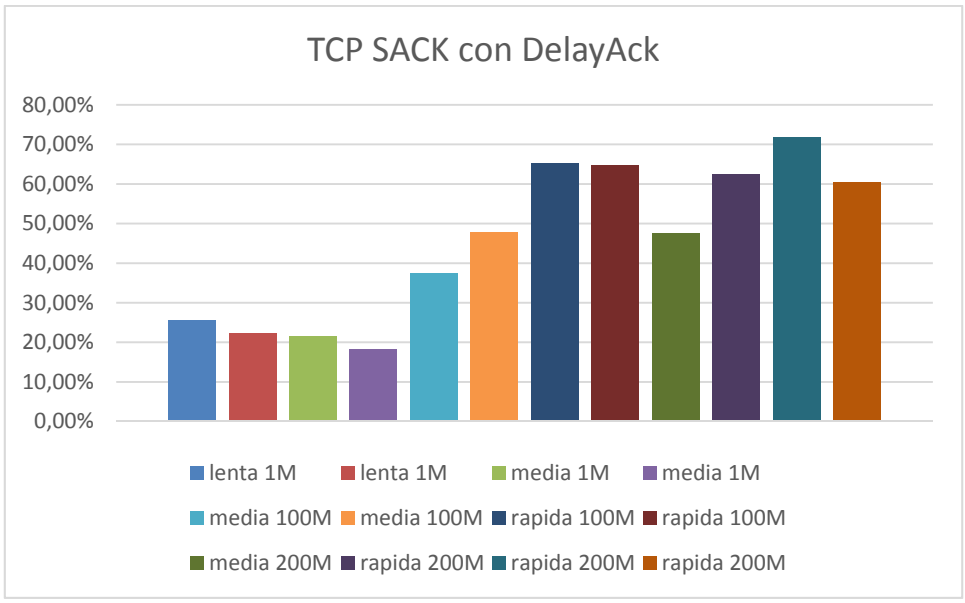


Figura 4.45 Error experimental para TCP SACK con ACK retardado, para T_b=1 ms - 30ms y p_b=0.01

Tabla 4.5 Comparativa del throughput de TCP Newreno con y sin ACK retardado para T_b=0.003 y p_b=0.01

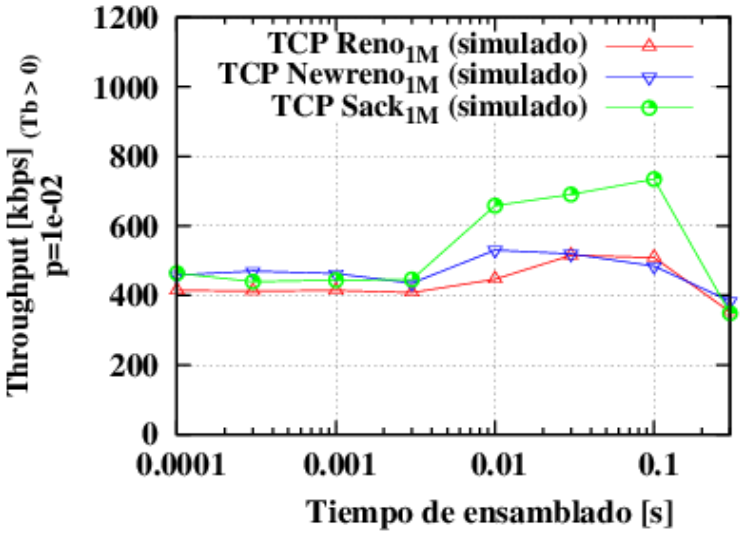
Variante de TCP	T_b	p_b	*Throughput* teórico [kbps]	*Throughput* simulado [kbps]	Clase	Ancho de banda de acceso [Mbps]	Error experimental%
SACK	0,003	0,01	4.082,951	3.140,947	media	100	23,07%
SACK	0,003	0,01	4.507,438	4.100,404	rápida	200	9,03%
SACK DelayAck	0,003	0,01	2.950,02	1.540,53	media	100	47,78%
SACK DelayAck	0,003	0,01	4.506,994	1.689,255	rápida	200	62,52%

De manera similar que en los casos anteriores, se puede evidenciar el efecto de ganancia DFL (ver Figura 4.39), que se experimenta en mayor medida para fuentes de clase rápida, al comparar el *throughput* de TCP sobre OBS con una red de conmutación de paquetes (ver Figura 4.38). Por otro lado, se puede apreciar también que el *throughput* de TCP presenta sus mayores valores dentro del rango de tiempos de ensamblado entre 3 ms y 30 ms (ver Figura 4.40) para el caso de fuentes de clase media y rápida. Finalmente, el resultado de la distribución del número de ráfagas de datos y la evolución de la ventana de transmisión de TCP (ver Figuras 4.42 y 4.43) presenta un comportamiento similar a los casos de TCP Reno y Newreno, pero con valores superiores.

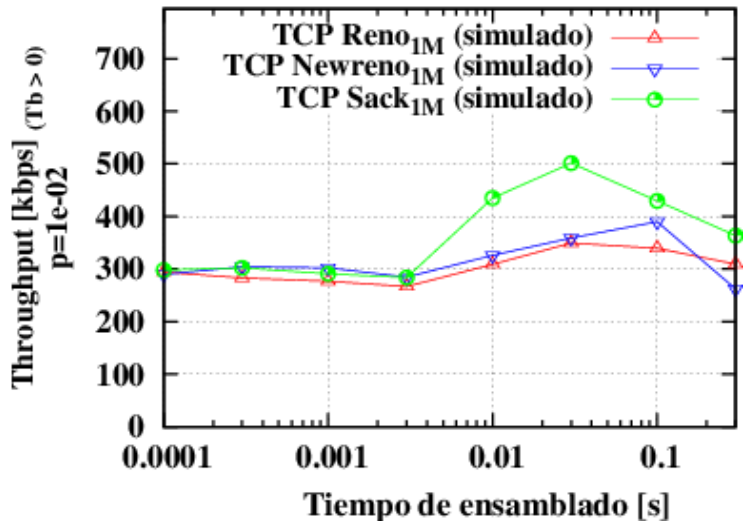
4) Comparativa variantes de TCP

Por último, al comparar los resultados de las tres variantes de TCP basadas en pérdidas, se puede observar en las Figuras 4.46, 4.47 y 4.48 que: 1) para el caso de fuentes de clase lenta, en el rango de tiempos de ensamblado inferiores a 3 ms TCP SACK y New Reno presentan un comportamiento comparable con un *throughput* mayor al de TCP Reno, mientras que de manera general para un tiempo de ensamblado mayor a 3 ms, TCP SACK presenta un mayor *throughput* con la particularidad de que en un rango mayor a 30 ms, TCP Reno y New Reno se comportan de manera similar; 2) para el caso de fuentes de clase media, en el rango de tiempos de ensamblado entre 0.1 ms y 3 ms, TCP Reno y New Reno presentan un *throughput* semejante, con un valor mayor por parte de TCP SACK, mientras que en un rango mayor a 3 ms, las tres

variantes tienen un *throughput* similar, con un mayor por parte de TCP SACK entre 3 ms y 30 ms para el caso de ACK retardado; y 3) para el caso de fuentes rápidas en el rango de tiempos de ensamblado entre 0.1 ms a 3 ms, el *throughput* de TCP Reno y New Reno es semejante pero TCP SACK es mayor; mientras que en un rango mayor a 3 ms las tres variantes son similares, con TCP Reno ligeramente mayor en el rango a partir de 30 ms para el caso de ACK retardado.

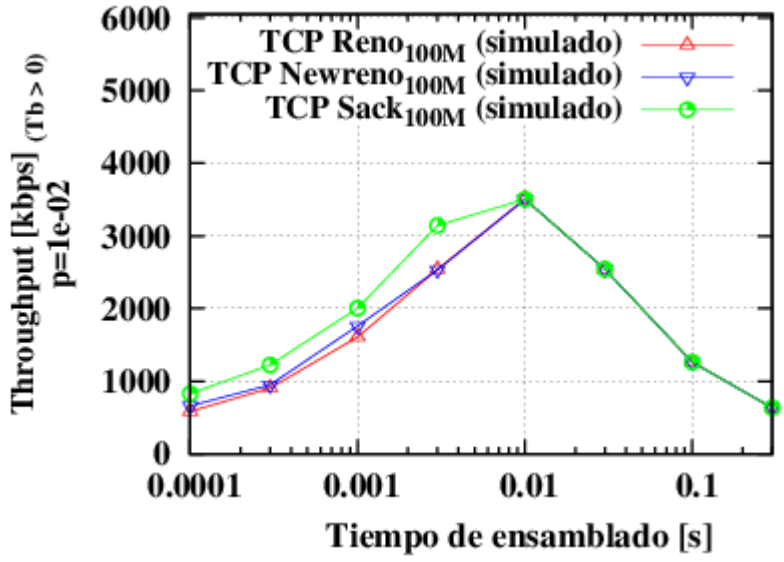


(*a*)

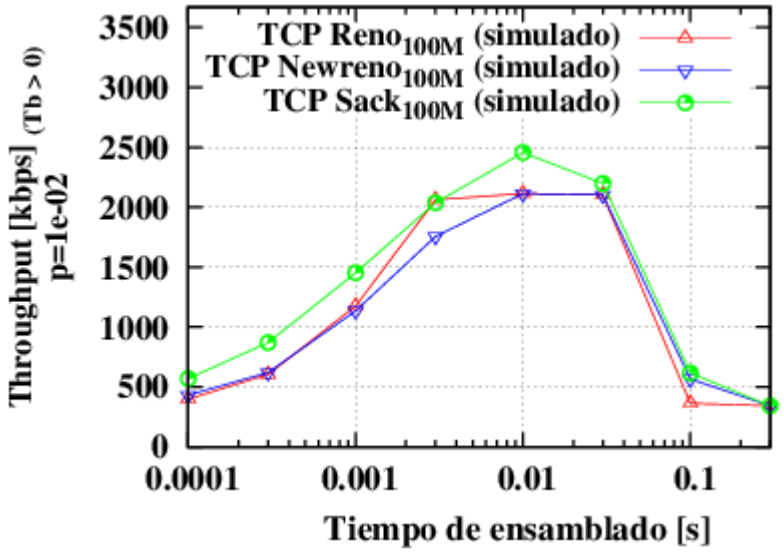


(*b*)

Figura 4.46 Comparación del throughput entre TCP Reno, Newreno y Sack para una fuente de 1 Mbps (clase lenta), con RTT= 106 ms, W_{max}=128, p_b=0.01 y diferentes tiempos de ensamblado: sin ACK retardado (a) y con ACK retardado (b)

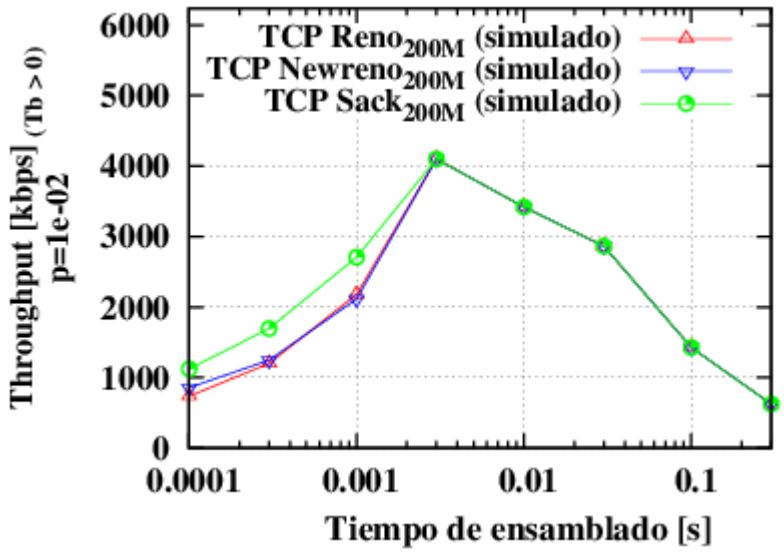


(*a*)

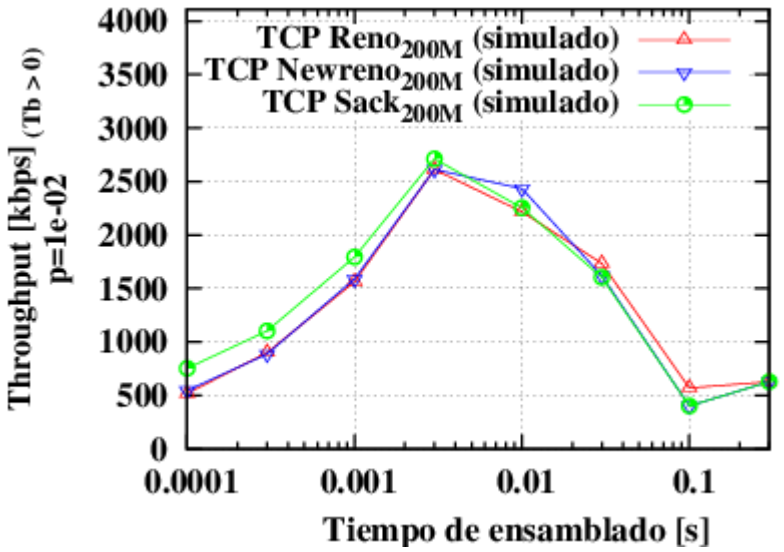


(*b*)

Figura 4.47 Comparación del throughput entre TCP Reno, Newreno y Sack para una fuente de 100 Mbps (clase media), con RTT= 106 ms, W_{max}=128, p_b=0.01 y diferentes tiempos de ensamblado; sin ACK retardado (a) y con ACK retardado (b)



(*a*)



(*b*)

Figura 4.48 Comparación del throughput entre TCP Reno, Newreno y SACK para una fuente de 200 Mbps (clase rápida), con RTT= 106 ms, W_{max}=128, p_b=0.01 y diferentes tiempos de ensamblado; sin ACK retardado (a) y con ACK retardado (b)

CONCLUSIONES Y RECOMENDACIONES

5.1 Conclusiones

- La conmutación óptica de ráfagas podría ser una alternativa candidata a una solución inicial para el Internet Óptico de nueva generación en el corto y mediano plazo, mientras se alcanza un desarrollo y grado de madurez en el paradigma de la conmutación óptica de paquetes que podría lograrse en el largo plazo.
- Las variantes de TCP utilizadas comúnmente hoy en día dentro de redes de conmutación de paquetes, no son apropiadas para el esquema de redes OBS dado que presentan muchas limitaciones en cuanto al crecimiento de la ventana de congestión y mecanismos de recuperación de pérdidas, que finalmente tienen un impacto importante sobre el *throughput*, por lo cual, en caso de que el paradigma de OBS llegue a implementarse en un ambiente real, se debe prestar mucha atención a las alternativas y propuestas de otras versiones del protocolo TCP que se adapten de mejor manera al elevado producto ancho de banda por retardo, que puede proporcionar esta tecnología en desarrollo.
- Existe un compromiso entre los distintos parámetros de una red OBS cuando se utiliza TCP como protocolo de transporte, que impactan tanto positiva como negativamente su *throughput*, alcanzando un punto óptimo donde este parámetro puede ser maximizado, y que depende en gran medida del tiempo de ensamblado de ráfagas y de la probabilidad de pérdida de ráfagas.
- El modelo matemático basado en los procesos regenerativos de Markov definido para el caso de flujos de clase lenta y rápida, establece funciones límite con umbrales mínimo y máximo dentro de los cuales varía el *throughput* de TCP, el mismo que fue validado mediante simulación y en contraste con su contraparte analítica, se aproxima en gran medida para el caso de fuentes de clase lenta y rápida, y presenta valores intermedios entre éstas para el caso de fuentes de clase media.

- El modelo matemático basado en la teoría de renovación definido para una fuente genérica, establece un comportamiento variable en cuanto al *throughput* de TCP de acuerdo a la clase de fuente utilizada, el mismo que fue validado mediante simulación y en contraste con su contraparte analítica, presenta un buen nivel de aproximación para el caso de pérdidas pequeñas (< 0.01), mientras que para el caso de pérdidas mayores, el modelo analítico sobrestima el *throughput* de TCP.

- El *throughput* de TCP experimenta una disminución significativa cuando se utiliza la opción de ACK retardado, debido a que se reduce la tasa de transmisión de ACKs, resultando en el envío de un menor número de paquetes por ráfaga.

- Para las versiones de TCP analizadas, fue posible comprobar que SACK ofrece normalmente un mayor *throughput* que las demás variantes, dado el esquema de reconocimiento selectivo que le permite recuperase más eficientemente de las pérdidas que podrían presentarse en la red.

- De manera general, el *throughput* de TCP para las diferentes variantes analizadas difiere debido a los distintas penalidades antes mencionadas, que se pueden experimentar dado el comportamiento heterogéneo que presentan estas versiones del protocolo TCP, producto de su variación en cuanto a las fases de *fast retransmit* y *fast recovery*.

- El simulador de redes OBS-ns ofrece una arquitectura interesante para incursionar en esta nueva tecnología que se encuentra en desarrollo e investigación, y es adaptable a cambios y nuevas funcionalidades que se podrían implementar como un paso adicional a este proyecto de tesis, para evaluar las diferentes características de esta tecnología que no han sido consideradas en este estudio, como el esquema de conversión de longitud de onda, mecanismos propios de recuperación de pérdidas, algoritmos de planificación y señalización, y otras versiones de TCP que se describen en la literatura; así como también nuevas alternativas que se podrían desarrollar en un futuro.

5.2 Recomendaciones

- Para obtener mejores resultados que permitan analizar con mayor detalle los modelos analíticos definidos anteriormente, se recomienda utilizar un hardware especializado para simulación a través de supercomputadoras, u otro esquema como el *grid computing*, para profundizar en el comportamiento del tráfico en escenarios más complejos, que de lo contrario incurrirían en tiempos demasiado extensos de simulación que serían prácticamente inaceptables.
- Para la evaluación de esta tecnología en desarrollo mediante herramienta de simulación, se recomienda profundizar previamente en cuanto a las técnicas y algoritmos de programación tanto en los lenguajes compilados, interpretados y de análisis posterior, así como también de un estudio apropiado de la teoría de la probabilidad, variables aleatorias y procesos estocásticos.

REFERENCIAS BIBLIOGRÁFICAS

[1] Agrawal, G. (2013). Nonlinear fiber optics

[2] Al Amin, A., Nishimura, K., Shimizu, K., Takenaka, M., Tanemura, T., Onaka, H., ... & Urino, Y. (2009). Development of an optical-burst switching node testbed and demonstration of multibit rate optical burst forwarding. *Journal of Lightwave Technology*, *27*(16), 3466-3475.

[3] Allman, M., Paxson, V., & Blanton, E. (2009). *TCP congestion control* (No. RFC 5681).

[4] Alzate, D. F., & Cárdenas, A. "Retos en la transmisión de 40/100 Gbps sobre fibra óptica", *Revista en telecomunicaciones e informatica*, (2011) *1*(2).

[5] Aracil, J., & Callegati, F. (Eds.). (2009). *Enabling optical internet with advanced network technologies*. Springer Science & Business Media.

[6] Arima, S., Tachibana, T., Yuichi, K. A. J. I., & Kasahara, S. (2007). FEC-Based reliable transmission for multiple bursts in OBS networks. *IEICE transactions on communications*, *90*(12), 3541-3551.

[7] Baldine, I., Cassada, M., Bragg, A., Karmous-Edwards, G., & Stevenson, D. (2003, December). Just-in-time optical burst switching implementation in the ATDnet all-optical *networking* testbed. In *Global Telecommunications Conference, 2003. GLOBECOM'03. IEEE* (Vol. 5, pp. 2777-2781). IEEE.

[8] Banerjee, S., & Sarkar, D. (2001). Hypercube connected rings: a scalable and fault-tolerant logical topology for optical networks. *Computer communications*, *24*(11), 1060-1079.

[9] Basha, M. (2007). Optical MEMS switches: theory, design, and fabrication of a new architecture.

[10] Battestilli, T. and Perros, H. (2005). Optical burst switching for the next generation internet, IEEE Potentials 23(5): 40-43. Battestilli, T., & Perros, H. (2004). Optical burst switching for the next generation Internet. *IEEE Potentials*, *23*(5), 40-43.

[11] Blanton, E., Allman, M., Wang, L., Jarvinen, I., Kojo, M., & Nishida, Y. (2012). *A Conservative Loss Recovery Algorithm Based on Selective Acknowledgment (SACK) for TCP* (No. RFC 6675).

[12] Bonetto, E., Chiaraviglio, L., Cuda, D., Castillo, G. A. G., & Neri, F. (2009, September). Optical technologies can improve the energy efficiency of networks. In *Optical Communication, 2009. ECOC'09. 35th European Conference on* (pp. 1-4). IEEE.

[13] Braden, R. (1989). Requirements for Internet hosts-communication layers.", RFC 1122, octubre 1989

[14] Brakmo, L. S., & Peterson, L. L. (1995). TCP Vegas: End to end congestion avoidance on a global Internet. *IEEE Journal on selected Areas in communications*, *13*(8), 1465-1480.

[15] Bregni, S., Guerra, G. I. A. C. O. M. O., & Pattavina, A. (2001). State of the art of optical switching technology for all-optical networks. *Communications World*.

[16] Brunn, Ines. "DWDM Dense Wavelength Division Multiplexing Pocked Guide JDSU". Recuperado del 1 de noviembre de 2015, de http://araneatechnology.cz/wp-content/uploads/2015/12/dwdm-pg-fop-tm-ae-1005.pdf

[17] Bucci Sam. "The 400G Photonic Service Engine" (2012). Recuperado el 27 de febrero de 2017, de https://insight.nokia.com/400g-photonic-service-engine

[18] Buchta, H. (2005). Analysis of physical constraints in an optical burst switching network.

[19] CAMERON, C., Zalesky, A., & Zukerman, M. (2005). Prioritized deflection routing in optical burst switching networks. *IEICE transactions on communications*, *88*(5), 1861-1867.

[20] Casoni, M., & Raffaelli, C. (2007). Analytical framework for end-to-end design of optical burst-switched networks. *Optical switching and Networking*, *4*(1), 33-43.

[21] Charcranoon, S., El-Bawab, T. S., Cankaya, H. C., & Shin, J. D. (2003, December). Group-scheduling for optical burst switched (OBS) networks. In *Global Telecommunications Conference, 2003. GLOBECOM'03. IEEE* (Vol. 5, pp. 2745-2749). IEEE.

[22] Chen, Y., Qiao, C., & Yu, X. (2004). Optical burst switching: a new area in optical *networking* research. *IEEE network*, *18*(3), 16-23.

[23] Chu, J., Cheng, Y., Dukkipati, N., & Mathis, M. (2013). Increasing TCP's initial window. RFC 6928, IETF Network Working Group.

[24] Chua, K. C., Gurusamy, M., Liu, Y., & Phung, M. H. (2007). *Quality of service in optical burst switched networks*. Springer Science & Business Media.

[25] Detti, A., & Listanti, M. (2002). Impact of segments aggregation on TCP Reno flows in optical burst switching networks. In *INFOCOM 2002. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE* (Vol. 3, pp. 1803-1812). IEEE.

[26] Detti, A., Eramo, V., & Listanti, M. (2002). Performance evaluation of a new technique for IP support in a WDM optical network: optical composite burst switching (OCBS). *Journal of Lightwave Technology*, *20*(2), 154-165.

[27] do Júri, C. (2009). *Quality of service in optical burst switching networks* (Doctoral dissertation, Universidade do Algarve).

[28] Dukkipati, N., Refice, T., Cheng, Y., Chu, J., Herbert, T., Agarwal, A., ... & Sutin, N. (2010). An argument for increasing TCP's initial congestion window. *Computer Communication Review*, *40*(3), 26-33.

[29] El-Bawab, T. S. (2006). Optical Switching.

[30] Fall, K. R., & Stevens, W. R. (2011). *TCP/IP illustrated, volume 1: The protocols*. addison-Wesley.

[31] Fall, K., & Varadhan, K. (2005). The ns Manual (formerly ns Notes and Documentation). The VINT project, 47.

[32] Farahmand, F., & Jue, J. P. (2003, June). Look-ahead window contention resolution in optical burst switched networks. In *High Performance Switching and Routing, 2003, HPSR. Workshop on* (pp. 147-151). IEEE.

[33] Floyd, S., Mahdavi, J., Mathis, M., & Podolsky, M. (2000). *An extension to the selective acknowledgement (SACK) option for TCP* (No. RFC 2883).

[34] Floyd, S., Balakrishnan, H., & Allman, M. (2001). Enhancing TCP's loss recovery using limited transmit.

[35] Floyd, S. (2003). HighSpeed TCP for large congestion windows, RFC 3649.

[36] Francoy, J. C., & Tamarit, B. O. (2006). *Redes ópticas*. Ed. Univ. Politéc. Valencia.

[37] Garcia, N. M. (2008). Architectures and algorithms for ipv4/ipv6-compliant optical burst switching networks. (PhD Thesis, University of Beira Interior).

[38] Gauger, C. M. (2003). Dimensioning of FDL *buffer*s for optical burst switching nodes. In *Next Generation Optical Network Design and Modelling* (pp. 117-132). Springer US.

[39] Ge, A., Callegati, F., & Tamil, L. S. (2000). On optical burst switching and self-similar traffic. *IEEE Communications Letters*, *4*(3), 98-100.

[40] Gerla, M., Sanadidi, M. Y., Wang, R., Zanella, A., Casetti, C., & Mascolo, S. (2001). TCP Westwood: Congestion window control using bandwidth estimation. In *Global Telecommunications Conference, 2001. GLOBECOM'01. IEEE* (Vol. 3, pp. 1698-1702). IEEE.

[41] González de Dios, Ó. (2012). Rendimiento de TCP y Cálculo de Rutas en Redes de Conmutación Óptica de Ráfagas (PhD Tesis, Universidad de Valladolid).

[42] González, Ó., de Miguel, I., Merayo, N., Fernández, P., Lorenzo, R. M., & Abril, E. J. (2005). The impact of delayed ACK in TCP flows in OBS networks. *Proceedings of NOC 2005*, 367-374.

[43] Guillemot, C., Renaud, M., Gambini, P., Janz, C., Andonovic, I., Bauknecht, R., ... & Chiaroni, D. (1998). Transparent optical packet switching: The European ACTS KEOPS project approach. *Journal of lightwave technology*, *16*(12), 2117-2134.

[44] Guo, H., Wu, J., Lan, Z., Gao, Z., Li, X., Lin, J., ... & Li, X. (2005, March). A Testbed for Optical Burst Switching Networks. In *Optical Fiber Communication Conference* (p. OFA6). Optical Society of America.

[45] Gwynne, Peter. "MEMS enables fast, reliable optical switching" (2000). Recuperado el 1 de noviembr de 2015, de http://spie.org/newsroom/mems-enables-fast-reliable-optical-switching

[46] Henderson, T., Floyd, S., Gurtov, A., & Nishida, Y. (2012). *The NewReno modification to TCP's fast recovery algorithm* (No. RFC 6582).

[47] Heron, R. W., Pfeiffer, T., van Veen, D. T., Smith, J., & Patel, S. S. (2008). Technology innovations and architecture solutions for the next-generation optical access network. *Bell Labs Technical Journal*, *13*(1), 163-181.

[48] Hong, X., Guo, H., Wu, J., Yin, Y., Liu, L., Zuo, Y., ... & Tsuritani, T. (2009, September). Testbed of OBS/GMPLS interworking. In *Broadband Communications, Networks, and Systems, 2009. BROADNETS 2009. Sixth International Conference on* (pp. 1-6). IEEE.

[49] Huang, X., Vokkarane, V. M., & Jue, J. P. (2005, May). Burst cloning: a proactive scheme to reduce data loss in optical burst-switched networks. In *Communications, 2005. ICC 2005. 2005 IEEE International Conference on* (Vol. 3, pp. 1673-1677). IEEE.

[50] HubTechInsider in Telecommunications. "What's the difference between a first generation and a second generation Optical Switch?" (2009). Recuperado el 1 de noviemre de 2015, de https://hubtechinsider.wordpress.com/2009/04/24/whats-the-difference-between-a-first-generation-and-a-second-generation-optical-switch/

[51] Hudek, G. C., & Muder, D. J. (1995, June). Signaling analysis for a multi-switch all-optical network. In *Communications, 1995. ICC'95 Seattle,'Gateway to Globalization', 1995 IEEE International Conference on* (Vol. 2, pp. 1206-1210). IEEE.

[52] Iizuka, M., Sakuta, M., Nishino, Y., & Sasase, I. (2002, November). A scheduling algorithm minimizing voids generated by arriving bursts in optical burst switched WDM network. In *Global Telecommunications Conference, 2002. GLOBECOM'02. IEEE* (Vol. 3, pp. 2736-2740). IEEE.

[53] Issariyakul, T., & Hossain, E. (2011). *Introduction to network simulator NS2*. Springer Science & Business Media.

[54] Iyengar, J., Caro, A., & Amer, P. (2003). Dealing with short TCP flows: A survey of mice in elephant shoes. *University of Delaware*.

[55] Izal, M., & Aracil, J. (2002, November). On the influence of self-similarity on optical burst switching traffic. In *Global Telecommunications Conference, 2002. GLOBECOM'02. IEEE* (Vol. 3, pp. 2308-2312). IEEE.

[56] Jacobson, V. (1988, August). Congestion avoidance and control. In *ACM SIGCOMM computer communication review* (Vol. 18, No. 4, pp. 314-329). ACM.

[57] Jacobson, V., Braden, R., & Borman, D. (1992). TCP extensions for high performance, RFC 1323.

[58] Jian, W., Wei, Z., & Minxue, W. (2007). Optical burst switching network testbed. In *Optical Network Design and Modeling* (pp. 166-175). Springer Berlin Heidelberg.

[59] Jue, J. P., & Vokkarane, V. M. (2005). Optical burst switched networks. *Optical networks series*.

[60] Kai Shi B.Sc., M.Eng Investigation of Wavelength Tunable Lasers for use in Coherent Optical Communication Systems (Ph.D. Dissertation)

[61] Kaminow, I. P., Li, T., & Willner, A. E. (2008). Optical Fiber Telecommunications VB: Systems and Networks.

[62] Kaminow, I. P., Li, T., & Willner, A. E. (2013). Optical Fiber Telecommunications Volume VIB, Sixth Edition: Systems and Networks (Optics and Photonics).

[63] Kitayama, K. I., Koga, M., Morikawa, H., Hara, S., & Kawai, M. (2005, March). Optical burst switching network testbed in Japan. In *Optical Fiber Communication Conference, 2005. Technical Digest. OFC/NFOEC* (Vol. 3, pp. 3-pp). IEEE.

[64] Kitayama, K., Arakawa, S., Matsuo, S., Murata, M., Notomi, M., Takahashi, R., & Itaya, Y. (2007, August). All-optical RAM-based *buffer* for packet switch. In *Photonics in Switching, 2007* (pp. 5-6). IEEE.

[65] Kitayama, K. I., Kubo, T., Takahashi, R., Matsuo, S., Arakawa, S., Murata, M., ... & Kato, K. (2011, March). All-optical RAM *buffer* subsystem demonstrator. In *Optical Fiber Communication Conference and Exposition (OFC/NFOEC), 2011 and the National Fiber Optic Engineers Conference* (pp. 1-3). IEEE.

[66] Klinkowski, M. (2008). *Offset time-emulated architecture for optical burst switching-modelling and performance evaluation*. Universitat Politècnica de Catalunya.

[67] Lee, S., Sriram, K., Kim, H., & Song, J. (2005). Contention-based limited deflection routing protocol in optical burst-switched networks. *IEEE Journal on Selected Areas in Communications*, *23*(8), 1596-1611.

[68] Leite,B., Pinto, P., Terroso, I. & Carvalho, J. (2002). IP over WDM Designing an Optical IP Router (Graduation project, Universidade do Porto).

[69] Li, L., Scott, S. D., & Deogun, J. S. (2003, October). Performance analysis of WDM optical packet switches with a hybrid *buffer*ing architecture. In *OptiComm 2003: Optical Networking and Communications* (pp. 346-356). International Society for Optics and Photonics.

[70] Li, V., Li, C. Y., & Wai, P. K. A. (2004). Alternative structures for two-dimensional MEMS optical switches [Invited]. *Journal of optical networking*, *3*(10), 742-757.

[71] LIGHT READING. "Intune Gets Another €3M" (2010). Recuperado el 1 de noviembre de 2015, de http://www.lightreading.com/document.asp?doc_id=193978

[72] LIGHT READING. "Intune goes commercial" (2011).). Recuperado el 1 de noviembre de 2015, de http://www.lightreading.com/document.asp?doc_id=208305

[73] Lu, X., & Mark, B. L. (2004). Performance modeling of optical-burst switching with fiber delay lines. *IEEE Transactions on Communications*, *52*(12), 2175-2183.

[74] Malathi. V. Modeling TCP Throughput and Delay, April 3, 2004.

[75] Maier, M. (2008). *Optical switching networks*. Cambridge University Press.

[76] Markussen, J. S. (2014). *Experimentally finding the right aggressiveness for retransmissions in thin TCP streams-How hard can it be?* (Master's thesis).

[77] Martinez-Yelmo, I., Soto, I., Larrabeiti, D., & Guerrero, C. (2007, July). A simulation-based study of TCP performance over an Optical Burst Switched *backbone* with 802.11 access. In *Meeting of the European Network of Universities and Companies in Information and Communication Engineering* (pp. 120-127). Springer Berlin Heidelberg.

[78] Mathis, M., Mahdavi, J., Floyd, S., & Romanow, A. (1996). *TCP selective acknowledgment options* (No. RFC 2018).

[79] MATSUMOTO, Craig. “Huawei Strives for Optical Respect” (2012). ”. Recuperado el 1 de noviembre de 2015, de http://www.lightreading.com/document.asp?doc_id=218487

[80] Meis, D. (2006). FTTH: The next great household amenity, Broadband Properties 2: 45-49.

[81] Monroy, I. T., Zhang, J., Chi, N., Holm-Nielsen, P. V., Peucheret, C., Koonen, A. M. J., ... & Khoe, G. D. (2003, October). Techniques for labeling of optical signals in burst switched networks. In Proc. 1st Workshop on Optical Burst Switching, Opticomm (pp. 1-11).

[82] Mukherjee, B. (2006). Optical WDM networks. Springer Science & Business Media.

[83] Nokia Siemens Networks. “Optical Transport Network Switching: Creating efficient and cost-effective optical transport networks”. Recuperado de, http://docplayer.net/18685163-Optical-transport-network-switching-creating-efficient-and-cost-effective-optical-transport-networks-white-paper.html

[84] O'Mahony, M. J., Politi, C., Klonidis, D., Nejabati, R., & Simeonidou, D. (2006). Future optical networks. *Journal of Lightwave Technology*, *24*(12), 4684-4696.

[85] Padhye, J., Firoiu, V., Towsley, D., & Kurose, J. (1998). Modeling TCP *throughput*: A simple model and its empirical validation. *ACM SIGCOMM Computer Communication Review*, *28*(4), 303-314.

[86] Papadimitriou, G. I., Papazoglou, C., & Pomportsis, A. S. (2003). Optical switching: switch fabrics, techniques, and architectures. *Journal of lightwave technology*, *21*(2), 384-405.

[87] Perrin, STERLING. “OBS: The Pipes Are Calling” (2009). Recuperado el 1 de noviembre de 2015, de http://www.lightreading.com/document.asp?doc_id=180392

[88] Pickavet, M., Van Caenegem, R., Demeyer, S., Audenaert, P., Colle, D., Demeester, P., ... & Gladisch, A. (2007). Energy footprint of ICT. In *Broadband Europe 2007*.

[89] Pleros, N., Apostolopoulos, D., Petrantonakis, D., Stamatiadis, C., & Avramopoulos, H. (2008, September). All-optical static RAM cell with read/write functionality at 5 Gbps. In *Optical Communication, 2008. ECOC 2008. 34th European Conference on* (pp. 1-2). IEEE.

[90] Ponnurangam, Rajesh. “CISC 856 – TCP Options Selective Acknowl*edge*ment (SACK) RFC 2018 Duplicate Selective Acknowledgment (DSACK) RFC 2883”. Recuperado el 1 de noviembre de 2015, de https://www.google.com.ec/url?sa=t&rct=j&q=&esrc=s&source=web&cd=2&ved=0ahUKEwjAmsWt7q_SAhWBQiYKHb2oCZ4QFggiMAE&url=https%3A%2F%2Fwww.eecis.udel.edu%2F~amer%2F856%2Fsack.07f.ppt&usg=AFQjCNHSB65PGN-LSm6-PrCrs1heTEDoWw

[91] Puerto, G., Ortega, B., Martínez, A., Manzanedo, M. D., Pastor, D., & Capmany, J. NODO DE CONMUTACIÓN DE PAQUETES PARA INTERNET ÓPTICA.

[92] Raffaelli, C., & Zaffoni, P. (2006, June). Simple Analytical Formulation of the TCP Send Rate in Optical Burst-Switched Networks. In *Computers and Communications, 2006. ISCC'06. Proceedings. 11th IEEE Symposium on* (pp. 109-114). IEEE.

[93] Ramakrishna Shenai, Sunil Gowda and Krishna M Sivalingam Optical Burst-Switched Wavelength Division Multiplexed *Network Simulator* (OBS-ns Design, Implementation and

User Manual, School of EECS, Washington State University, Pullman, WA 99164 Dr. Candan Cankaya, Alcatel CRC, Richardson, TX Research Time Period: December 2000 – November 2001

[94] Ramaswami, R., Sivarajan, K., & Sasaki, G. (2009). Optical networks: a practical perspective. Morgan Kaufmann.

[95] Rodrigues, J. J., Freire, M. M., Garcia, N. M., & Monteiro, P. M. (2007, July). Enhanced just-in-time: a new resource reservation protocol for optical burst switching networks. In *Computers and Communications, 2007. ISCC 2007. 12th IEEE Symposium on* (pp. 121-126). IEEE.

[96] Rodrigues, J. J., Gregório, J. M., & Vasilakos, A. V. (2010). Enhanced just-in-time plus protocol for optical burst switching networks. *Optical Engineering*, *49*(7), 075001-075001.

[97] Sahara, A., Tsukishima, Y., Shimano, K., Koga, M., Mori, K., Sakai, Y., ... & Kawai, M. (2004). Demonstration of connection-oriented optical burst switching network utilising PLC and MEMS switches. *Electronics Letters*, *40*(25), 1597-1599.

[98] Sahara, A., Kasahara, R., Yamazaki, E., Aisawa, S., & Koga, M. (2005, March). The demonstration of congestion-controlled optical burst switching network utilizing two-way signaling-field trial in JGN II testbed. In *Optical Fiber Communication Conference, 2005. Technical Digest. OFC/NFOEC* (Vol. 5, pp. 3-pp). IEEE.

[99] Shi, K. (2012). *Investigation of wavelength tunable lasers for use in coherent optical communication systems* (Doctoral dissertation, Dublin City University).

[100] Shieh, W., & Djordjevic, I. (2010). Orthogonal frequency division multiplexing for optical communication. *ELSEVIER Inc.*, 38-39.

[101] Shihada, B., & Ho, P. H. (2008). Transport control protocol in optical burst switched networks: issues, solutions, and challenges. *IEEE Communications Surveys & Tutorials*, *10*(2).

[102] Simeonidou, D., Nejabati, R., Arnaud, B. S., Beck, M., Clarke, P., Hoang, D. B., ... & Mambretti, J. (2004, August). Optical network infrastructure for grid. In *Grid Forum Draft, GFD-I* (Vol. 36).

[103] Sullivan, J., Charbonneau, N., & Vokkarane, V. M. (2010, December). Performance evaluation of TCP over optical burst switched (OBS) networks using coordinated burst cloning and forward-segment redundancy. In *Global Telecommunications Conference (GLOBECOM 2010), 2010 IEEE* (pp. 1-6). IEEE.

[104] Sun, Y., Hashiguchi, T., Minh, V. Q., Wang, X., Morikawa, H., & Aoyama, T. "Design and implementation of an optical burst-switched network testbed". *IEEE Communications Magazine*, (2005) *43*(11), S48-S55.

[105] Tanemura, T., Al Amin, A., & Nakano, Y. (2009, September). Development and field demonstration of an optical burst switching testbed with PLZT optical matrix switch. In *Photonics in Switching, 2009. PS'09. International Conference on* (pp. 1-4). IEEE.

[106] Tena Martín, Miguel Ángel, Evaluación de Arquitecturas de Red Híbridas OBS/OCS, Master thesis, Universitat Politècnica de Catalunya, 2009. Tena Martín, M. Á. (2009).

Evaluación de Arquitecturas de Red Híbridas OBS/OCS, Arquitecturas de Red Híbridas OBS/OCS (Master thesis, Universitat Politècnica de Catalunya).

[107] Towsley, D., Liu, Y., Qiao, C., & Yu, X. (2003). Performance Evaluation of TCP Implementations in OBS Networks.

[108] Trowbridge Steve. "Understanding Optical Transport Network (G.709)" (2010). Recuperado el 1 de noviemre de 2015 de, www.cvt-dallas.org/March2010.pdf

[109] Tucker, R. S., & De Zhong, W. (1999). Photonic packet switching: An overview. *IEICE Transactions on Electronics*, *82*(2), 202-212.

[110] Turner, J. S. (1999). Terabit burst switching. *Journal of High Speed Networks*, *8*(1), 3-16.

[111] Venkatesh, T., Murthy, C. S. R., & Murthy, C. S. R. (2010). *An analytical approach to optical burst switched networks* (Vol. 2010). New York e-: Springer.

[112] Verma, S., Chaskar, H., & Ravikanth, R. (2000). Optical burst switching: a viable solution for terabit IP *backbone*. *IEEE network*, *14*(6), 48-53.

[113] Walker, Timothy. (2007). "Optical transport network (OTN) tutorial". Recuperado el 1 de noviembre de 2015, de https://www.itu.int/ITU-T/studygroups/com15/otn/OTNtutorial.pdf

[114] Wang, X., Morikawa, H., & Aoyama, T. (2000, September). Burst optical deflection routing protocol for wavelength routing WDM networks. In *Opticom 2000* (pp. 257-266). International Society for Optics and Photonics.

[115] Wehrle, K., Günes, M., & Gross, J. (Eds.). (2010). *Modeling and tools for network simulation*. Springer Science & Business Media.

[116] Wei, J. Y., & McFarland, R. I. (2000). Just-in-time signaling for WDM optical burst switching networks. *Journal of lightwave technology*, *18*(12), 2019-2037.

[117] Welzl, M. (2005). Network congestion control: managing internet traffic. John Wiley & Sons.

[118] Willner, A. E. Optical Burst Switching (OBS): Issues in the Physical Laye.University of Southern California Los Angeles, CA

[119] Xu, J., Qiao, C., Li, J., & Xu, G. (2003, March). Efficient channel scheduling algorithms in optical burst switched networks. In *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies* (Vol. 3, pp. 2268-2278). IEEE.

[120] Xu, L., Harfoush, K., & Rhee, I. (2004, March). Binary increase congestion control (BIC) for fast long-distance networks. In *INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies* (Vol. 4, pp. 2514-2524). IEEE.

[121] Yamaguchi, J., Sakata, T., Shimoyama, N., Ishii, H., Shimokawa, F., & Yamamoto, T. (2007). High-yield fabrication methods for MEMS tilt mirror array for optical switches. *NTT Technical Review*, *5*(10), 1-6.

[122] Yao, S., Mukherjee, B., Yoo, S. B., & Dixit, S. (2003). A unified study of contention-resolution schemes in optical packet-switched networks. *Journal of lightwave technology*, *21*(3), 672-683.

[123] Yariv, A., & Yeh, P. (2007). *Photonics: optical electronics in modern communications* (Vol. 6). New York: oxford university press.

[124] Yoo, M., & Qiao, C. (1997, August). Just-enough-time (JET): A high speed protocol for bursty traffic in optical networks. In *Vertical-Cavity Lasers, Technologies for a Global Information Infrastructure, WDM Components Technology, Advanced Semiconductor Lasers and Applications, Gallium Nitride Materials, Processing, and Devi* (pp. 26-27). IEEE.

[125] Yu, X., Chen, Y., & Qiao, C. (2002, July). A Study of traffic statistics of assembled burst traffic in optical burst-switched networks. In *ITCom 2002: The Convergence of Information Technologies and Communications* (pp. 149-159). International Society for Optics and Photonics.

[126] Yu, X., Chen, Y., & Qiao, C. (2002, November). Performance evaluation of optical burst switching with assembled burst traffic input. In *Global Telecommunications Conference, 2002. GLOBECOM'02. IEEE* (Vol. 3, pp. 2318-2322). IEEE.

[127] Yu, X., Qiao, C., & Liu, Y. (2004, March). TCP implementations and false time out detection in OBS networks. In *INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies* (Vol. 2, pp. 774-784). IEEE.

[128] Yu, X., Li, J., Cao, X., Chen, Y., & Qiao, C. (2004, December). Traffic statistics and performance evaluation in optical burst switched networks. *Journal of lightwave technology*, *22*(12), 2722-2738.

[129] Yu, X., & Qiao, C. (2006, October). TCP performance over OBS networks with multiple flows input. In *Broadband Communications, Networks and Systems, 2006. BROADNETS 2006. 3rd International Conference on* (pp. 1-10). IEEE.

[130] Zervas, G., Nejabati, R., Wang, Z., Simeonidou, D., Yu, S., & O'Mahony, M. (2007, March). A fully functional application-aware optical burst switched network test-bed. In *Optical Fiber Communication Conference* (p. OWC2). Optical Society of America.

[131] Zervas, G. S., De Leenheer, M., Sadeghioon, L., Klonidis, D., Qin, Y., Nejabati, R., ... & Demeester, P. (2009). Multi-granular optical cross-connect: Design, analysis, and demonstration. *Journal of Optical Communications and Networking*, *1*(1), 69-84.

[132] Zhang, Q., Vokkarane, V. M., Wang, Y., & Jue, J. P. (2005, October). Evaluation of burst retransmission in optical burst-switched networks. In *Broadband Networks, 2005. BroadNets 2005. 2nd International Conference on* (pp. 276-282). IEEE.

[133] Zhang, W., Wu, J., Lin, J., Ji, Y., Xu, K., Guo, H., ... & Liu, X. (2006, March). TCP performance experiment on OBS network testbed. In *Optical Fiber Communication Conference* (p. OThF1). Optical Society of America.

[134] Zhang, W., Wu, J., Lin, J., Minxue, W., & Jindan, S. (2007). TCP Performance Experiment on LOBS Network Testbed. *Optical Network Design and Modeling*, 186-193.

[135] Zyskind, J., & Srivastava, A. (Eds.). (2011). *Optically amplified WDM networks*. Academic press.

[136] "Cooperation agreement with KISTI". Recuperado el 23 de febrero de 2017, de http://www.ist-phosphorus.eu/event_10.php

[137] "France Telecom-Orange y Alcatel-Lucent ponen en servicio un enlace óptico de 400 Gbps por longitud de onda". Recuperado el 23 de febrero de 2017, de http://www.europapress.es/portaltic/sector/noticia-france-telecom-orange-alcatel-lucent-ponen-servicio-enlace-optico-400-gbps-longitud-onda-20130206165944.html

[138] "Intune Networks". Recuperado el 23 de febrero de 2017, de http://archive.is/dhiHx

[139] "Memory RAM Speed - Access Time, Megahertz (MHz), Bytes Per Second". Recuperado el 23 de febrero de 2017, de http://www.computermemoryupgrade.net/measuring-ram-speed.html

[140] "Verisma iVX 8000 CR 2.1.1 PRODUCT DATA SHEE". Recuperado el 1 de noviembre de 2015, de http://archive.is/WlSKL

[141] "EtherBurst® Traffic Aware Transport" (2007). Recuperado el 1 de noviembre de 2015, de http://0299d3f.netsolhost.com/NewPages/EtherBurst.pdf

[142] "Company Overview of Matisse Networks Inc." (2009). Recuperado el 1 de noviembre de 2015, de http://investing.businessweek.com/research/stocks/private/snapshot.asp?privcapId=7719949

[143] "Modeling TCP Throughput Padhye" (2010). Recuperado el 1 denoviembre de 2015, de http://www.mathcs.emory.edu/~cheung/Courses/558/Syllabus/07-TCP-Anal/

[144] "Huawei Demonstrates OBTN Prototype at OFC/NFOEC 2011" (2011). Recuperado el 23 de febrero de 2017, de http://pr.huawei.com/en/news/hw-062737.htm#.WLPdToWcHIU

[145] "Transform the Performance and Economics of Optical Networks" (2012). Recuperado el 23 de febrero de 2017, de https://insight.nokia.com/transform-performance-and-economics-optical-networks

ANEXOS

Anexo 1. Programa principal de la simulación (obs.sh)

```
#!/bin/bash

#Inicializacion de variables
errmsg="Entrada incorrecta, se toma el valor por defecto"
Src="Reno"
blp="0.001 0.003 0.01 0.03 0.1"
Ba="1000000 100000000 200000000"
Tb="0.0001 0.0003 0.001 0.003 0.01 0.03 0.1 0.3"
Tbu="0.003"
markov="N"
simtime=1000

declare -A flow=( [delay]=0.01 [pktsize]=512 [wmax]=128 [dack]="N" [Ba]=$Ba
)
declare -A flowd=( [delay]="Retardo[s] d <0.01>: " [pktsize]="TamaÃ±o del
paquete[bytes] pktsize <512>: " [wmax]="TamaÃ±o mÃ¡ximo de ventana
TCP[pkts] Wm <128>: " [dack]="ACK retardado[S/N] DAck <N>: " [Ba]="Ancho de
banda de acceso Ba[b/s] <$Ba>: " )
declare -A burst=( [Tp]=0.03 [ncc]=2 [ndc]=8 [bwpc]=1000000000
[maxsize]=100000 [model]=simplificado [Tb]="0 $Tbu" )
declare -A burstd=( [Tp]="Tiempo de propagaciÃ³n[s] Tp <0.03>: "
[ncc]="Numero de canales de control ncc <2>: " [ndc]="Numero de canales de
datos ndc <8>: " [bwpc]="Ancho de banda por canal[bps] bwpc <10000000000>:
" [maxsize]="TamaÃ±o mÃ¡ximo de rÃ¡faga[bytes] Bmaxsize <100000>: "
[model]="Modelo a simular[simplificado/basico] model <simplificado>: "
[Tb]="Tiempo de ensamblado[s] Tb <$Tbu>: " )

arrL=($blp); sizeL=${#arrL[@]}
arrT=($Tb); sizeT=${#arrT[@]}
pm=${arrL[$sizeL-1]}
Tbm=${arrT[$sizeT-1]}
block=0
idx=$((sizeL/2))
pu=${arrL[$idx]}

#Declaracion de funciones
function Settings_BurstFlow()
{
   local -a 'arraykeys=("${!'"$1"'[@]}")'

   for i in ${arraykeys[*]}
   do
      prompt=$2"[$i]"
      read -p "${!prompt}" value
      if [ $i == "dack" ]
      then
         [[ $value = "S" || $value = "N" ]] && eval $1"[$i]"=$value || echo
$errmsg
      elif [ $i == "model" ]
```

```
      then
         [[ $value = "simplificado" || $value = "basico" ]] && eval
$1"[$i]"=$value || echo $errmsg
      elif [ $i == "Ba" ]
      then
         n=$1"[$i]"; n=(${!n})
         k=0
         for j in $value
         do
            if [[ $j =~ ^[0-9]+([.][0-9]+)?$ ]]; then k=$((k+1)); fi
       done
         [[ $k = ${#n[@]} ]] && eval $1'[$i]=$value' || echo $errmsg
      elif [ $i == "Tb" ]
         then
            [[ $value =~ ^[0-9]+([.][0-9]+)?$ ]] && { Tbu=$value;
value="$Tbu"; eval $1'[$i]=$value'; } || echo $errmsg
            temp=$(eval echo \${$1"[$i]"})
           temp=$temp" $Tb"
            temp=$(echo $temp | tr ' ' '\n' | sort -u | tr '\n' ' ')
            eval $1'[$i]=$temp'
      else
            [[ $value =~ ^[0-9]+([.][0-9]+)?$ ]] && eval $1'[$i]=$value' ||
echo $errmsg
      fi
      data=$1"[$i]"
      echo "$data= ${!data}"
    done
}

function Ingress_Data()
{
   #Ingreso informacion del usuario
   echo -e '\nINGRESAR INFORMACION RED DE ACCESO'
   echo '-----------------------------------'
   Settings_BurstFlow flow flowd

   echo -e '\nINGRESAR INFORMACION RED OBS'
   echo '---------------------------'
   Settings_BurstFlow burst burstd

   read -p "Tiempo de la simulaciÃ³n simtime <1000>: " value
   [[ $value =~ ^[0-9]+([.][0-9]+)?$ ]] && simtime=$value || echo $errmsg
   echo "simtime=$simtime"
}

function Delete_Data()
{
   #EliminaciÃ³n archivos residuales
   rm -f *.thput
   rm -f *.tr
   rm -f *.nam
   rm -f *.out
   rm -f *.wnd
   rm -f *.brst
   rm -f *.dat
   rm -f *.eps
```

```
}

function Simulation ()
{
 eval local bw=$2"[Ba]"; bw="${!bw}"
 eval local delay=$2"[delay]"; delay="${!delay}"
 eval local pktsize=$2"[pktsize]"; pktsize="${!pktsize}"
 eval local wmax=$2"[wmax]"; wmax="${!wmax}"
 eval local dack=$2"[dack]"; dack="${!dack}"
 eval local Tp=$3"[Tp]"; Tp="${!Tp}"
 eval local ncc=$3"[ncc]"; ncc="${!ncc}"
 eval local ndc=$3"[ndc]"; ndc="${!ndc}"
 eval local bwpc=$3"[bwpc]"; bwpc="${!bwpc}"
 eval local Tbs=$3"[Tb]"; Tbs="${!Tbs}"
 eval local maxsize=$3"[maxsize]"; maxsize="${!maxsize}"
 eval local model=$3"[model]"; model="${!model}"

 fid=0

 for B in $Ba
 do
    i=0
    for T in $Tbs
    do
       for L in $blp
       do
          echo "$1    $L    $B"
          ns obs.tcl $1 $B $delay $pktsize $wmax $dack $fid $L $Tp $ncc
$ndc $bwpc $T $maxsize $model $4 $simtime
          if [[ ( $T = $Tbu ) && ( $L = $arrL || $L = $pu || $L =
${arrL[$sizeL-1]} ) ]]
          then
             block=$i
             if [[ -e "winfile.wnd" ]]; then cat "winfile.wnd" >> "winfile-
$1_$fid.out";fi
             [[ ! -e "files-$1.wnd" ]] && echo "winfile-$1_$fid.out" >
"files-$1.wnd" || echo "winfile-$1_$fid.out" >> "files-$1.wnd"
           if [[ -e "trace.tr" && ( $T = $Tbu && $L = $pu ) ]]
           then
                cp "trace.tr" "trace-$1_$fid.tr"
                rm "trace.tr"
             fi
          fi
       done
       echo "-----------------"
       i=$((i+1))
     done
      fid=$((fid+1))
 done
}

#Funcion para calcular la ganancia DFL
function CF_Calculation ()
{

   declare -A NB; declare -A B
```

```
   if [[ -e "files-$1.thput" ]]
   then
      i=0
      filenames="$(sort "files-$1.thput" | uniq)"
      for f in $filenames; do
         grep -P "^${1}\t0\t" $f > "cf0-${1}_${i}.tmp"
         grep -P "^${1}\t${Tbu}\t" $f > "cf-${1}_${i}.thput"
         [[ ! -e "cf-$1.thput" ]] && echo "cf-${1}_${i}.thput" > "cf-
$1.thput" || echo "cf-${1}_${i}.thput" >> "cf-$1.thput"
         while IFS=$'\t' read -r typ T p psm Bths Bthf Bs class; do
            NB["$i,$p"]="$Bths"
          B["$i,$p"]="$Bs"
         done < "cf0-${1}_${i}.tmp"
         i=$((i+1))
      done
      filenames="$(sort "cf-$1.thput" | uniq)"
      i=0
      for f in $filenames; do
         while IFS=$'\t' read -r typ T p psm Bths Bthf Bs class; do
          CFs=$(echo "$Bths/${NB[$i,$p]}" | bc -l)
            CFf=$(echo "$Bthf/${NB[$i,$p]}" | bc -l)
            CF=$(echo "$Bs/${B[$i,$p]}" | bc -l)
          printf "%s\t%s\t%s\t%s\t%f\t%f\t%f\t%s\n" "$typ" "$T" "$p" "$psm"
"$CFs" "$CFf" "$CF" "$class";
         done < $f > "$f.tmp" && mv "$f.tmp" $f
         i=$((i+1))
      done
      rm *.tmp
   fi
}

#Funcion para calcular del histograma de la distribucion de la rafagas
function Hist_Calculation ()
{

   if [[ ! -e "files-$1.brst" && -e "files-$1.thput" ]]
   then
      filenames="$(sort "files-$1.thput" | uniq)"
      i=0
      for f in $filenames; do
         [[ ! -e "files-$1.tr" ]] && echo "trace-${1}_${i}.tr" > "files-
$1.tr" || echo "trace-${1}_${i}.tr" >> "files-$1.tr"
         i=$((i+1))
      done
      filenames="$(sort "files-$1.tr" | uniq)"
      i=0
      for f in $filenames; do
         gawk -v Edge=0 -v Core=1 -f measure-burst.awk $f > "burst-
${1}_${i}.brst"
         [[ ! -e "files-$1.brst" ]] && echo "burst-${1}_${i}.brst" >
"files-$1.brst" || echo "burst-${1}_${i}.brst" >> "files-$1.brst"
         i=$((i+1))
      done
   fi
}
```

```
function MainMenu ()
{

   clear
   echo '**************************'
   echo '**TCP SOBRE REDES OBS v.1**'
   echo '**************************'
   echo '1. MODELO DE FLUJOS CLASE LENTA/RAPIDA'
   echo '2. MODELO DE PROCESO REGENERATIVO DE MARKOV'
   echo -e '3. SALIR\n'

   echo -n 'Elegir una opcion: '
}

function SubMenu()
{
   clear
   echo '**************************'
   echo '**TCP SOBRE REDES OBS v.1**'
   echo '**************************'
   echo '1. INGRESAR/MODIFICAR DATOS'
   echo '2. INICIAR SIMULACION'
   echo '3. OBTENER THPUT/BFCR vs. BLP'
   echo '4. OBTENER THPUT vs. BLP&TB'
   echo '5. OBTENER HISTOGRAMA (TAMAÃ‘O DE RAFAGA)'
   echo '6. OBTENER CWND vs. TIME'
   echo -e '7. SALIR\n'

   echo -n 'Elegir una opcion: '
}

#Inicio del script
Delete_Data

op1=0
while [ $op1 -lt 3 ]
do
   MainMenu
   read op1
   case $op1 in
      1|2)
         op2=0
         while [ $op2 -lt 7 ]
         do
            SubMenu
            read op2
            case $op2 in
               1)
                  Ingress_Data
                  ;;
               2)
              start=$(date +%s)
                  [[ $op1 -eq 1 ]] && { Src="Reno"; markov="N"; } || {
Src="Reno Newreno Sack"; markov="S"; }
                  for S in $Src; do
                     Simulation $S flow burst $markov
```

```
                done
                end=$(date +%s)
            elapse=$((end-start))
            echo "***********************"
            echo "**Simulacion completa**"
            echo "***********************"
            echo "Tiempo de ejecucion: `date -d@"$elapse" -u +%H:%M:%S`"
                ;;
             3)
                for S in $Src; do
                   CF_Calculation $S
                   gnuplot -e
"TCPtype='$S';block=$block;sizeL=$sizeL;Tbu=$Tbu;pm=$pm;pu=$pu;idx=$idx;mar
kov='$markov'" thput-blp.gpi
                   evince thput-$S-blp.eps
            done
            if [ $op1 -eq 2 ]
            then
               gnuplot -e
"TCPtype='$Src';block=$block;sizeL=$sizeL;Tbm=$Tbm;pu=$pu;idx=$idx" thput-
compare.gpi
               evince thput-compare.eps
            fi
                ;;
             4)
                RTT=$(echo "4.0*${flow[delay]}+2.0*${burst[Tp]}+2.0*$Tbu"
| bc -l)
            for S in $Src; do
                   gnuplot -e
"TCPtype='$S';Wm=${flow[wmax]};RTT=$RTT;sizeL=$sizeL;sizeT=$sizeT;pm=$pm;Tb
m=$Tbm;markov='$markov'" thput-blp3d.gpi
               evince thput-$S-blp3d.eps
            done
                ;;
             5)
               for S in $Src; do
                   Hist_Calculation $S
                   gnuplot -e
"TCPtype='$S';Wm=${flow[wmax]};block=$block;sizeL=$sizeL;pu=$pu;Tbu=$Tbu;ma
rkov='$markov'" histogram-burst.gpi
               evince histogram-$S-burst.eps
            done
                ;;
             6)
            for S in $Src; do
                   gnuplot -e
"TCPtype='$S';Wm=${flow[wmax]};pu=$pu;markov='$markov';simtime=$simtime"
window.gpi
                   evince window-$S.eps
            done
                ;;
             7)
                break
                ;;
          *) ;;
        esac
```

```
            if [ $op2 -lt 7 ]; then read -p "`echo -e '\nPresionar una
tecla para continuar...'`" key; fi
          done
         ;;
      3)
         exit 0
         ;;
      *) ;;
   esac
   if [ $op1 -lt 3 ]; then read -p "`echo -e '\nPresionar una tecla para
continuar...'`" key; fi
done
```

Anexo 2. Script de simulación para *ns-2* (obs.tcl)

```
#Simulacion TCP sobre OBS
#2 escenarios a) simplificado; b) básico
#Canales de 1 Gbit/s
#8 DCs; 2 CCs

# Incluye la librerias del módulo OBS
source ../../lib/ns-obs-lib-hierarchical.tcl
source ../../lib/ns-obs-defaults.tcl
source ../../lib/ns-optic-link.tcl

# Añade sólo las cabeceras requeridas y no todas (por defecto)
remove-all-packet-headers
add-packet-header Common IP Message TCP Flags

#=====================================================================#
# Definiciones de constantes
StatCollector set debug_ 1
BurstManager set debug_ 1
Classifier/BaseClassifier/EdgeClassifier set type_ 0
Classifier/BaseClassifier/CoreClassifier set type_ 1

#v 6 utiliza de 1 - 70 us para el tiempo de offset
BurstManager offsettime 0.00007

# Configura el tiempo de procesamiento del bhp a 1 us
Classifier/BaseClassifier/CoreClassifier set bhpProcTime 0.000001
Classifier/BaseClassifier/EdgeClassifier set bhpProcTime 0.000001
Classifier/BaseClassifier set option 0

# Establece el tamaño de las colas de los enlaces
Queue set limit_ 500000

# Inicialización de variables
set ns [new Simulator]
set sc [new StatCollector]
set flow_list "type bw delay pktsize wmax dack id"
set burst_list "blp Tp ncc ndc bwpc Tb maxsize model markov"
```

```
set sim_start 1
set sim_end 1000

$ns color 1 Red
$ns color 2 Blue
$ns color 3 Green


#=====================================================================#
# Definiciones de procedimientos

proc min { a b } {
   set x [expr $a<$b ? $a: $b]

   return $x
}

proc log2 { num } {
   set x [expr log($num)/log(2)]

   return $x
}

#=============================================================#
# Procedimiento para calcular el Throughput de fuentes lentas #
#=============================================================#
proc Bs { flow burst rtt } {

   upvar 1 $flow flowr
   upvar 1 $burst burstr

   set b [expr {$flowr(dack) eq "N"} ? 1 : 2]
   set Wm $flowr(wmax)
   set p [expr $burstr(blp) ? $burstr(blp): 1.e-12]

   set fp [expr
1+$p+2.0*pow($p,2)+4.0*pow($p,3)+8.0*pow($p,4)+16.0*pow($p,5)+32.0*pow($p,6
)]
   set EW [expr (2.0+$b)/(3*$b)+sqrt(8.0*(1-
$p)/(3*$b*$p)+pow((2.0+$b)/(3*$b),2))]
   set EY [expr $EW<$Wm ? [expr (1.0-$p)/$p+$EW]: [expr (1.0-$p)/$p+$Wm]]
   set Q [expr $EW<$Wm ? [min 1 3.0/$EW]: [min 1 3.0/$Wm]]
   set ER [expr 1.0/(1-$p)]
   set EX [expr $EW<$Wm ? [expr $b*$EW/2.0]: [expr $b/8.0*$Wm+(1.0-
$p)/($p*$Wm)+1]]
   set EA [expr ($EX+1.0)*$rtt]
   set EZ [expr $rtt*$fp/(1.0-$p)]

   set B [expr ($EY+$Q*$ER)/($EA+$Q*$EZ)]
   set Bmax [expr 1.0*$Wm/$rtt]
   puts "EW= $EW"

   return [expr $B>$Bmax ? $Bmax: $B]
}

#=============================================================#
```

```
# Procedimiento para calcular el Throughput de fuentes rapidas #
#=============================================================#
proc Bf { flow burst rtt } {

   upvar 1 $flow flowr
   upvar 1 $burst burstr

   set b [expr {$flowr(dack) eq "N"} ? 1 : 2]
   set Wm $flowr(wmax)
   set p [expr $burstr(blp) ? $burstr(blp): 1.e-12]

   set fp [expr
1+$p+2.0*pow($p,2)+4.0*pow($p,3)+8.0*pow($p,4)+16.0*pow($p,5)+32.0*pow($p,6
)]
   set EY [expr $p>1.0/$Wm ? [expr 1.0/pow($p,2)]: [expr $Wm/$p]]
   set EH [expr $p/(1.0-$p)]
   set EA [expr (1.0-$p)*$rtt/$p]
   set EZ [expr $rtt*$fp/(1.0-$p)]

   set B [expr ($EY+$EH)/($EA+$EZ)]
   set Bmax [expr 1.0*$Wm/$rtt]

   return [expr $B>$Bmax ? $Bmax: $B]
}

#=========================================#
# Procedimiento para calcular el RTO de TCP #
#=========================================#
proc GetRTO { filename } {

   set ERTO 0
   set n [exec wc -l < $filename]
   set fi [open $filename r]

   while {[gets $fi line] >=0} {
      lappend fr([lindex $line 6]) [lindex $line 6]
   }

   foreach rto [array names fr] {
      set ERTO [expr {$ERTO+$rto*[llength $fr($rto)]/$n}]
   }

   close $fi
   puts "ERTO= $ERTO"
   return $ERTO
}

#=========================================================#
# Procedimiento para calcular el Throughput de TCP Reno #
#=========================================================#
proc BReno { flow burst rtt } {
   global tcp0
   upvar 1 $flow flowr
   upvar 1 $burst burstr

   set b [expr {$flowr(dack) eq "N"} ? 1 : 2]
```

```
   set Wm [$tcp0 set window_]
   set flowr(pktsize) [expr [$tcp0 set packetSize_] + [$tcp0 set
tcpip_base_hdr_size_]]
   set p [expr $burstr(blp) ? $burstr(blp): 1.e-12]
   set STb [expr (($flowr(bw)*$burstr(Tb)+1.0)/8)/$flowr(pktsize)]
   set burstr(size) [expr $STb>$Wm ? $Wm: ceil($STb)]
   set S $burstr(size)
   set RTO [GetRTO "rto.dat"]

   set EWx [expr
1.0/2+1.0/(2*$b)+sqrt(pow(1.0/2+1.0/(2*$b),2)+2.0*$S/($b*$p))]
   set EY [expr [log2 $EWx]>$S ? [expr (1-$p)/$p*$S+(5.0/2-1/(pow(2,$S-
1)))*$EWx]: [expr (5.0/2-1/(pow(2,[log2 $EWx])))*$EWx-[log2 $EWx]+$S/$p]]
   set EX [expr [log2 $EWx]>$S ? [expr $b*$EWx-$b*$EWx/(pow(2,$S))]: [expr
$b*$EWx]]
   set ETDP [expr [log2 $EWx]>$S ? ($EX+$S)*$rtt: ($EX+[log2 $EWx])*$rtt ]

if [catch {set QEWx [expr pow($p,$EWx/$S-1)]}] {
   puts "$::errorCode"
   if {[lrange $::errorCode 0 1] eq "ARITH UNDERFLOW"} { set QEWx 0 }
   if {[lrange $::errorCode 0 1] eq "ARITH DIVZERO"} { set QEWx 1 }
}
   set fp [expr
1+$p+2.0*pow($p,2)+4.0*pow($p,3)+8.0*pow($p,4)+16.0*pow($p,5)+32.0*pow($p,6
)]
   set EH [expr $p/(1.0-$p)]
   set ETOP [expr $RTO*$fp/(1.0-$p)]

   set B [expr ($EY+$QEWx*$EH)/($ETDP+$QEWx*$ETOP)]
   set Bmax [expr 1.0*$Wm/$rtt]
   puts "p=$p S= $S  log(EWx)= [expr log($EWx)]"

   return [expr $B>$Bmax ? $Bmax: $B]
}

#=================================================================#
# Procedimiento para calcular el Throughput de TCP NewReno/Sack #
#=================================================================#
proc BNewrenoSack { flow burst rtt } {

   global tcp0 RTT0
   upvar 1 $flow flowr
   upvar 1 $burst burstr

   set b [expr {$flowr(dack) eq "N"} ? 1 : 2]
   set Wm [$tcp0 set window_]
   set flowr(pktsize) [expr [$tcp0 set packetSize_] + [$tcp0 set
tcpip_base_hdr_size_]]
   set p [expr $burstr(blp) ? $burstr(blp): 1.e-12]
   set STb [expr (($flowr(bw)*$burstr(Tb)+1.0)/8)/$flowr(pktsize)]
   set burstr(size) [expr $STb>$Wm ? $Wm: ceil($STb)]
   set S $burstr(size)
   set RTO [GetRTO "rto.dat"]

   set EWx [expr
($b+2.0)/(3.0*$b)+sqrt(pow((($b+2.0)/(3.0*$b)),2)+8.0*$S/(3.0*$b*$p))]
```

```
   set EX [expr $EWx<$Wm ? [expr $b/2.0*$EWx]: [expr
1.0*$S/(1.0*$p*$Wm)+$Wm/8.0+1.0/2]]
   set EY [expr $EWx<$Wm ? [expr 3.0/2*$EWx+(1.0-$p)/$p*$S]: [expr
3.0*$Wm/2+(1.0-$p)*$S/$p]]
   set ETDP [expr $EWx<$Wm ? ([expr {$flowr(type) eq "Sack"}] ? [expr
($EX+1)*$rtt]: [expr ($EX+$S)*$rtt]): [expr
$rtt*($S/($p*$Wm)+$Wm/8.0+3.0/2)]]
   if {[catch {set QEWx [expr pow($p,[expr $EWx<$Wm ? [expr $EWx/$S-1]:
[expr $Wm/$S-1]])]} errmsg]} {
      puts "Error: $errmsg"
      if {[lrange $::errorCode 0 1] eq "ARITH UNDERFLOW"} {set QEWx 0}
      if {[lrange $::errorCode 0 1] eq "ARITH DIVZERO"} {set QEWx 1}
   }

   set fp [expr
1+$p+2.0*pow($p,2)+4.0*pow($p,3)+8.0*pow($p,4)+16.0*pow($p,5)+32.0*pow($p,6
)]
   set EH [expr $p/(1.0-$p)]
   set ETOP [expr $RTO*$fp/(1.0-$p)]

   set lmda [expr $flowr(bw)/(8.0*$flowr(pktsize))]

   set Tbo [expr $EWx<$Wm ? [expr $RTT0/2.0]: [expr
sqrt($p*$Wm/(2.0*$lmda)*$RTT0*($Wm/8.0+3.0/2))]]
   set Tbp [expr sqrt($p*$Wm/(2.0*$lmda)*($RTT0+$Wm/8.0+3.0/2))]
   set Bm [expr $EWx<$Wm ? [expr
1.0/(4*$RTT0)*sqrt(3.0*$lmda*$RTT0/($b*$p))]: [expr
1.0/($p*($Wm+12.0)/(4*$lmda)+1.0*$RTT0/$Wm+sqrt(8.0*$p*$RTT0*($Wm/8.0+3.0/2
)/(1.0*$lmda*$Wm)))]]
   set Bmp [expr
1.0/($p*($Wm+12.0)/(4*$lmda)+1.0*$RTT0*$p/$Wm+sqrt(8.0*$p*($RTT0+$Wm/8.0+3.
0/2)/(1.0*$lmda*$Wm)))]

   puts "p=$p S= $S  EWx= $EWx Wm=$Wm"
   puts "Tbo=$Tbo Bm=$Bm Tbp=$Tbp Bmp=$Bmp"

   set B [expr ($EY+$QEWx*$EH)/($ETDP+$QEWx*$ETOP)]
   set Bmax [expr 1.0*$Wm/$rtt]
   return [expr $B>$Bmax ? $Bmax: $B]
}

#=============================================================================#
# Procedimiento para calcular los resultados una vez que finaliza la
transmisión de datos #
#=============================================================================#
proc finish { flow burst tcp } {
   global ns nf sc tf wf sim_start sim_end RTT0 RTT class rtof
   upvar 1 $flow flowr
   upvar 1 $burst burstr

   set seq [$tcp set t_seqno_]
   set MSS [expr $flowr(pktsize)+[$tcp set tcpip_base_hdr_size_]]
   set thput_s [format "%.3f" [expr 1.0*$seq/($sim_end-
$sim_start)*$MSS*8.0/1000]]

   if [info exists rtof] {close $rtof}
```

```
   if [expr {$burstr(markov) eq "N"}] {
      set thput_ts [format "%.3f" [expr [Bs flowr burstr
$RTT]*$MSS*8.0/1000]]
      set thput_tf [format "%.3f" [expr [Bf flowr burstr
$RTT]*$MSS*8.0/1000]]
   } else {
        if [expr {$flowr(type) eq "Reno"}] {
           set thput_ts [format "%.3f" [expr [BReno flowr burstr
$RTT]*$MSS*8.0/1000]]
        } else {
             set thput_ts [format "%.3f" [expr [BNewrenoSack flowr burstr
$RTT]*$MSS*8.0/1000]]
        }
        set thput_tf 0
   }

   set BSND [$sc get-value "BURSTFROMTOSND"]
   set BRCV [$sc get-value "BURSTFROMTORCV"]
   set ps [expr $BSND? [expr 1.0*($BSND-$BRCV)/$BSND]: 0]
   set psim [string range $ps 0 [expr [string first "." $ps 0]+4]]
   set rate [expr $flowr(bw)/1000000.0]
   set pos [string first "." $rate]
   set BMb [string range $rate 0 [expr [string match "*.0" $rate] ? [expr
$pos-1]: [expr $pos+4]]]M

   set thputfile "thput-$flowr(type)\_[lindex $class 0].thput"
   set out [open "$thputfile" a+]
   set nfiles [open "files-$flowr(type).thput" a+]

   puts $out
"$flowr(type)\t$burstr(Tb)\t$burstr(blp)\t$psim\t$thput_ts\t$thput_tf\t$thp
ut_s\t[lindex $class 1]\t$BMb"
   puts $nfiles "$thputfile"

   $ns flush-trace
   $ns flush-nodetrace

   #Cierra los archivos de traza
   close $nf
   close $tf
   close $wf
   close $out
   close $nfiles

   set code {
        {
                if($6 == NAME && $7 >= 0) {
                        print $1, $7
                }
        }
   }

   #Ejecuta nam sobre el archivo de traza correspondiente
   exec nam obs.nam &

   $sc display-sim-list
```

```
   puts "Clase [lindex $class 1]"
   puts "RTT0=$RTT0\[s\] RTT=$RTT\[s\] Bths= $thput_ts\[kbps\]  Bthf=
$thput_tf\[kbps\]  Bsim= $thput_s\[kbps\]"
   puts "Simulacion Ok para Tb= $burstr(Tb) pth=$burstr(blp) psim=$psim"

   exit 0
}

#=============================================================#
# Procedimiento registrar la información de la ventana de TCP #
#=============================================================#
proc plotWindow { p Tb bw TCPSrc reg } {
   global ns class

   set t 0.1
   set Wm [$TCPSrc set window_]
   set now [$ns now]
   set cwnd [$TCPSrc set cwnd_]

   puts $reg "$now\t$p\t$Tb\t[expr $cwnd<$Wm ? $cwnd: $Wm]\t[$TCPSrc set
t_seqno_]\t[lindex $class 1]\t[expr $bw/1000000]M"
   $ns at [expr $now+$t] "plotWindow $p $Tb $bw $TCPSrc $reg"
}

#=============================================================================#
# Procedimiento para crear un flujo con un generador de tráfico sobre un
agente TCP #
#=============================================================================#
Simulator instproc create_tfgen_connection {flow tfgen tcp tcpSrc sink
sinkDst src dest start0 stop0} {

 upvar 1 $flow flowr
 upvar 1 $tfgen tfgenr
 upvar 1 $tcp tcpr
 upvar 1 $sink sinkr
 upvar 1 $src srcr
 upvar 1 $dest destr

 set tcpr [new Agent/$tcpSrc]
 $tcpr set fid_ 3
 $self attach-agent $srcr $tcpr

 set sinkr [new Agent/$sinkDst]
 $self attach-agent $destr $sinkr
 $self connect $tcpr $sinkr

 set tfgenr [new Application/Traffic/MyExponential]
 $tfgenr set packetSize_ $flowr(pktsize)
 $tfgenr set rate_ $flowr(bw)
 $tfgenr attach-agent $tcpr

 #Inicia y detiene la transmisión de datos TCP
 $self at $start0 "$tfgenr start"
 $self at $stop0 "$tfgenr stop"

 puts "flujo de trafico TCP entre $src = $srcr y $dest = $destr creado"
```

```
}

#=========================================================#
# Procedimiento para crear la topología de cada escenario #
#=========================================================#
Simulator instproc create_topology { flow burst } {

   global n E C core_count edge_count node_count RTT0 RTT
   upvar 1 $flow flowr
   upvar 1 $burst burstr

   set obs_count [expr $edge_count+$core_count]
   set maxch [expr $burstr(ncc)+$burstr(ndc)]

   #Establece el tipo de direccionamiento jerárquico
   $self node-config -addressType hierarchical
   #Establece el número de dominios
   AddrParams set domain_num_ $obs_count ;

   for {set i 0} {$i < $obs_count} {incr i} {
      lappend cluster_num 1 ;
      if {$i < $edge_count} {
         lappend eilastlevel [expr
($edge_count+$node_count)/($edge_count)];
      } else {
           lappend eilastlevel 1;
      }
   }

   #Establece el número de clusters
   AddrParams set cluster_num_ $cluster_num ;
   #Establece el número de nodos por cada dominio
   AddrParams set nodes_num_ $eilastlevel ;

   # Crea los nodos de edge
   set i 0
    while {$i < $edge_count} {
      puts "Creando nodo edge $i"
      set E($i) [$self create-edge-node $i $edge_count]
      $E($i) color blue
      $E($i) shape square
      $E($i) label "edge$i"
      $E($i) label-at up
      puts "E($i) node id: [$E($i) id]"
      incr i
   }

   # Crea los nodos de core
   set i 0
   while {$i < $core_count} {
      puts "Creando nodo core $i"
      set C($i) [$self create-core-node [expr $i+$edge_count] $core_count]
      $C($i) color red
      $C($i) shape hexagon
      $C($i) label "core$i"
      $C($i) label-at up
```

```
      puts "C($i) node id: [$C($i) id]"
      incr i
    }

   # Crea los nodos cliente
   set i 0
   while {$i < $node_count} {
      set j $i
      set k 1
      puts "Creando nodo cliente$i"
      set n($i) [$self node [$E($j) id].0.$k]
      $n($i) label "nodo$i"
      $n($i) label-at up
      puts "n($i) node id: [$n($i) id] adress: [$E($j) id].0.$k"
      incr i
   }

   $self createSimplexFiberLink $E(0) $E(1) $burstr(bwpc) $burstr(Tp)
$burstr(ncc) $burstr(ndc) $maxch
   $self createSimplexFiberLink $E(1) $E(0) $burstr(bwpc) $burstr(Tp)
$burstr(ncc) $burstr(ndc) $maxch
   $self simplex-link-op $E(0) $E(1) orient right
   for {set i 0} {$i < $edge_count} {incr i} {
      for {set j 0} {$j < [expr $node_count/2]} {incr j} {
         set k [expr !($i%2) ? $j: [expr $j+$node_count/2]]
         $self duplex-link $E($i) $n($k) $flowr(bw) $flowr(delay) DropTail
         if [expr !($i%2)] {
          $self duplex-link-op $E($i) $n($k) orient [expr
270+$j*360/$edge_count]deg
         } else {$self duplex-link-op $E($i) $n($k) orient [expr 270-
$j*360/$edge_count]deg }
      }
   }
   Agent/IPKT set from_ [$E(0) id]
   Agent/IPKT set to_ [$E(1) id]

   #Crea un modelo de error del tipo Bernoulli
   set em [new ErrorModel]
   $em set rate_ $burstr(blp)
   $em unit pkt
   $em ranvar [new RandomVariable/Uniform]
   $em drop-target [new Agent/Null]
   $self lossmodel $em $E(0) $E(1)
   $self duplex-link-op $n(0) $E(0) queuePos 0.5

   $self build-routing-table
   puts "Se han creado los NEs y los enlaces correspondientes"
}

#Lee argumentos de la linea de comandos
if {$argc > 1} {

    for {set i 0} {$i < [llength $flow_list]} {incr i} {
       set flow([lindex $flow_list $i]) [lindex $argv $i]
    }
```

```
    for {set i 0} {$i < [llength $burst_list]} {incr i} {
       set burst([lindex $burst_list $i]) [lindex $argv [expr [llength
$flow_list]+$i]]
    }

    set sim_end [lindex $argv [expr $argc-1]]
    set maxch [expr $burst(ncc)+$burst(ndc)]
    set nf [open "obs.nam" w]
    set tf [open "| grep -P \".* 0 tcp.*\|.* 0 selfsim.*\|.* 1 IPKT.*\" >
trace.tr" w]
    set wf [open "winfile.wnd" w]

    #Inicializa variables de la red en función del escenario elegido
    set core_count 0
    set edge_count 2
    set node_count 2
    Agent/TCP set window_ $flow(wmax)
    Agent/TCP set ssthresh_ [expr $flow(wmax)/2]
    Agent/TCP set packetSize_ $flow(pktsize)
    Agent/TCPSink/DelAck set interval_ 200ms
    set RTT0 [format "%.3f" [expr 4.0*$flow(delay)+2.0*$burst(Tp)]]
    if [expr {$burst(model) eq "simplificado"}] {
       set RTT [format "%.3f" [expr $RTT0+$burst(Tb)]]
       BurstManager burstless 1 "ack"
    } else {
       set RTT [format "%.3f" [expr $RTT0+2.0*$burst(Tb)]]
       BurstManager burstless 0
    }
    BurstManager bursttimeout $burst(Tb)
    BurstManager maxburstsize $burst(maxsize)
    puts "BurstManager bursttimeout= [BurstManager bursttimeout]"

    set src [expr (1.0*$flow(bw)*$burst(Tb))/(8.0*$flow(pktsize))]

    if {$burst(Tb)} {
       if {$src <=1} {
          set class [list "$flow(id)" "lenta"]
       } elseif {$src >= $flow(wmax)} {
            set class [list "$flow(id)" "rapida"]
       } else {
          set class [list "$flow(id)" "media"]
       }
     } else {
          set class [list "$flow(id)" "flujo$flow(id)"]
     }
     $ns trace-all $tf
     $ns namtrace-all $nf
} else {
    puts "usage: ns obs1.tcl <protocol> <burst loss probability> <access
rate> <delay> <pktsize> <TCP Wmax> <DelayACK> <Tp> <CCh> <DCh> <OBS rate>
<bursttimeout> <maxburst>"
    puts "<protocol> is Reno, Newreno, Sack"
    puts "<burst loss probability> is the link error rate (0.001 = 0.1%).
Use 0 for no errors"
    puts "<access rate> is bandwidth of access network"
    puts "<delay> is delay of access network in seconds"
```

```
    puts "<pktsize> is size of packet in bytes"
    puts "<TCP Wmax> is maximum TCP window in pkts"
    puts "<DelayACK> select Delay ACK option"
    puts "<Tp> is propagation time within OBS Network"
    puts "<CCh> is the number of control channel"
    puts "<DCh> is the number of data channel"
    puts "<obs rate> is bandwidth of OBS network"
    puts "<bursttimeout> is timeout of burst generation in seconds"
    puts "<maxburst> is maximum size of burst in bytes"
    puts "<model> is an escenario of simulation"

    exit 1
}

 $ns create_topology flow burst

 #Verifica la variante de TCP para establecer el agente respectivo
 switch $flow(type) {
        "Reno" {
            set Src TCP/Reno
                  set Dst TCPSink
      }
        "Newreno" {
            set Src TCP/Newreno
            set Dst TCPSink
      }
        "Sack" {
            set Src TCP/Sack1
            set Dst TCPSink/Sack1
      }
        default {
            set Src TCP
            set Dst TCPSink
      }
 }
 #Fija el uso de ACK retardado si fue elegido
 if [expr {$flow(dack) eq "S"}] { append Dst "/DelAck" }
 #Crea una conexion TCP con el generador de tráfico definido en
MyExponential
 $ns create_tfgen_connection flow tfgen tcp0 $Src sink0 $Dst n(0) n([expr
$node_count/2]) $sim_start $sim_end

 set rtof [open "rto.dat" w]
 $tcp0 trace rto_
 $tcp0 attach $rtof


 #Inicia el regitro de la evolución de la ventana de transmisión y detiene
la simulación
 $ns at $sim_start "plotWindow $burst(blp) $burst(Tb) $flow(bw) $tcp0 $wf"
 $ns at [expr $sim_end+2] "finish flow burst $tcp0"

 #Inicia la simulación
 $ns run
```

Anexo 3. Script para graficar los resultados en 2D (thput-blp.gpi)

```
set terminal postscript eps enhanced color dashed 'Times-Bold'
set output sprintf("thput-%s-blp.eps",TCPtype)
set key right top
set grid
set logscale x
set xrange [:pm]
set xlabel 'Prob. perdida de rafagas' font ", 15" offset 2
set tics font ", 15"

#Obtener la lista de archivos a graficar
file1=system(sprintf("sort files-%s.thput | uniq | awk '{print
$1}'",TCPtype))
file2=system(sprintf("sort cf-%s.thput | uniq | awk '{print $1}'",TCPtype))
n1=words(file1)
n2=words(file2)
print file1
print file2

#Establecer estilos de línea
if (markov eq 'N') {
   set style line 1 lt 1 lc 1
   set style line 2 lt 1 lc 2
   set style line 3 lt 2 lc 1 pt 8
   set style line 4 lt 2 lc 3 pt 10
   set style line 5 lt 2 lc 2 pt 17
} else {
   set style line 1 lt 1 lc 1
   set style line 2 lt 1 lc 3
   set style line 3 lt 1 lc 2
   set style line 4 lt 2 lc 1 pt 8
   set style line 5 lt 2 lc 3 pt 10
   set style line 6 lt 2 lc 2 pt 17
}
set style increment user

data(fname,s,b)=sprintf("<awk 'NR>%i*%i && NR<=%i*(%i+1){print}'
%s",s,b,s,b,fname)
hdr(fname,s,b,c)=system(sprintf("awk 'NR==1+(%i*%i) {print $%i}'
%s",s,b,c,fname))

set multiplot layout 2,2

set ylabel sprintf("{Throughput [kbps] _{(Tb >
0)}}\n{Tb=%.0e}",hdr(word(file1,1),sizeL+0,block+0,2)+0) offset 1
if (markov eq 'N') {
max_thput=system(sprintf("awk 'BEGIN {max=-1} {if ($6>max) max=$6} END
{print max}' %s",word(file1,n1)))
set yrange [0:max_thput*2]
plot data(word(file1,1),sizeL+0,block+0) u 3:5 t 'Clase lenta (teorico)' w
lp, data(word(file1,n1),sizeL+0,block+0) u 3:6 t 'Clase rapida (teorico)' w
lp, for [i=1:n1] data(word(file1,i),sizeL+0,block+0) u 3:7 t sprintf("Clase
%s_{%s}
```

```
(simulado)",hdr(word(file1,i),sizeL+0,block+0,8),hdr(word(file1,i),sizeL+0,
block+0,9)) w lp
} else {
max_thput=system(sprintf("awk 'BEGIN {max=-1} {if ($5>max) max=$5} END
{print max}' %s",word(file1,n1)))
set yrange [0:max_thput*2.5]
plot for [i=1:n1] data(word(file1,i),sizeL+0,block+0) u 3:5 t sprintf("TCP
%s_{%s} (teorico)",TCPtype,hdr(word(file1,i),sizeL+0,block+0,9)) w lp, for
[i=1:n1] data(word(file1,i),sizeL+0,block+0) u 3:7 t sprintf("TCP %s_{%s}
(simulado)",TCPtype,hdr(word(file1,i),sizeL+0,block+0,9)) w lp
}

set ylabel "Throughput [kbps] _{(Tb = 0)}" offset 1
if (markov eq 'N') {
max_thput=system(sprintf("awk 'BEGIN {max=-1} {if ($6>max) max=$6} END
{print max}' %s",word(file1,n1)))
set yrange [0:max_thput*0.7]
plot data(word(file1,1),sizeL+0,0) u 3:5 t 'Clase lenta (teorico)' w lp,
data(word(file1,1),sizeL+0,0) u 3:6 t 'Clase rapida (teorico)' w lp,  for
[i=1:n1] data(word(file1,i),sizeL+0,0) u 3:7 t sprintf("Clase %s_{%s}
(simulado)",hdr(word(file1,i),sizeL+0,0,8),hdr(word(file1,i),sizeL+0,0,9))
w lp
} else {
max_thput=system(sprintf("awk 'BEGIN {max=-1} {if ($5>max) max=$5} END
{print max}' %s",word(file1,n1)))
set yrange [0:max_thput*0.7]
plot for [i=1:n1] data(word(file1,i),sizeL+0,0) u 3:5 t sprintf("TCP
%s_{%s} (teorico)",TCPtype,hdr(word(file1,i),sizeL+0,0,9)) w lp, for
[i=1:n1] data(word(file1,i),sizeL+0,0) u 3:7 t sprintf("TCP %s_{%s}
(simulado)",TCPtype,hdr(word(file1,i),sizeL+0,0,9)) w lp
}

set ylabel sprintf("{Ganancia DFL _{(Tb >
0)}}\n{Tb=%.0e}",hdr(word(file2,1),sizeL+0,0,2)+0) offset 1
if (markov eq 'N') {
max_crb=system(sprintf("awk 'BEGIN {max=-1} {if ($6>max) max=$6} END {print
max}' %s",word(file2,n2)))
set yrange [0:max_crb*2.5]
plot word(file2,1) u 3:5 t 'Clase lenta (teorico)' w lp, word(file2,n2) u
3:6 t 'Clase rapida (teorico)' w lp, for[i=1:n2] word(file2,i) u 3:7 t
sprintf("Clase %s_{%s}
(simulado)",hdr(word(file2,i),sizeL+0,0,8),hdr(word(file2,i),sizeL+0,0,9))
w lp
} else {
max_crb=system(sprintf("awk 'BEGIN {max=-1} {if ($7>max) max=$7} END {print
max}' %s",word(file2,n2)))
set yrange [0:max_crb*2.5]
plot for[i=1:n2] word(file2,i) u 3:5 t sprintf("TCP %s_{%s}
(teorico)",TCPtype,hdr(word(file2,i),sizeL+0,0,9)) w lp, for[i=1:n2]
word(file2,i) u 3:7 t sprintf("TCP %s_{%s}
(simulado)",TCPtype,hdr(word(file2,i),sizeL+0,0,9)) w lp
}

set ylabel sprintf("{Throughput [kbps] _{(Tb > 0)}}\n{p=%.0e}",pu) offset 1
set xlabel 'Tiempo de ensamblado [s]' font ", 15" offset 2
if (markov eq 'N') {
```

```
max_thput=system(sprintf("awk 'BEGIN {max=-1} {if ($6>max) max=$6} END
{print max}' %s",word(file1,n1)))
set yrange [0:max_thput*1.5]
plot word(file1,1) every sizeL+0::idx+0 u 2:5 t 'Clase lenta (teorico)' w
lp, word(file1,n1) every sizeL+0::idx+0 u 2:6 t 'Clase rapida (teorico)' w
lp, for[i=1:n1] word(file1,i) every sizeL+0::idx+0 u 2:7 t sprintf("Clase
%s_{%s}
(simulado)",hdr(word(file1,i),sizeL+0,block+0,8),hdr(word(file1,i),sizeL+0,
block+0,9)) w lp
} else {
max_thput=system(sprintf("awk 'BEGIN {max=-1} {if ($5>max) max=$5} END
{print max}' %s",word(file1,n1)))
set yrange [0:max_thput*2]
plot for[i=1:n1] word(file1,i) every sizeL+0::idx+0 u 2:5 t sprintf("TCP
%s_{%s} (teorico)",TCPtype,hdr(word(file1,i),sizeL+0,block+0,9)) w lp,
for[i=1:n1] word(file1,i) every sizeL+0::idx+0 u 2:7 t sprintf("TCP %s_{%s}
(simulado)",TCPtype,hdr(word(file1,i),sizeL+0,block+0,9)) w lp
}
unset multiplot
```

Anexo 4. Script para graficar los resultados en 3D (thput-blp3d.gpi)

```
set terminal postscript eps enhanced color dashed 'Times-Bold'
set output sprintf("thput-%s-blp3d.eps",TCPtype)
set key right top
set grid
set xlabel "{Tb [s]}" font ", 13" offset -2
set ylabel "{BLP}" font ", 13" offset 2
set zlabel "{Throughput}\n{[kbps]}" font ", 13" rotate by 90 offset -2.1
set tics font ", 10"
set ztics offset 1

#Obtener la lista de archivos a graficar
file=system(sprintf("sort files-%s.thput | uniq | awk '{print
$1}'",TCPtype))
n=words(file)
hdr(fname,s,b,c)=system(sprintf("awk 'NR==1+(%i*%i) {print $%i}'
%s",s,b,c,fname))

set multiplot layout 2,3
set logscale x
set logscale y
set palette rgb 33,13,10
set dgrid3d sizeL+0,sizeT+0 qnorm 4
set pm3d
set pm3d hidden3d 100
set view 55,210

if (markov eq 'N') {
   print file
   splot word(file,1) u ($2!=0?$2:1/0):($3!=0?$3:1/0):5 w l pal t "TCP
Clase_{lenta} (teorico)"
   splot word(file,n) u ($2!=0?$2:1/0):($3!=0?$3:1/0):6 w l pal t "TCP
Clase_{rapida} (teorico)"
```

```
   do for [i=1:n] {
      splot word(file,i) u ($2!=0?$2:1/0):($3!=0?$3:1/0):7 w l pal t
sprintf("Fuente TCP_{%s} (simulado)",hdr(word(file,i),sizeL+0,0,9))
   }
} else {
     do for [i=1:n] {
        splot word(file,i) u ($2!=0?$2:1/0):($3!=0?$3:1/0):5 w l pal t
sprintf("TCP %s_{%s} (teorico)",TCPtype,hdr(word(file,i),sizeL+0,0,9))
     }
     do for [i=1:n] {
        splot word(file,i) u ($2!=0?$2:1/0):($3!=0?$3:1/0):7 w l pal t
sprintf("TCP %s_{%s} (simulado)",TCPtype,hdr(word(file,i),sizeL+0,0,9))
    }
}
unset multiplot
```

Anexo 5. Script para graficar la comparativa de las tres variantes de TCP (thput-compare.gpi)

```
set terminal postscript eps enhanced color dashed 'Times-Bold'
set output 'thput-compare.eps'
set key right top
set grid
set logscale x
set xrange [:Tbm]
set xlabel 'Tiempo de ensamblado [s]' font ", 15" offset 2
set ylabel sprintf("{Throughput [kbps] _{(Tb > 0)}}\n{p=%.0e}",pu) offset 1
set tics font ", 15"

#Obtener la lista de archivos a graficar
file(fname)=system(sprintf("sort files-%s.thput | uniq | awk '{print
$1}'",fname))

n=words(file(word(TCPtype,1)))
hdr(fname,s,b,c)=system(sprintf("awk 'NR==1+(%i*%i) {print $%i}'
%s",s,b,c,fname))

#Establecer estilos de línea
set style line 1 lt 1 lc 1 pt 8
set style line 2 lt 1 lc 3 pt 10
set style line 3 lt 1 lc 2 pt 17
set style increment user

set multiplot layout 2,2

do for [i=1:n] {
   max=system(sprintf("awk 'BEGIN {max=-1} {if ($7>max) max=$7} END {print
max}' %s",word(file(word(TCPtype,n)),i)))
   set yrange [0:max*1.2]
   plot for[j=1:n] word(file(word(TCPtype,j)),i) every sizeL+0::idx+0 u 2:7
t sprintf("TCP %s_{%s}
(simulado)",word(TCPtype,j),hdr(word(file(word(TCPtype,j)),i),sizeL+0,0,9))
w lp
```

```
}
```

Anexo 6. Script para calcular el número de paquetes TCP porráfaga (measure-burst.awk)

```
#Programa utilizado para calcular el número de paquetes contenidos en una
ráfaga
BEGIN {
}
{
   action = $1;
   time = $2;
   from = $3;
   to = $4;
   type = $5;
   size = $6;
   flow_id = $8;
   packet_id = $12;
   if (to==Edge && action == "r" && type!="IPKT")
   {
      if (type=="tcp")
         npkt["tcp",size]++;
      else
         npkt["other",size]++;
   }
   if (to==Core && action== "+" && type=="IPKT" && size!="64")
   {
       for (ij in npkt)
       {
         split(ij,s,SUBSEP);
         Bnpkt[packet_id,s[1]]+=npkt[ij];
         Bsize[packet_id]+=npkt[ij]*s[2];
       }
       Btime[packet_id]=time;
       delete npkt;
   }
}
END {
   for (ij in Bnpkt)
   {
       split(ij,s,SUBSEP);
       B[s[1]]+=Bnpkt[ij];
   }
   for (i in B)
      printf
("%f\t%d\t%d\t%d\t%d\t%d\n",Btime[i],i,Bsize[i],B[i],Bnpkt[i,"tcp"],Bnpkt[i
,"other"]);
}
```

Anexo 7. Script para graficar el histograma de la distribución de las ráfagas con (histogram-burst.gpi)

```
set terminal postscript eps enhanced color solid 'Times-Bold'
set output sprintf("histogram-%s-burst.eps",TCPtype)
set key right top
set grid
set yrange [1e-3:1.7]
set xlabel "Tamano de Rafaga [pkts]"
set ylabel "{Numero de Rafagas}\n{Normalizado}" offset 3
set style fill solid 1.0 noborder

#Obtener la lista de archivos a graficar
file1=system(sprintf("sort files-%s.brst | uniq | awk '{print
$1}'",TCPtype))
file2=system(sprintf("sort files-%s.thput | uniq | awk '{print
$1}'",TCPtype))
n=words(file1)
print file1

bin(x,width)=width>0 ? sprintf("%.0f",width*floor(x/width+0.5))+0: 0
hdr(fname,s,b,c)=system(sprintf("awk 'NR==1+(%i*%i) {print $%i}'
%s",s,b,c,fname))
ndata=min=max=""
range=interval=binwidth=""

do for [col=0:1] {
   do for [i=1:n] {
      j=i+col*n
      ndata=sprintf("%s %s",ndata,system(sprintf("wc -l < burst-
%s_%i.brst",TCPtype,i-1)))
      min=sprintf("%s %s",min,system(sprintf("awk 'BEGIN {min=1e6} {if
($%i<min) min=$%i} END {print min}' burst-
%s_%i.brst",col+5,col+5,TCPtype,i-1)))
      max=sprintf("%s %s",max,system(sprintf("awk 'BEGIN {max=-1} {if
($%i>max) max=$%i} END {print max}' burst-
%s_%i.brst",col+5,col+5,TCPtype,i-1)))
      range=sprintf("%s %i",range,word(max,j)+0-word(min,j)+0)
      interval=sprintf("%s
%i",interval,ceil(1+3.33*log10(word(ndata,j)+0)))
      binwidth=sprintf("%s %f",binwidth,word(range,j)+0>0 ?
(word(range,j)+0)/(word(interval,j)+0): word(min,j)*0.1)
   }
}

#Establecer estilos de línea
set style line 1 lt 1 lc 1
set style line 2 lt 1 lc 3
set style line 3 lt 1 lc 2

set multiplot layout 2,2
do for [i=1:n] {
   set boxwidth word(binwidth,i)+0>word(binwidth,i+n)+0 ?
(word(binwidth,i)+0)/2: (word(binwidth,i+n)+0)/2
```

```
   set xrange [0:word(max,i)+0>word(max,i+n)+0 ? (word(max,i)+0)*1.05:
(word(max,i+n)+0)*1.05]
   set label 1 at graph 0.5,0.75 center
   set label 1 sprintf("p=%.0e Tb=%.0e",pu+0,Tbu+0)
   titulo= markov eq 'N' ? sprintf("TCP Clase
%s_{%s}",hdr(word(file2,i),sizeL+0,block+0,8),hdr(word(file2,i),sizeL+0,blo
ck+0,9)) : sprintf("Trafico TCP
%s_{%s}",TCPtype,hdr(word(file2,i),sizeL+0,block+0,9))

  set table "hist.dat"
  plot word(file1,i) u (bin($5,word(binwidth,i)+0)):(word(binwidth,i)+0>0 ?
1: 0) smooth freq, word(file1,i) u
(bin($6,word(binwidth,i+n)+0)):(word(binwidth,i+n)+0>0 ? 1: 0) smooth freq
  unset table

  plot "< grep -v u hist.dat" index 0 u ($1):($2>0 ? $2/(word(ndata,i)+0):
1e-12) t titulo smooth freq w boxes ls i,"" index 0 u 1:($2>0 ?
($2*1.05)/(word(ndata,i)+0): 1e-12):(sprintf("%i",$2)) w labels font ", 6"
notitle#, "< grep -v u hist.dat" index 1 u 1:($2>0 ?
$2/(word(ndata,i+n)+0): 1e-12) t 'Trafico de background' smooth freq w
boxes lt -1 fs pattern 2, "" index 1 u 1:($2>0 ?
($2*1.3)/(word(ndata,i+n)+0): 1e-12):(sprintf("%i",$2)) w labels font ", 6"
notitle

  unset label 1
}
unset multiplot
```

Anexo 8. Script para graficar la venta de transmisión de TCP (window.gpi)

```
set terminal postscript eps enhanced color solid 'Times-Bold'
set output sprintf("window-%s.eps",TCPtype)
set key right top
set grid
set xrange [:simtime]
set yrange [0:Wm*2]
set xlabel "Tiempo [s]"

#Obtener la lista de archivos a graficar
file=system(sprintf("sort files-%s.wnd | uniq | awk '{print $1}'",TCPtype))
n=words(file)
data(fname,s,b)=sprintf("<awk 'NR>%i*%i && NR<=%i*(%i+1){print}'
%s",s,b,s,b,fname)
hdr(fname,s,b,c)=system(sprintf("awk 'NR==1+(%i*%i) {print $%i}'
%s",s,b,c,fname))
ndata(fname)=system(sprintf("wc -l < %s",fname))
sizeL=ndata(word(file,1))/3

set style line 1 lt 1 lc 2
set style line 2 lt 1 lc 3
set style line 3 lt 1 lc 1
set style increment user
```

```
set multiplot layout 2,2
set ylabel "{Ventana de transmision}\n{[pkts]}" offset 2

do for [i=1:n] {
   set label 1 at graph 0.5,0.65 center
   set label 1 markov eq 'N' ? sprintf("TCP Clase
%s_{%s}",hdr(word(file,i),sizeL+0,0,6),hdr(word(file,i),sizeL+0,0,7)):
sprintf("TCP %s_{%s}",TCPtype,hdr(word(file,i),sizeL+0,0,7))
   plot for[j=0:2] data(word(file,i),sizeL+0,j) u 1:4 t sprintf("p=%.0e
Tb=%.0e",hdr(word(file,i),sizeL+0,j,2)+0,hdr(word(file,i),sizeL+0,j,3)+0) w
l
   unset label 1
}
unset multiplot
```