PONTIFICIA UNIVERSIDAD CATÓLICA DEL ECUADOR

FACULTAD DE INGENIERÍA

MAESTRÍA EN REDES DE COMUNICACIONES

PERFIL DEL TRABAJO PREVIO LA OBTENCION DEL TÍTULO DE:

MAGÍSTER EN REDES DE COMUNICACIONES

TEMA:

"DISEÑO DE UN NODO CON TECNOLOGÍA CLOUD-ORIENTED CONTENT DELIVERY NETWORK, UTILIZANDO LA INTEGRACIÓN CON MODELOS DE SERVICIO DE CLOUD COMPUTING, PARA EL DATACENTER DE UN PROVEEDOR DE SERVICIOS DE CONTENIDO"

SANTIAGO DAVID GÁLVEZ HUERTA

Quito, mayo 2016

DECLARACIÓN

Yo, Santiago David Gálvez Huerta, declaro bajo juramento que el trabajo aquí descrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y que he consultado las referencias bibliográficas que se incluyen en este documento.

A través de la presente declaración cedo mis derechos de propiedad intelectual correspondientes a este trabajo, a la Pontificia Universidad Católica del Ecuador – PUCE, según lo establecido por la Ley de Propiedad Intelectual, por su Reglamento y por la normatividad institucional vigente.

Santiago David Gálvez Huerta

C.I. 1716655806

DEDICATORIA

Este trabajo está dedicado especialmente para mi esposa e hijo, quienes han sido mi guía para lograr alcanzar todas las metas de estudio que me he propuesto, ellos son y siempre serán mi apoyo para lograr superar con esfuerzo todas esas horas de trabajo.

De igual manera agradezco profundamente a mis padres, hermanos y todas aquellas personas que me han apoyado durante toda mi carrera y desarrollo profesional.

ÍNDICE DE FIGURAS:

ÍNDICE DE TABLAS

INTRODUCCIÓN

Este trabajo presenta, el diseño de un nodo hibrido con tecnología de Red de Distribución de Contenido - *Contect Delivery Network* (CDN) orientado a operar en un ambiente de red en la nube - *Networked Cloud Computing Environment* (NCE), mediante la integración con modelos de distribución y servicio de Computación en la Nube "*Cloud Computing*" de pago o código abierto "*Open Source*" y lograr así una arquitectura *Cloud-Oriented Content Delivery Network* (CCDN), para optimizar el rendimiento e incrementar el desempeño de la distribución del contenido dentro de la red de un proveedor de Contenidos - *Content Delivery Service Provider* (CDSP) (Papagianni et al, 2013). El sistema tendrá la capacidad de mejorar la entrega del tráfico de contenido, y de igual manera considerar todos los aspectos de seguridad necesarios para una adecuada operación. Al integrar una arquitectura de red que entregue servicios CDN, a través de métodos de entrega de contenido orientados a servir en la nube "*Cloud*" y que emplee los diferentes modeles de distribución (*Public – Private – Hybrid e Inter-Cloud*) y de servicio SPI (*Software – Platform e Infraestructure*), se puede hacer el servicio más eficiente en costos, más competitivo y a gran escala (Halpert, 2011).

La primera parte de este proyecto de tesis de maestría en comunicaciones presentará, un estudio y análisis de las principales tecnologías de CDN y *Cloud Computing*, para lo cual se revisarán todos los conceptos y características relacionadas con las tecnologías expuestas y las mejores técnicas de integración dentro del entorno de la red hibrida CCDN, así como los retos del diseño, fortalezas y vulnerabilidades para lograr las metas con la mayor certeza y seguridad. También a manera de ejemplo de diseño se analizará la arquitectura del nodo dentro de la red del proveedor de contenidos (CDSP), tomando en cuenta aspectos claves como: servicios a prestar, aplicaciones que harán posible el manejo del contenido, protocolos, acceso a los recursos del sistema para operarlo remotamente, modelos de infraestructura existentes y así determinar los principal requisitos de operación del Nodo.

Finalmente ya con los objetivos claros y expuestos, se realizará un desarrollo estructurado del diseño en el plan, teniendo en cuenta siempre el índice tentativo y el cronograma del desarrollo de la investigación. Es importante recalcar que toda la información adquirida para

el desarrollo de la investigación está sólidamente sustentada con bibliografía de documentos científicos.

JUSTIFICACIÓN

En la actualidad las personas y las organizaciones, se han visto en la necesidad de buscar alternativas para almacenar, transferir, analizar o adquirir grandes cantidades de información "*Big Data*", debido a la creación de nuevas tecnologías de almacenamiento y equipos con mayor capacidad de procesamiento, así como la evolución de los sistemas para entregar un mayor ancho de banda o la imprescindible necesidad de adaptarse al uso de las redes virtuales y sociales, como un medio colaborativo a través del denominado *Social Content Delivery Network* (S-CDN) (Kugler et al, 2013). Todas estas variables han creado la necesidad para que las empresas de telecomunicaciones ofrezcan servicios a través de la denominada tecnología de Red de Distribución de Contenido (CDN), ofreciendo un servicio completo para manejar y mejorar la distribución de la información de contenido multimedia del usuario y también haciéndolo de manera segura. A toda esta revolución tecnológica, surgió la idea de integrarlo con modelos de distribución y servicio de contenido en *Cloud Computing*, por lo que este trabajo busca utilizar este modelo hibrido CCDN para mitigar y ofrecer soluciones a los problemas que vayan apareciendo debido al incremento en la demanda del tráfico, el cual puede ser optimizado; desarrollando nuevos tipos de servicios, que sean escalables y flexibles, bajo demanda o por necesidad del usuario, diseñados de tal manera que permitan la continuidad permanente del servicio ante fallos (Papagianni et al, 2013).

En este proyecto se desea investigar todos los aspectos relacionados a la tecnología de *CDN* y *Cloud Computing* más actuales y que sean factibles de implementar para un proveedor de contenido - *Cloud Service Providers* (CPs) (Papagianni et al, 2013), otorgándole beneficios en costo, ahorrándole hasta un 30% en el gasto de ancho de banda y a demás que garanticen la mitigación de los riesgos de seguridad, con disponibilidad completa 24/7 de los servicios e información.

Como consecuencia de la optimización, gracias al correcto uso de los recursos, una infraestructura *CDN/Cloud* bien diseñada representaría un enorme ahorro de recursos, tanto de energía, como de refrigeración durante las horas donde la carga eléctrica de uso es baja, mejorando así todo el sistema y haciéndolo más beneficioso para el medio ambiente "*Green*", teniendo en cuenta que actualmente, los "*Datacenters"* consumen el 1,5% de toda la electricidad generada en los Estados Unidos como base, recalcando que esta medida no incluye la energía consumida por la Internet, pero de igual manera haciendo que los gastos de operación y alimentación eléctrica de los centros de datos puedan llegar a ser extremos (Jarabek & Wang, 2011).

La importancia de este proyecto radica en la implementación de la tecnología de CCDN como un modelo de negocios atractivo para los proveedores de contenido, ya que el *Cloud Computing* no se trata solo de un negocio de empresas individuales que compran el más grande y mejor hardware, por ejemplo en lo que Oracle llama una "*Cloud in a box*", sino que el negocio en la Nube trata acerca de empresas que dan el costo, la carga de la gestión y el mantenimiento del hardware, todos esto junto como servicio (Kepes, 2011). Esto permitirá definir niveles de acceso a los diferentes servicios contratados por los usuarios, sean estos internos o externos, priorizando la seguridad y denegando cualquier tipo de acceso no permitido. Para esto el proyecto tendrá la necesidad de realizar un análisis comparativo de las diferentes tecnologías empleadas dentro del Nodo y los distintos servicios de *Cloud Computing*, sean estos implementados a través de herramientas de pago o código abierto. Confirmando que mecanismos se ajusten mejor a las necesidades del Nodo CCDN y cuales le permitan:

- Determinar los diferentes métodos de integración entre los servicios de CDN y los diferentes tipos de servicios en *Cloud*, en particular dentro de una *Hybrid Cloud*.
- Comprender las principales funcionalidades y confiabilidad del modelo de servicio de Plataforma (*Platform as a Service* - PaaS), para lograr el mejor método de adopción del servicio de Video bajo Demanda (*Video on Demand* - VoD) por parte del usuario (Li et al, 2011).

ANTECEDENTES

La necesidad de información, la creación de nuevas tecnologías, la evolución de los sistemas, la latencia en la adquisición de la información durante el transporte y la masificación de contenido multimedia en las redes virtuales, han permitido que el almacenamiento de los datos de los usuarios se conviertan gradualmente en el punto de inflexión con el mayor porcentaje de consumo de tráfico dentro de la red, obligado a varios usuarios, grandes empresas y sitios web a utilizar los servicios de la denominada tecnología de Red de Distribución de Contenido (CDN) para mejorar y acelerar la calidad de navegación de sus servicios de datos y especialmente la gestión de información o aplicaciones con video embebido y bajo demanda (VoD), esto ya dentro de la segunda generación de CDN (Ling et al, 2013). Por otro lado las redes de Distribución de Contenido que utilizan o están orientadas a servir en la nube "*Cloud*" como medio de almacenaje, recientemente han empezado a emerger y constituyen una prometedora alternativa a las redes de distribución de contenidos CDN tradicionales (Papagianni et al, 2013). A su vez las Redes de Distribución de Contenido Orientadas a la Nube (CCDNs) aprovecharían las ventajas de ser más eficientes en costos y destacarían los principios de los conceptos de servir en la Nube, para ofrecer un adecuado servicio de alojamiento "*pseudo-Hosting*" a los diferentes proveedores de contenido Multimedia que no poseen la infraestructura necesaria de red y según el modelo de *Cloud Computing* para el negocio que requieran, sea este de Infraestructura como Servicio (IaaS) – Plataforma como Servicio (PaaS) o Software como Servicio (SaaS) y teniendo en cuenta también la dispersión de los recursos almacenados dependiendo de la ubicación geografía (Chen et al, 2012). Este modelo de CCDN propone una arquitectura de CDN abierta, con el propósito de fomentar la creación de ecosistemas de distribución de contenido competitivos, colaborativos, de fácil acceso y a gran escala (Zhang, 2014). Para ofrecer a los usuarios una plataforma de servicios web rápidos, alojados en la nube, de fácil manejo, alto rendimiento, flexibilidad y con bajo costo; esta arquitectura integrará un ambiente de infraestructura en la nube y un sistemas de red de distribución de contenido, con la capacidad y adaptabilidad necesarias para ser implementados en varios lugares de mundo, como cúmulos "Clúster" de almacenamiento de datos y así tener alojado el contenido multimedia del usuario final en el nodo más cercano (Lin et al, 2011). De igual manera los diferentes modelos de negocio y

beneficios técnicos del *Cloud Computing* son una muestra clara de la evolución de la tecnología del almacenamiento de los datos, ya que han traído un eficiente incremento de rendimiento a través del aumento en el cálculo computacional, mediante una estructura centralizada única de procesamiento, memoria, almacenamiento (tecnología-cache) y ancho de banda. Es por esto que la convergencia de ambas tecnologías podría marcar un legado en la industria de las comunicaciones y la distribución del contenido (Ramachnadran & Sivaprakasam, 2010). Teniendo en cuenta que una de las soluciones para manejar problemas de red dentro de los sistemas de *Cloud Computing* es la integración con la tecnología de CDN, ya que ha demostrado logros importantes y beneficios significativos en pruebas de alta descarga sobre datos multimedia de uso común (Yale et al, 2012).

Ahora que ya se encuentran centradas las bases para la integración de las tecnologías *CDN/Cloud*, es necesario conocer los retos claves de ambos servicios, para mitigar y ofrecer soluciones poderosas a los problemas de optimización de tráfico, seguridad y cuellos de botella, debido al creciente incremento en la demanda del tráfico en los recursos de red, promoviendo la colaboración entre infraestructuras y sus diferentes capas de operación, a través de un mutuo intercambio de información y logrando una efectiva optimización del rendimiento, desarrollando nuevos tipos de servicios que se adapten a las expectativas del usuario, mientras se va improvisando la calidad en la experiencia (QoE) (Selim et al, 2013). En lo referente al manejo del tráfico crítico como el de video, se abordarán varios retos, sobre los diferentes modelos en el diseño y los beneficios en la entrega de servicios de Video bajo Demanda (VoD) hacia la integración con el modelo hibrido de CDN orientado o asistido por la Nube (CCDN), logrando ahorrar así hasta un 30% en el gasto de ancho de banda, en comparación con el modelo clásico de Cliente/Servidor (Li et al, 2011), y teniendo en cuenta que el paradigma del *Cloud Computing* está siendo ampliamente empleado de manera eficiente dentro de los sistemas, debido a sus técnicas de gestión de recursos y escalabilidad de demanda (Okamoto et al, 2012). Se debe considerar que la distribución de los recursos almacenados dentro de los diferentes modelos de *Cloud Computing* de Infraestructura, Plataforma o Software, deben llegar al usuario final bajo demanda, con suscripciones de pago por uso, de la manera más transparente y efectiva, sin importar donde está alojado el servicio alrededor del mundo (Buyya, 2009). Todo esto nos hace entender que se ha creado la

necesidad de establecer una atmosfera computacional (NCE), con sistemas en la nube orientadas al mercado e interconectadas con múltiples dominios y proveedores de contenido, que han ido adoptando la tecnología del almacenamiento distribuido y la tecnología caché, facilitándoles la entrega de servicios CDN baratos y facultando a los consumidores de todo este tráfico de contenidos, a tener menores tiempos de respuesta y amplios anchos de banda en servicios que utilicen Protocolo de Aplicaciones Inalámbricas - *Wireless Application Protocol* (WAP) para servicios móviles (Wang et al, 2011).

La virtualización de las redes está siendo adoptada rápidamente por varios campos de la industria de las telecomunicaciones, para ser distribuida a través de la Internet y sabiendo que al estar siendo aplicada a la Arquitectura Orientada al Servicio - *Service-Oriented Architecture* (SOA), facilitará la convergencia de la Red y el *Cloud Computing* a través del modelo de Red como un Servicio - *Network-as-a-Service* (NaaS) (Duan et al, 2012). De igual manera se debe considerar que para que el Instituto Nacional de Estándares y Tecnología – *National Institute of Standards and Technology* (NIST) un servicio de carácter orientado a la Nube "*Cloud*" debe tener ciertas características esenciales, como la del auto-servicio bajo demanda, un mayor acceso a la red, disponibilidad de un conjunto de recursos, rápida flexibilidad y servicios a la medida (Kepes, 2011). Para identificar el modelo bajo demanda que se adapte fácilmente a la estructura hibrida de la (CCDN) expuesta, se debe tener siempre en cuenta las diferentes categorías de *Cloud Computing*, que a su vez tienen ciertas características que los diferencian, como el IaaS que es el hardware y software que lo gestiona todo, el PaaS que es el conjunto de herramientas y servicios que permiten levantar las aplicaciones y el SaaS que está diseñado para los usuarios finales, siendo su servicio entregado a través de la Internet (Rackspace, 2015).

OBJETIVOS DE LA INVESTIGACIÓN

Objetivo General:

Diseñar un Nodo con tecnología hibrida *Cloud-Oriented Content Delivery Network* (CCDN), mediante la integración con Modelos de Distribución y Servicio en un *Networked Cloud Computing Environment* (NCE), utilizando herramientas *Open Source* o Propietarias, para mejorar la Calidad de la Experiencia (QoE) y el Servicio (QoS) del tráfico de contenido multimedia y el *Video on Demand* (VoD) en el *Datacenter* de un *Contect Delivery Service Provider* (CDSP).

Objetivos Específicos:

1. Estudiar el estado emergente de las tecnologías de CDN y los entornos de *Cloud Computing*, identificando los beneficios de los modelos de distribución y servicio disponibles que mejor se ajusten para la consecución del proyecto y sus mejores prácticas de seguridad dentro de CCDN.

2. Analizar la arquitectura de tres tipos de Nodo CCDN, que utilizan la integración de las capas de servicio (SPI) de *Cloud Computing* con la tecnología CDN, e identificar el modelo más óptimo para la distribución de servicios multimedia.

3. Diseñar el Nodo híbrido CCDN e identificar el método de ubicación más eficiente de los *Surrogates* Virtuales, para mejorar el QoS en la prestación de servicios de *Video on Demand* (VoD).

CAPÍTULO 2: ESTADO DEL ARTE

Marco Teórico

Redes para la Distribución de Contenidos

(Content Delivery Network - CDN)

(Cloud Computing)

CONTENT DELIVERY NETWORK (CDN)

2.1 INTRODUCCIÓN

Durante los últimos años se ha producido una evolución de las tecnologías que tienen como objetivo mejorar la entrega de contenidos y la prestación de servicios a través de la Internet. Mercados enteros han sido creados ofreciendo equipamiento y dispositivos con conectividad a la red, herramientas de software y nuevos tipos de servicios de red. Es por esto que cuando se utilizan conjuntamente, estas tecnologías forman un nuevo tipo de red, que se refiere a menudo como red de contenido [RFC 3466] (Hofmann & Beaumont, 2005). Las Redes para la Distribución de Contenido (CDNs) han sido introducidas en la infraestructura de comunicación estándar a nivel mundial, para sobrellevar las limitaciones que han aparecido en la Internet, debido a la rápida aceptación de los accesos de banda ancha "*broadband*", el incremento en la complejidad de los sistemas y a la inminente necesidad que los usuarios tienen por percibir un alto grado de calidad de servicio (QoS) al acceder y obtener contenido multimedia o interactuar con el negocio de la transmisión de video, que han ido gradualmente convirtiéndose en la principal fuente de tráfico dentro de una red. Es por esto que para mejorar esta calidad se servicio, evitar congestión, cuellos de botella "*bottlenecks"*, *Flash-Crowds*, *SlashDot* o indisponibilidad temporal, algunos *websites* están empezando a utilizar la tecnología CDN para acelerar el *video browsing*. Según Buyya et al (2008, p.3), consideran que CDN es una red de servidores diseñados para entregar de manera distribuida el contenido en caché: "gracias a su funcionalidad de ser una colaborativa colección de elementos de red que abarcan la Internet, donde el contenido se replica sobre los recursos de varios *mirrored Web servers* y *cache servers* (conocidos también como *edge* o *surrogate servers*), esparcidos por todo el mundo para mejorar la capacidad de respuesta y la localidad de contenido a expensas de un aumento en el precio del servicio (fig. 1), con el fin de realizar la entrega transparente y eficaz de los contenidos a los usuarios finales".

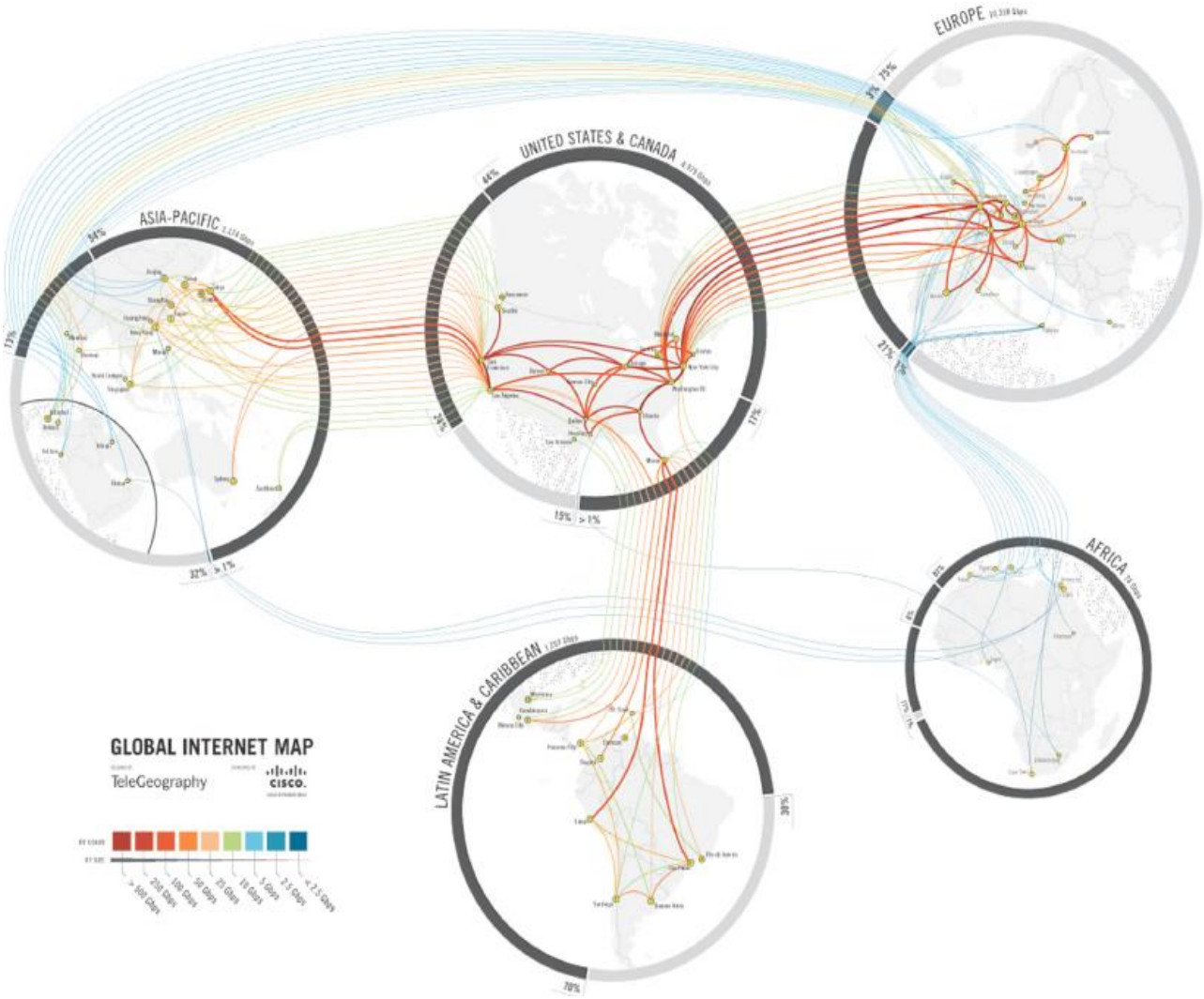


Figura 1. Tráfico de Internet a Nivel Mundial (2010) (Fuente: A. Hurtado)

Las solicitudes de cliente son redirigidas a la ubicación más adecuada en función de criterios relacionados con la red *"network-related"*, como el volumen de tráfico, calidad y proximidad, así como los criterios orientados al servicio *"service-oriented"*, tales como la carga del servidor, tiempo de respuesta y la disponibilidad. La colaboración entre componentes de las CDNs distribuidas puede ocurrir sobre nodos de ambos entornos homogéneos y heterogéneos. Las CDNs tradicionales tienen éxito en apoyar el tráfico de Internet, pero requieren esfuerzos significativos en inversión e implementación y son en gran medida específicas para una aplicación (Papagianni et al, 2013).

2.1.1 Conceptos Básicos de CDN

Hoy en día la tecnología CDN actúa sobre varios niveles de la tradicional pila del protocolo de red, contando con un almacenamiento de contenido (*caching)* dinámico y proactivo, en la implementación automática de aplicaciones y la migración en el borde de la red, para que sea este contenido el más próximo a los usuarios finales. Las réplicas de contenido en un CDN se distribuyen geográficamente, para permitir la entrega rápida y fiable a cualquier ubicación donde se encuentre el usuario final. A través de los servicios de CDN, un contenido actualizado puede ser adquirido por los usuarios finales a nivel local en lugar de forma remota (Cardellini et al, 2002).

La primera solución adoptada para distribuir el contenido a través de la Internet consiste en la creación de reflejos "*mirroring*". Esta técnica replica de forma estática contenido web en muchos lugares a través de la Internet. El mecanismo de selección de réplicas, de una lista de servidores, fue automatizado para hacerlo más adecuado y se hizo transparente para los usuarios finales con la introducción de los sistemas de servidores web distribuidos. Con la introducción de las técnicas de *Proxy Caching* para difundir el contenido a través de la Internet, los cuellos de botella a nivel de servidor y en los puntos de interconexión se redujeron considerablemente (Bartolini et al, 2004), aunque no garantizan una completa capacidad de control de estos sistemas por parte del proveedor de contenido, debido a la ausencia de una capa automatizada que lleve a cabo medidas en el servidor y que solicite redirección.

En CDN se utilizan dos tipos de servidores: *origin server* y *replica server*. (fig. 2)

- El *origin server* es aquel servidor en el cual reside la versión definitiva del contenido y es actualiza por el *content provider*. El *origin server* se comunica con los *replica servers* distribuidos para actualizar el contenido almacenado en ellos.

- Por otro lado, el *replica server* almacena una copia del contenido, pero puede actuar como una referencia autorizada de las respuestas que entregue el contenido de los clientes. Un *replica server* dentro de un CDN puede servir como *media server, Web*

server o como un *cache server*. Un *media server* puede servir cualquier contenido digital y codificado.

En un CDN, al mover el contenido desde múltiples servidores ubicados en el borde de la Internet, se obtiene un modelo mucho más escalable de distribución de la información y servicios a usuarios finales, que es el llamado *edge delivery*. Es por esto que los servidores Web (a los *CDN cache servers* también se los llama *edge servers* o *surrogates*), los cuales se encuentran repartidos en los extremos de las redes de Internet, dispersos por todo el mundo. La CDN es la encargada de distribuir el contenido a cada uno de estos *surrogates* remotos, de tal forma que el contenido final llega al usuario de forma fiable y en unos tiempos (retardos) aceptables (Buyya et al, 2008). Las CDNs distribuyen el contenido a los *edge servers*, de tal manera que todos ellos comparten el mismo contenido y la misma URL. Las solicitudes de cliente son redirigidas al más cercano y óptimo de los *edge servers* y distribuye el contenido requerido a los usuarios finales. Además, los *edge server* de una CDN en conjunto se los conoce como *Web cluster* y envían información contable del contenido entregado directamente al sistema de contabilidad de la CDN a efectos de registrar información como reportes de tráfico o con propósitos de facturación y de esta manera se consigue un servicio transparente para los usuarios finales

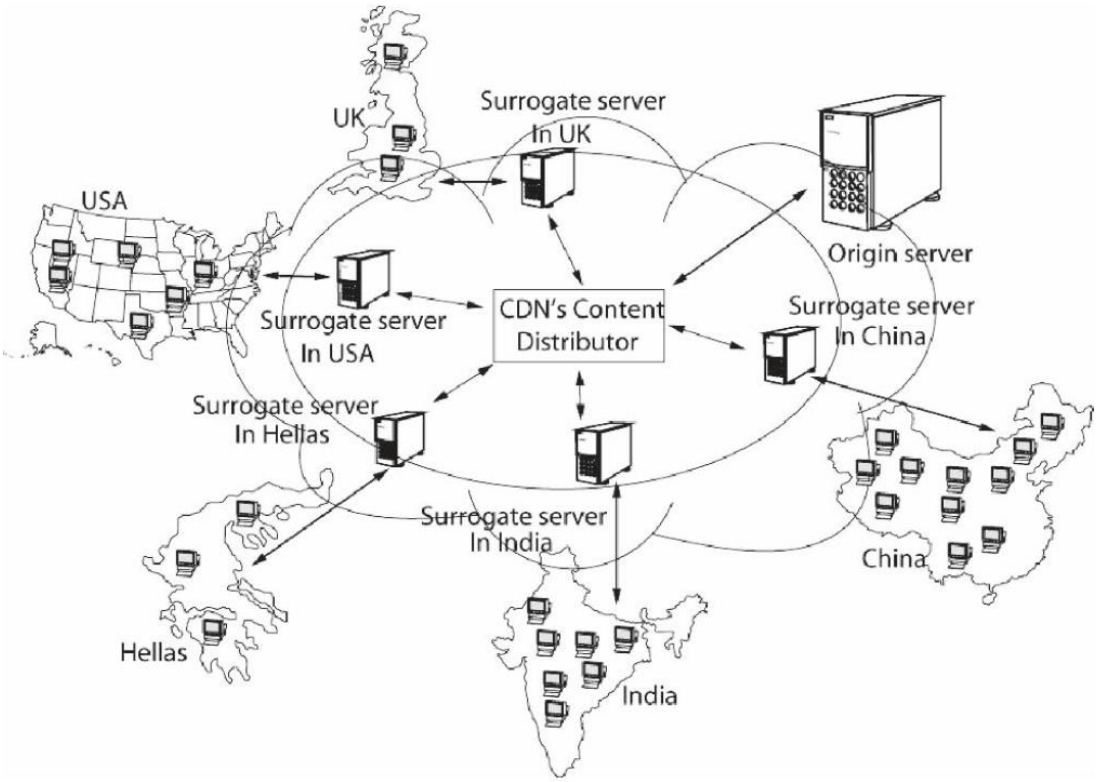


Figura 2. Modelo de un CDN (2005) (Fuente: R. Buyya)

2.1.2 Definiciones

En el contexto de CDN, la entrega de contenido describe la acción de servir contenido basándose en solicitudes de los usuarios finales. El contenido se refiere a cualquier tipo de dato digital y consta de dos partes principales: los medios codificados y la *metadata* (Verma, 2002). Dentro de los medios codificados se incluye datos de medios estáticos, dinámicos y continuos (ej. audio, video, documentos, imágenes y páginas Web). *Metadata* es la descripción del contenido, la cual permite la identificación, el descubrimiento y la administración de los datos multimedia, aparte de esto también facilita su interpretación.

El contenido puede ser pre-grabado o adquirido de fuentes con transmisión en vivo; a su vez pueden ser datos persistentes o transitorios dentro del sistema (Verma, 2002). Las CDNs pueden ser vistas como una nueva superposición virtual en el modelo de red de

referencia *Open Systems Interconnection* (OSI). Esta capa proporciona servicios de red de superposición que dependen de los protocolos de la capa de aplicación, tales como *Hyper Text Transfer Protocol* (HTTP) o *Real Time Streaming Protocol* (RTSP) para el transporte (Gilmore et al, 2004) y que en conjunto con (RTP) se ha convertido en el protocolo más común para la transmisión de vídeo dentro de los sitios web, y en verdad funciona bastante bien, ya que aprovecha el uso correcto de las técnicas de *caching* para apoyar la distribución de fragmentos de contenido y mejorar la fiabilidad en el rendimiento; a veces se utilizan varias velocidades de bits para optimizar la experiencia del usuario en el dispositivo receptor.

Aparte de la correcta distribución de los *replica servers* sobre el borde de la red, para el diseño de una red CDN, se requiere de un conjunto de servicios y capacidades de alto soporte. Con la condición necesaria de ser eficaz para un significativo número de usuarios y para un área considerablemente amplia, los *edge servers* deben ser desplegados a través de miles de redes y en diferentes lugares geográficamente separados. Lograr el más óptimo rendimiento y la mejor confiabilidad, dependerá de la granularidad con que distribuyan el contenido los *edge servers* (Bartolini et al, 2004).

Existen tres principales entidades dentro de un sistema CDN y estas son las siguientes: *content provider,* los usuarios finales y el *CDN provider.*

- Un *content provider* o cliente es aquel que delega el nombre del espacio *Uniform Resource Locator* (URL) del objeto Web a ser distribuido. El *origin server* del proveedor de contenidos es aquel que mantiene dichos objetos.

- Los usuarios finales o clientes son las entidades que tienen acceso al contenido almacenado en el sitio web del *content provider*.

- Un *CDN provide*r es una organización propietaria o empresa que proporciona las instalaciones de infraestructura necesaria a los *content providers*, con el fin de distribuir el contenido de manera oportuna y confiable (fig. 3). A más utilizan a los *caching* y *replica servers* ubicados en diferentes puntos geográficos para replicar el contenido.

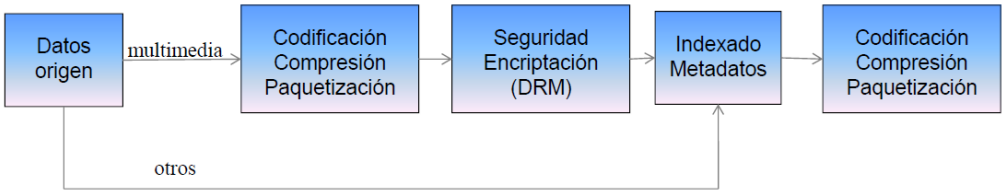


Figura 3. Procesado de datos en el proveedor CDN. (2013) (Fuente: B. Molina)

2.1.3 Evolución de CDN

Desde que los proveedores de contenidos empezaron a utilizar y optimizar sus sitios Web mediante las mejoras que trajo la tecnología de CDN, muy pronto se dieron cuenta de su utilidad para recibir una mayor fiabilidad y escalabilidad sin la necesidad de mantener una infraestructura costosa a través de un *Datacenter* propio. Es por esta razón, que poco a poco se fueron dando varias iniciativas, para emprender aún más el desarrollo de la infraestructura CDN tal como la conocemos hoy en día. Con el pasar de los años, varios investigadores como por ejemplo los del MIT quienes en su afán de combatir el problema del *flash-crowd* y entregar una solución de fondo al grave problema de *caching* que sufrieron varios sitios Web que fueron inundados por peticiones durante algún evento de gran proporción, crearon lo que hoy en día es la red CDN más grande conocida como *Akamai Technologies* y así varios grupos más de científicos han venido desarrollando un conjunto de algoritmos de vanguardia para el enrutamiento inteligente y la replicación del contenido a través de una amplia red de servidores distribuidos por todo el mundo, logrando así que varias empresas se convirtieran en especialistas en proveer una rápida y confiable distribución de contenido, logrando así que CDN sea un enorme mercado para generar grandes ingresos (Buyya et al, 2008) (fig. 4). Por tal razón se hablará sobre algunos de los años más importante durante la historia de CDN, su enorme evolución como tecnología y como logró ser lo que es hoy en día.

1998: En este año aparece la primera CDN, en vista que ciertas empresas se dan cuenta de que podían ahorrar dinero colocando sus sitios Web dentro de una red CDN, economizando

así considerablemente en la inversión de hardware costoso, consiguiendo incrementar la confiabilidad y escalabilidad en la prestación de información (Pallis & Vakali, 2006).

1999: Durante este año surgieron algunas de las compañías más prestigiosas en técnicas de CDN (tales como *Akamai y Mirror Image*) las cuales se convirtieron en especialistas para proveer una rápida y confiable entrega del contenido Web, logrando así obtener grandes beneficios dentro de su infraestructura y en el servicio que prestaban al usuario.

2000: En el transcurso de este año, sólo dentro del territorio de USA, el enorme mercado de CDN ya generaba aproximadamente $905 millones con la expectativa de llegar a los $12 billones para el 2007 (Pallis & Vakali, 2006).

2001: En este año se suscitó un evento que está dentro de la categoría del *flash crowd* (sucede cuando numerosos usuarios tratan de acceder al mismo tiempo a un solo sitio Web) (Jung et al, 2002), este evento marcó desde cierto punto de vista un inicio para la estrategia de mercado sobre el cual se enfocarían los servicios CDN y que servirán para evitar estos inconvenientes. Uno de los ejemplos históricos, fue el evento que se produjo en ese país el 11 de septiembre del 2001, cuando una enorme cantidad de usuarios inundaron con peticiones recurrentes de información a los sitios más populares de noticias y esto dio como resultado problemas de almacenamiento, que a la final causaron que los sitios estén indisponibles.

2002: Esta época se caracterizó debido a que ISPs de gran categoría como AT&T, comiencen con tendencias de negocio inteligentes como las de empezar a forjar su propia funcionalidad CDN, con la idea de proporcionar servicios personalizados (Pallis & Vakali, 2006).

2004: Ya para estos años la popularidad de CDN empezó a expandirse y ya se tenían registradas alrededor de 3.000 empresas empleando la funcionalidad de CDN para su negocio, lo que trajo a CDN una inversión aproximada de más de $20 millones mensuales (CDN Market, 2005). Y esto en favor de que muchos proveedores de CDN estaban tratando de mover los servicios web (como *Microsoft .NET y Java 2 Platform Enterprise Edition*) para que se familiaricen más a las actividades cotidianas de los usuarios.

2005: Con los servicios de CDN ya fortalecidos, la rentabilidad de ingresos tanto para el *streaming* de vídeo y radio por Internet, se estimaron para que vayan creciendo en un 40%,

y que sea un servicio donde se invierta $450 millones para la entrega de información noticiera, repositorios de películas, eventos deportivos, música y entretenimiento (CDN Market, 2005).

Continuando con el desarrollo de su historia, durante estos periodos fueron apareciendo las distintas generaciones de CDNs, que se describen a continuación:

2.1.3.1 CDN 1.0

La primera generación de CDN surgió hace más de una década para que los sitios de Internet puedan mantener el ritmo ante el crecimiento desmedido del uso de Internet y mayor ancho de banda. Esto llevó al desarrollo del concepto de "*web acceleration*" y eran CDNs que almacenaban frecuentemente peticiones de contenido en servidores distribuidos lo más cerca de los puntos de consumo, para disminuir la demanda y reducir la latencia. Dentro de esta generación ya se introdujeron conceptos claves, como el despliegue del *intelligent routing* y el *edge computation*, y llevado al crecimiento de otro de los grandes actores globales en CDN conocido como *Limelight*, que desarrolló en gran medida un fenómeno estadounidense debido al enorme liderazgo de ese país en desarrollos sobre Internet (Schwarz, 2012).

En esta generación a más de realizar solamente procesos de *caching*, realmente se introdujo la capacidad de manejar un *caching* del contenido más veloz y dinámico, así como el del material estático tradicional mediante técnicas de "*pushing*" sobre los objetos con datos pesados para que sean distribuidos en tiempo real a los diferentes cachés. Aquí se utilizaron técnicas emergentes como la desarrollada por las herramientas de *Linux* basadas en *Squid*, el cual basó su funcionamiento en el no tan sofisticado *proxy caching* que se conoce hoy en día, para automatizar el frecuente proceso de descarga de páginas web de los servidores distribuidos. Principalmente, la primera generación de CDN se centró sobre el uso de los documentos Web estáticos o dinámicos (Schwarz, 2012).

2.1.3.2 CDN 2.0

Esta generación de CDN empezó a surgir cuando el vídeo comenzó a ser integrado dentro de los sitios web, eso sí, en un principio el contenido solamente era pre-grabado y entregado bajo demanda. En cierto sentido, esto significaba que el vídeo era solamente un gran archivo de datos. En el uso del contenido pre-grabado no se conocía en detalle si el usuario deseaba dejar de ver el archivo de video en cualquier momento. Por lo tanto, cuando surgió la necesidad y la demanda de manipular el video, fue cuando se requirió del *streaming*, y esto estimuló el desarrollo de varios CDNs optimizados para la entrega de este servicio, como algunos de los grandes referentes dentro de los servicios *over-the-top* y que lo son *Hulu* y el popular *Netflix*, los cuales están desarrollando su propia infraestructura CDN, pero que en su gran mayoría aun continúan utilizando la infraestructura proporcionada por un proveedor especializado en CDN.

Luego llegó la creciente demanda de transmisión de video en vivo, y en un principio parecía que este sería el final del camino para el uso del *caching*. En definitiva, se concluyó que el tráfico de video en vivo no podía ser almacenado en caché como se lo hizo previamente para el contenido grabado, y que por lo tanto, la infraestructura de la CDN tenía que ser modificada para dar cabida a las elevadas tasas de transferencia en la transmisión punto a punto entre la fuente de origen del contenido y el usuario final. El problema fue que los costos involucrados en la construcción y el mantenimiento de la transmisión en vivo para eventos populares serían extremadamente costosos, así como técnicamente exigentes.

Esta segunda generación de CDN se centró en el *Video on-Demand (VoD), news on-demand*, la transmisión de audio y video con alta interactividad para el usuario, así como también dedicaron recursos de investigación sobre los métodos de entrega de contenido para los dispositivos móviles.

2.1.3.3 CDN 3.0

Esta siguiente generación de CDN se ha venido anticipando durante el transcurso de estos últimos años y trae gran cantidad de desarrollo e investigación, los expertos asumen que podría ser conocida como *community-based CDNs* y que permitiría un poco más de control para el usuario promedio, por lo cual estos tienen que ser investigados antes de que *Wireless Mesh Network* (WMN) se convierta en una parte completamente integral de las redes de contenido (Buyya et al, 2008). Se prevé que las redes *multihomed* integradas sean funcionales a finales de la década, basado en el progreso que ha traído la investigación en WMN, la selección de red y otras cuestiones pendientes de investigación relacionadas, teniendo en cuenta que solo las infraestructuras altamente heterogéneos basadas en diferentes tipos de redes serán capaces de hacer frente a esto y proporcionar el apoyo necesario.

Los operadores se están alejando paulatinamente de las redes cerradas como el IPTV, ya que todos los nuevos conceptos que trae esta generación, muestran una clara oportunidad de reutilizar la infraestructura existente; debido a la fuerte presión para agregar más y más canales, hace que esto sigua creciendo junto con el concepto de *TV-Everywhere* (incluida la movilidad de dispositivos), por lo que el tráfico *unicast* aumentará inevitablemente (Schwarz, 2012). Como por ejemplo, el concepto de "*Multicast To The Home*" que está siendo desarrollado por *Time Warner Cable* (TWC) que es una de las operadoras más avanzadas en servicio de cable, a través del proceso de *Adaptive Bit-Rate* (ABR) sobre un *Single Program Transport Stream* (SPTS) para tomar ventaja de la infraestructura multicast existente, fortalecerla y obtener beneficios. Un aspecto importante de la generación CDN 3.0 es que será el surgimiento de las *federated CDNs*.

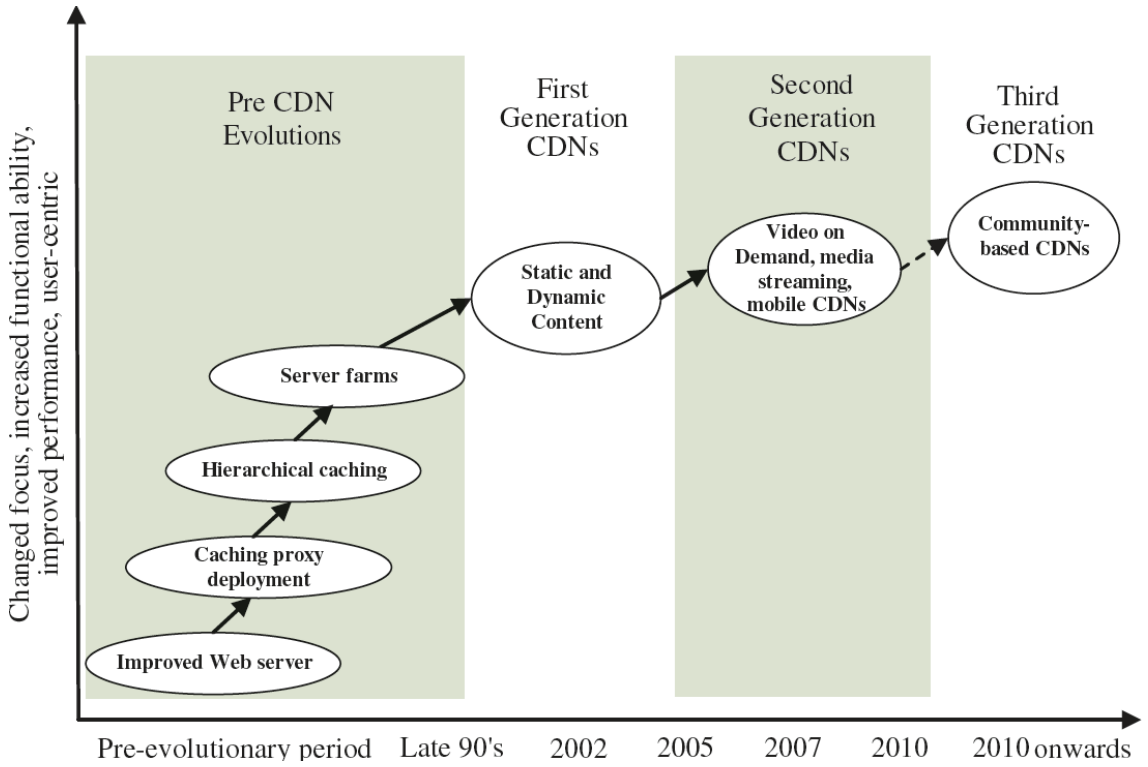


Figura 4. Evolución de CDN. (2005) (Fuente: R. Buyya)

2.1.4 Tipos de CDNs

Existen varios tipos de Redes de Distribución de Contenido (CDN), a continuación se describirán cada uno de ellos y se mostraran ejemplos de sus principales referentes:

- ***Pure-play CDN*.**

Son el tipo de CDNs que proporcionan la entrega de contenido de audio y vídeo *over-the-top* (OTT) sin que un ISPs esté involucrado en el control y distribución de los contenidos en sí. Las *Pure-play CDNs* entregan contenido a través de la red de varios ISPs (por ejemplo, Akamai) o por infraestructura propia (en el caso de, Limelight Networks o Level 3) (Pathan et al, 2014).

- ***Carrier/Telco CDN*.**

Los proveedores de banda ancha y las empresas de telecomunicaciones, por ejemplo, Verizon, Telstra y AT&T, que proporcionan la entrega de contenido como un medio para reducir las demandas sobre el *backbone* de la red y reducir la inversión en infraestructura, utilizando hardware y software de los vendedores, por ejemplo, Cisco, Juniper o Alcatel-Lucent (Pathan et al, 2014).

- ***Managed CDN*.**

Para este caso, los *Pure-play CDNs* pueden entregar soporte a los *carriers* para construir y gestionar el componente CDN de la red del operador a través de su personal de servicios profesionales, por ejemplo, *Limelight Deploy*. Este enfoque permite que se aproveche la experiencia, infraestructura y software de un *pure-play CDN* (Pathan et al, 2014).

- ***Licensed CDN*.**

Los *Pure-play CDN* también pueden proporcionar software CDN para la integración, pruebas y despliegue de la infraestructura de un *carrier*. Aunque la asistencia de integración está inicialmente disponible desde el proveedor *pure-play CDN*, el *licensed CD*N estará siendo administrado por el operador de la red. Por ejemplo, *EdgeCast*, *Highwinds* y *Akamai Aura* proporcionan productos para este tipo de CDN (Pathan et al, 2014).

- ***Federated CDN*.**

Múltiples CDNs pueden interconectar sus redes y competir directamente en contra de los *pure-play CDN*. Será interesante y provechoso para un *content provider* entregar su contenido a uno de los participantes de la federación. *Cisco* y *Highwinds* están trabajando para ser *Federated CDNs* (Pathan et al, 2014).

- ***Academic CDN.***

A diferencia de las CDNs de tipo comercial, las CDNs académicas utilizan en gran parte tecnologías P2P, que las hace muy eficaces para manejar contenido estático, pero son incapaces de manejar el tipo de contenido generado dinámicamente, debido a que este no se lo puede almacenar. La entrega de contenido sigue un enfoque descentralizado y la solicitud de carga se extiende por todos los *hosts* participantes, razón por la cual el sistema puede manejar errores en los nodos y sobrecargas repentinas. Como ejemplo existen tres CDN académicas conocidas como *CoDeeN*, *Coral* y el *Globule* (Buyya, 2008).

2.1.5 Situación Actual de la Tecnología

CDN es una tecnología que permite una gran conectividad, es por esto que los *content provider* necesitan de alguna manera integrarse con los principales *Internet Providers* o de ser el caso, llegar ellos mismos a ser quienes entreguen servicios de CDN, para obtener una conectividad a gran escala, eficiente, confiable y con gran diversidad que les permita llegar a muchos más usuarios. Además de esto se debe tener en cuenta que las CDNs deben ofrecer soluciones hibridas, que permitan mejorar el rendimiento del tráfico de *Streaming*, así como tratar de abaratar siempre en costos. Por estadísticas se identificó que en el 2014 el 56% del tráfico en la Internet se relacionó con video (Shen, 2010). Como consecuencia inmediata del crecimiento del tráfico de vídeo, todos los mercados que cuenten con tecnología CDN y que distribuyan de manera eficiente el contenido de los usuarios, tendrán un futuro prometedor para la inversión; ya que desde hace varios años se ha ido integrando también hacia aplicaciones inmersas con el uso de las redes sociales, a tal punto que estos están en medio del despliegue de sus redes CDN.

Durante más de una década, las redes de distribución de contenidos (CDN) han jugado un papel crucial en el funcionamiento eficaz de la Internet, ayudando a mitigar problemas como la congestión de la red, pérdida de paquetes, *jitter* y retardo. El aumento del número de usuarios en la Internet y un mayor uso del ancho de banda, han tenido un papel importante

en la transformación del mercado para la prestación de servicios de CDN. Así como el paisaje CDN ha dado paso a nuevos tipos de proveedores - incluyendo una amplia gama de proveedores de "*pure play*" y un número cada vez mayor de operadores de redes CDN. También se ha marcado últimamente los usos potenciales de la CDN en nuevos mercados, incluyendo la distribución de contenido *over-the-top* (OTT) (fig. 5), a través del manejo de vídeo en dispositivos móviles y servicios en la nube (*Cloud*) (Drake, 2012).

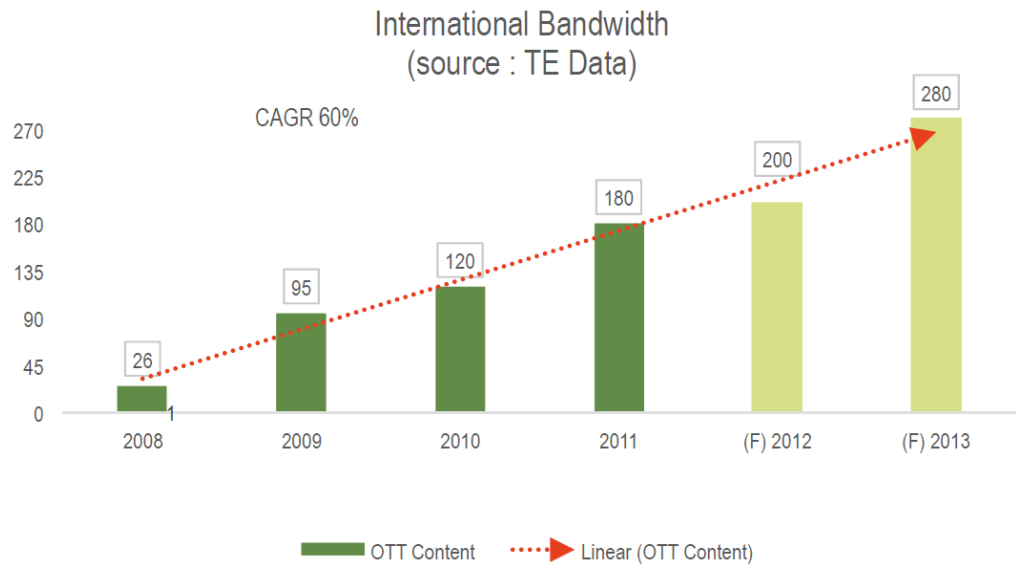


Figure 5. OTT Content. (2012) (Fuente: CDN 3.0 – B. Schwarz)

Una de las principales razones del crecimiento de CDN, es la rápida expansión de los servicios conocidos como los *Internet-based services*, los cuales incluyen el *online video*, *streaming media*, *online music*, *online games* y la transferencia de archivos de medios enriquecidos a través de la Internet, como páginas html, imágenes, documentos con formato o aplicaciones, lo que traerá una fuerte demanda sobre la actual infraestructura de Internet y una mejorar propuesta para incrementar la calidad de servicio (QoS). CDN se ha forjado como una solución ideal para aplacar todos estos problemas y satisfacer la necesidad en la entregar del contenido con la mejor calidad. El mercado mundial de CDN está impulsando el elevado crecimiento de soluciones de servicio de CDN, entre las regiones de U.S, *Asia*

Pacific (APAC) y la Unión Europa (UE) (Buyya et al, 2008). Se espera que el mercado de CDN crezca de $3.71 billones en el 2014 a $12,16 billones para el 2019, a una tasa *Compound Annual Growth Rate (CAGR)* del 26,8% (fig. 6). La región de APAC está creciendo rápidamente con una tasa compuesta anual CAGR del 38,2% desde el 2014 al 2019.

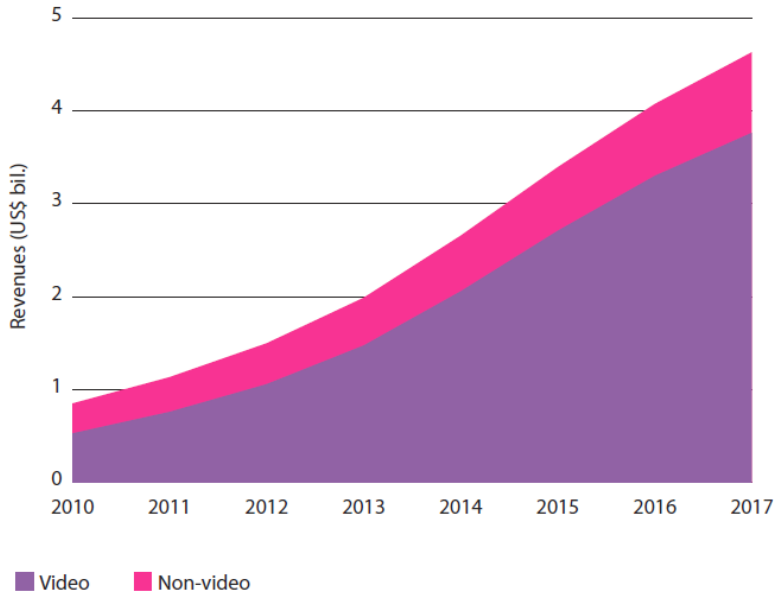


Figura 6. Global, CDN revenues, 2010-2017. (Fuente: Informa – C. Drake)

2.1.6 Mercados de CDN

El valor del mercado de CDN, depende fielmente de la diversidad de los participantes en la industria (fig. 7):

- Para los proveedores de servicios de telecomunicaciones y los propietarios de contenido con su propio CDN, el valor de un CDN reside en parte en su capacidad para mejorar la prestación de servicios al por menor y apoyar sus esfuerzos para ganar y retener clientes.

- Para los fabricantes de la industria y los proveedores de servicios de CDN gestionados, el valor de mercado se basa en la demanda por parte de los operadores de telecomunicaciones, los propietarios del contenido y otras empresas, a tener su propio CDN.

- Para los proveedores de servicios de CDN comerciales, el valor del mercado depende de la continua necesidad entre los propietarios de contenidos y las empresas en línea para una amplia gama de servicios de gestión y distribución de contenidos. Estos se extienden más allá del básico trato por GB "*bit delivery*" e incluyen los servicios *premium*, como la aceleración de aplicaciones, seguridad en línea y optimización de recursos para los sitios web.

- Por último, para los propietarios de los contenidos, el valor de CDN radica en la medida en que proporcionan una experiencia en línea confiable y de alta calidad para los usuarios finales de su contenido.

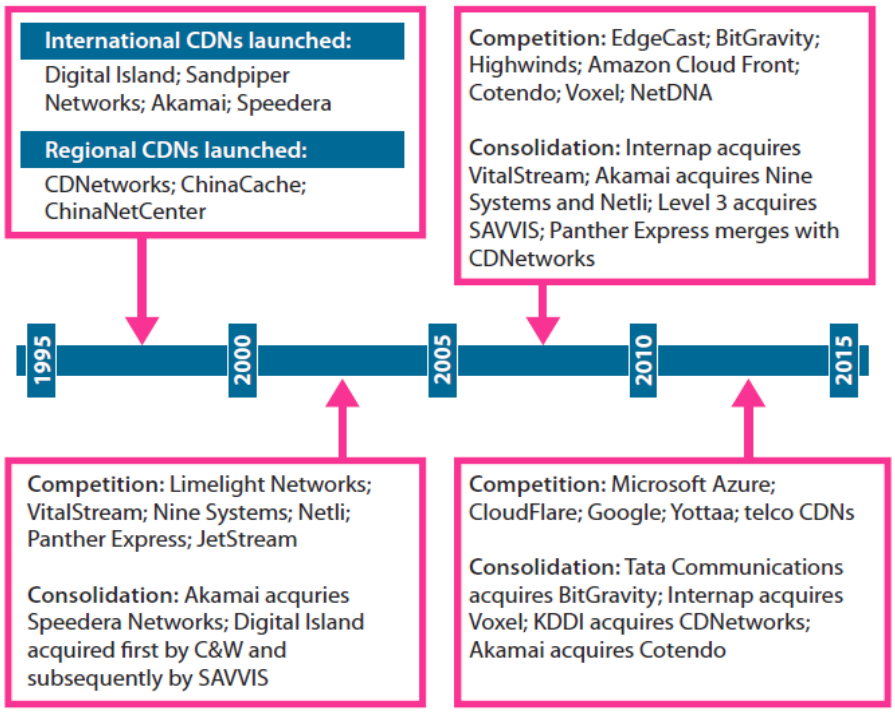


Figura 7. CDN market timeline. (2012) (Fuente: Informa T&M – C. Drake)

Se ha evaluado el potencial de crecimiento y las tendencias específicas del mercado para la entrega de contenido global. Es por esto que *Informa* declara que para el año 2017, se espera que el mercado de los servicios de CDN comerciales lleguen a valer US $ 4.63 billones, lo que refleja un aumento de tres veces a partir de 2012. El video será el mayor contribuyente de ingresos para el crecimiento de una CDN comercial y que representan el 81% de los ingresos totales (Drake, 2012).

Tabla 1.

M&A Transactions Announced GEA, LLC - A. Solganick (2014)

Date Announced	Target/Issuer	Buyers/Investors	Total Transaction Value
6/16/2014	**TW Telecom**	Level 3 Communications	$3B
2/18/2014	**Prolexic Technologies**	Akami	
2/9/2014	**RagingWire Data Centers**	NTT Communications	$350M for 80%
1/24/2014	**Arkadin International**	NTT Communications	
1/10/2014	**Virtela Technology Services**	NTT Communications	

Generation Equity Advisors es una firma de asesoría enfocada exclusivamente en los sectores de la industria de medios digitales y Tecnología global, que tiene como meta enfocarse en la inversión de la banca y fusiones de las diferentes firmas en el mercado. Según su estadística publicada en el 2014, informa que hay una serie de transacciones de M&A y el capital creciente en el mercado de CDN. Por ejemplo, *NTT Communications* adquirió tres empresas con experiencia en el mercado CDN, incluyendo *Raging Wire, Arkadin y Virtela*. Por lo tanto, estima que habrá una mejora del mercado M&A para empresas que buscan expandirse dentro de CDN a través de fusiones, adquisiciones o salidas en 2014 y 2015 (Generation Equity Advisors, 2014).

Tabla 2.

Publicly Traded Content Delivery Networks (Selected) – Valuation Table. (2014)

CDN Industry Sector
Public Comparables

CDN	Ticker	Stock Price (as of July 14, 2014)	Market Cap	EV/Revenue	EV/EBITDA
Rackspace	RAX	32.00	4.5 B	2.7	10.5
Akamai	AKAM	60.00	10.7 B	6.4	17.9
InterNAP	INAP	6.75	340 M	2.3	14.9
Limelight Networks	LLNW	2.90	285 M	1.0	N/A
Level 3 Communications	LVLN	45.50	10.8 B	2.9	11.2
			Mean (Average)	**3.06**	**13.63**

2.1.7 Servicios Emergentes a CDN

- ***Unified Content Network***

Son servicios utilizados para hacer transformaciones y procesamiento de contenido, como servicio de infraestructura tienen que ser accesibles por el usuario, en este caso los distribuidores del servicio han implementado los recursos de red provistos por *Content Service Network* (CSN) como un servicio de distribución para canales de servicio de valor agregado (Valloppillil & Ross, 1998), que actúan como otra capa de infraestructura de red construida sobre CDN y proporcionan próxima generación de servicios de CDN, con el fin de prestar el servicio de sus aplicaciones como un servicio de infraestructura.

- ***Dynamic Content Delivery***

Es la generación de contenido bajo demanda utilizando aplicaciones Web basadas en requerimientos de los usuarios finales, como por ejemplo: scripts, animaciones, DHTML o XML. La generación dinámica de páginas web puede ser realizada con el uso de técnicas de *hosting* de aplicaciones web escalables, como: *Edge Computing*, *context-aware data caching*, *content replication*, *content blind data caching* o soluciones propietarias.

- ***Web Services Hosting***

Como describe el W3C's *Web Services Architecture Working Group* "Un servicio Web es una aplicación de software identificado por un URI, que provee a los usuarios de Internet un sistema para poder almacenar información, imágenes, vídeo o cualquier contenido accesible vía web. Sus interfaces y especificaciones son capaces de ser definidas y descritas como artefactos XML. Un servicio Web soporta interacciones directas con otros agentes de software que utilizan mensajes *XML-based* y que son intercambiados a través de protocolos basados en la Internet. (Jenny at al, 2004).

Algunas empresas externalizan su gestión de la infraestructura de TI a proveedores terceros. En este caso, los servicios prestados por el proveedor de terceros pueden ser o no ser *Cloud-enabled*. Por lo general este tipo de modelos de implementación son similares a las *private clouds*, con la única diferencia de que éstos están alojados o gestionados por otra empresa. También existen ciertos proveedores de servicios que ofrecen servicios de *hosting* gestionados en *public clouds*, como por ejemplo: Amazon y Google.

Los Servicios Web básicos combinan el poder de dos tecnologías: XML, que es el lenguaje universal de descripción de datos y el protocolo de transporte HTTP que es ampliamente soportado por navegadores y los servidores Web. El coste medio de los servicios proporcionados por un proveedor de CDN es superior comparado con servicios normales de *hosting* (Jenny at al, 2004).

Web services = XML + *transport protocol* (*such as HTTP*)

- El uso de análisis de XML, la serialización de *Java*, copia de reflexión, copia clon.
- Red de entrega de aplicaciones (*Application Delivery Network* - ADN) para albergar las aplicaciones .NET y J2EE
- Capacidad de aprovisionamiento de red (*Capacity Provisioning Network* - CPN) para capacidades de caché de negociación

- ***Service-Oriented Architecture (SOA)***

Cada cierto tiempo, la introducción de una nueva tecnología o conjunto de tecnologías crean en la industria de TI lo que algunos llaman una "disrupción" y se lo conoce así, porque esta nueva implementación cambia radicalmente la manera en que se hacen los negocios. Un buen ejemplo de disrupción tecnológica fue la introducción de *Service-Oriented Architecture* a mediados de los años 2000.

SOA proporciona una capa de abstracción que permite a una organización continuar aprovechando su inversión en TI envolviendo estos activos existentes como los servicios que proporcionan las funciones del negocio. Las organizaciones pueden potencialmente continuar recibiendo valor de los recursos existentes en lugar de tener que reconstruir desde cero (Jenny at al, 2004).

- Se espera que la gestión de contenido pueda ser motivado por las preferencias del usuario.
- Personalización de usuario basado en la minería de datos.

Muchas de las afirmaciones hechas por los proveedores de SOA con respecto al potencial de SOAs eran pertinentes y válidas. Sin embargo, el entusiasmo que rodeó SOA causó comentarios que exageraron algunas de sus capacidades. Así que a pesar de los éxitos relativos, las SOAs no siempre estuvieron a la altura de las expectativas de los clientes.

2.1.8 Sistemas Relacionados con CDN

Existen tres sistemas distribuidos que tienen algunas características en común con las CDNs, estos con: *Data Grids, Distributed Databases, and Peer-to-Peer (P2P) networks*. Estos tres sistemas serán descritos en términos de requisitos, funcionalidades y características.

- ***Data Grids***

Un *Data Grid* (Krishnan et al, 2000) es un entorno de cómputo intensivo de datos que proporciona servicios a los usuarios en diferentes lugares para descubrir, transferir y manipular grandes conjuntos de datos almacenados en repositorios distribuidos. Siendo su objetivo general el de reunir recursos distribuidos existentes para obtener una ganancia de rendimiento a través de la distribución de datos (Stamos et al, 2006). Proporciona dos funciones básicas: la primera función es un mecanismo de alto rendimiento con una confiable transferencia de datos y la segunda función es un mecanismo de descubrimiento escalable de réplicas y de gestión (Bartolini et al, 2004). Consta de recursos computacionales y repositorios de almacenamiento en diferentes lugares de todo el mundo conectados por redes de alta velocidad y transfiere información a otros sitios de almacenamiento a través del mecanismo de replicación de datos.

Las *Data Grids* están especialmente dirigidas a instituciones que se reúnen para compartir recursos con alguna meta común formando una Organización Virtual (VO) o sobre las grandes aplicaciones científicas tales como experimentos de física de alta energía en el Gran Colisionador de Hadrones (Pathan, 2009)), proyectos de astronomía en Observatorios Virtuales (Freedman, 2004) y también en proyectos de simulación de proteínas como *BioGrid* que requieren el análisis de una gran cantidad de datos. Promueven un entorno a los usuarios para analizar datos, compartir los resultados con los colaboradores y mantener información de primera mano.

- ***Distributed Databases* (DDB)**

Un *Distributed Database* (DDB) (Cardellini et al, 2002) es un conjunto organizado de datos distribuidos lógicamente en varias ubicaciones físicas, formado mediante la división de una única base de datos o por un conglomerado de múltiples bases de datos existentes y gestionado por una única entidad autorizada como lo hace CDN. Puede ser almacenada en múltiples ordenadores situados en la misma ubicación física o puede dispersarse en una red de ordenadores interconectados. Cada computadora en el sistema de base de datos distribuido es un nodo; dependiendo de la situación el nodo del sistema de base de datos distribuido

actúa como un cliente, servidor o ambos. DDB se caracteriza por proporciona algunas aplicaciones como el procesamiento distribuido de transacciones o consultas (Radoslavov et al, 2001), la optimización de consultas distribuidas y la gestión eficiente de los recursos.

Cada sitio tiene un grado de autonomía que es capaz de ejecutar una consulta local y participar en la ejecución de una consulta global, con transacciones transparentes que mantienen su integridad a través de múltiples bases de datos. Las *Distributed Databases* han sido desarrolladas para atender las necesidades de las grandes organizaciones que necesitan reemplazar sus existentes sistemas de bases de datos centralizadas, cuando requieren incrementar unidades organizativas.

- ***P2P Networks***

Las *P2P networks* (Duan, 2012) son redes de contenido y compartición directa de recursos informáticos o archivos, que se centran principalmente en la creación de estrategias eficientes para localizar archivos en particular dentro de un grupo de *peers*, para proporcionar transferencias fiables de este tipo de archivos en caso de alta volatilidad y para gestionar el tráfico pesado, diseñadas para no requerir ninguna autoridad intermediaria y/o central. Se caracterizan como las redes de recuperación de información que se forman por agregación *ad-hoc* de los recursos para formar un sistema total o parcialmente descentralizado. Dentro de un sistema P2P, cada *peer* es autónomo, ya que puede unirse o abandonar la red en cualquier momento, lo que no sucede en CDN (Jamin et al, 2001), pero al contrario depende de otros *peers* para tener los recursos e información requeridos. Lo ideal en una red P2P es que no hay un punto central de control, haciéndola más tolerable a errores ya que no tiene un punto único de fallo.

Por lo tanto, las entidades participantes colaboran para realizar tareas tales como la búsqueda de otros nodos, localización o caché del contenido, solicitudes de enrutamiento, cifrado, recuperación, descifrando y verificar el contenido. Las redes P2P son más adecuadas para los proveedores de contenido individuales que no son capaces de acceder o permitirse un servicio común de CDN. Sirven también para mitigar eventos de tipo *flash crowds* que son causados por la demanda de archivos muy populares. Su operatividad está en total

contraste a lo que aplica CDN, donde el objetivo principal radica en el respeto de los requisitos de desempeño del cliente en lugar de compartición eficientemente de archivos o contenidos entre *peers*. Un ejemplo del sistema es *BitTorrent* (CDN Market, 2005) que es una aplicación muy popular de intercambio P2P.

2.2 TAXONOMÍA DE LAS REDES DE CONTENIDO - CDNs

Las redes de contenido se pueden implementar en una gran variedad de formas, sean estas a nivel de área local (LAN) o a través de redes de área amplia (WAN). Cuando se habla de soluciones locales estos son los conocidos *web clusters*, que normalmente albergan servicios *single site* y los *web farms*, que normalmente son utilizadas para alojar servicios *multiple sites*.

Mientras que las soluciones WAN incluyen: los *distributed web server systems*, que son utilizados para alojar sitios únicos o múltiples; los *cooperative proxy cache networks* que tienen la característica de ser una infraestructura de servicios para reducir la latencia en la descarga de objetos web) (Bartolini et al, 2004) y finalmente las *content delivery networks* CDN, sobre las cuales se describirá en detalle su taxonomía (fig. 8).

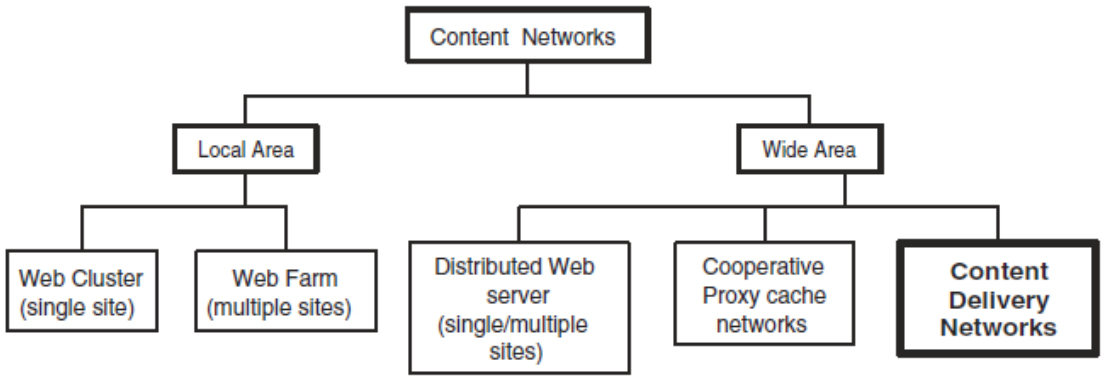


Figura 8. Taxonomy of Content Networks. (2004) (Fuente: N. Bartolini, et al)

2.2.1 Arquitectura y Componentes de CDN.

Dada la naturaleza heterogénea de los contenidos que se entregan dentro de CDN, se pueden adoptar diversas arquitecturas y tecnologías para diseñarla y desarrollarla, lo que requiere de un conjunto de servicios de soporte y ciertas capacidades, que colaboren junto con la distribución que realizan los *replica servers* en el borde de la red. Con el afán de que el servicio sea eficaz para un número significativo de usuarios y un área considerablemente amplia, los *edge servers* deben ser desplegados en miles de redes y dentro de diferentes lugares separadas geográficamente. Lograr un buen rendimiento y una confiabilidad adecuada dependerá de la granularidad con que se distribuyan los *edge servers*.

El establecimiento de una CDN requerirá del diseño de algunas características importantes y ciertas funcionalidades básicas (fig. 9).

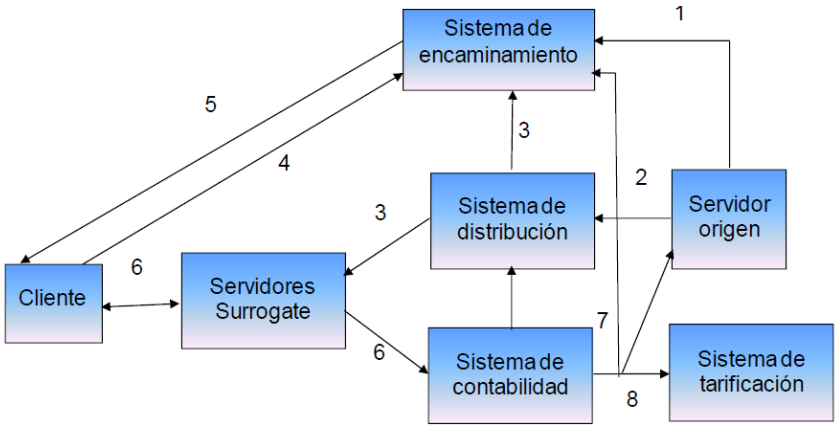


Figura 9. Arquitectura general de una CDN. (2013) (Fuente: B. Molina)

La arquitectura de CDN se basa en una actividad de medición que se realiza mediante routers de acceso cooperativo, para evaluar las condiciones del tráfico, la capacidad computacional y la disponibilidad de cada servidor réplica capaz de servir a una solicitud (fig. 10 y 12). Cuando una CDN es implementada exitosamente, puede acelerar el acceso del

usuario final a los contenidos, reducir el tráfico de red y reducir los requisitos de hardware del proveedor de contenidos (Buyya et al, 2008).

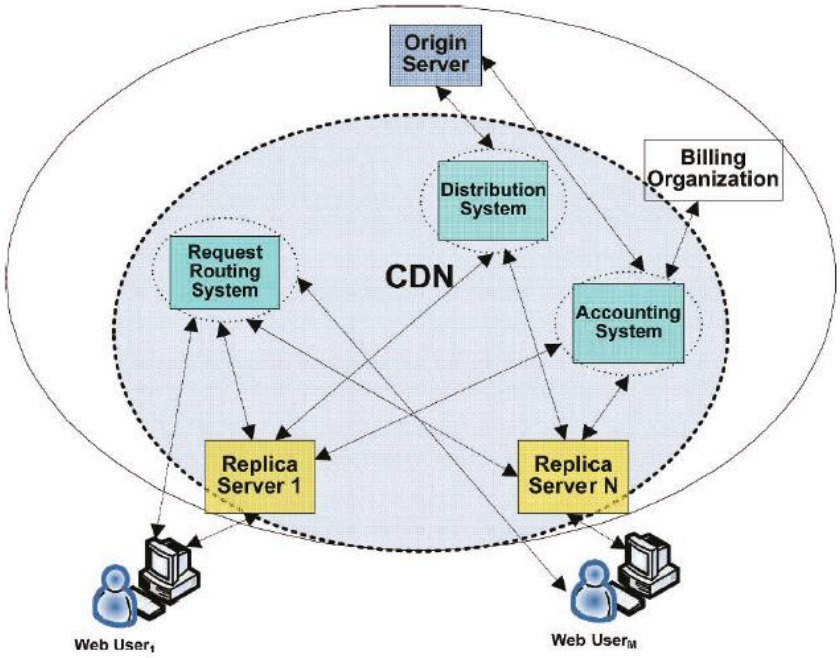


Figura 10. Componentes arquitectónicos de una CDN. (2008) (Fuente: Buyya)

Se considera que los temas relacionados con la taxonomía proporcionan un reflejo completo de las propiedades de las redes de contenido existentes. El objetivo principal de un servidor de replicación de información en CDN es el de evitar que grandes cantidades de datos atraviesan repentinamente enlaces posiblemente congestionados en la Internet. Es por esto que a la taxonomía de CDNs se presentará en base a cuatro aspectos (fig. 11):

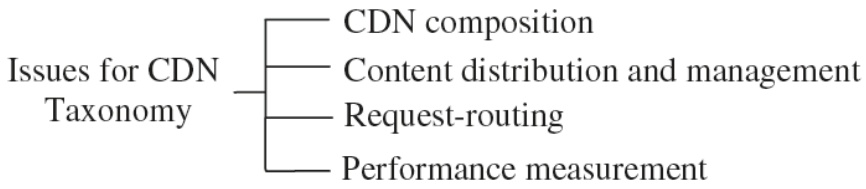


Figura 11. Issues for CDN taxonomy. (2009) (Fuente: A. Pathan y R. Buyya)

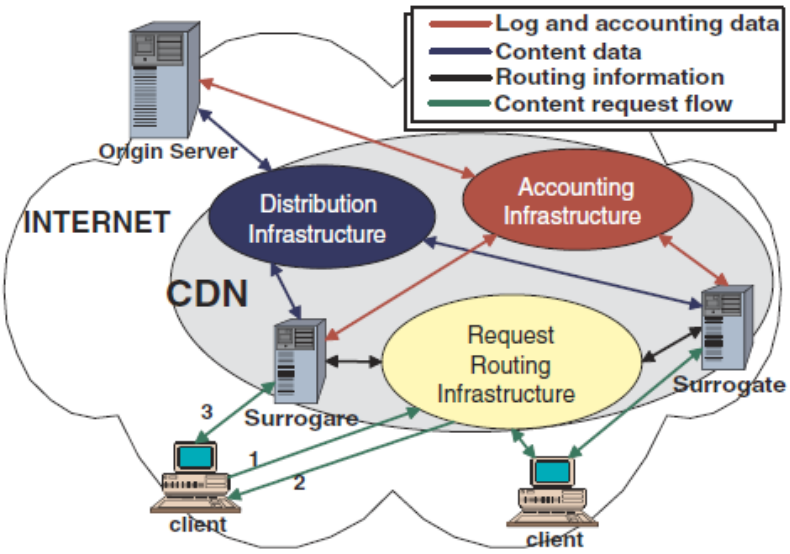


Figura 12. Infrastructure components CDN. (2004) (Fuente: N. Bartolini, et al)

2.2.2 Composición de CDN

Hace referencia a los aspectos de estructuración, organización y arquitectura, clasificando las CDNs en base a sus atributos estructurales (Pathan & Buyya, 2009). Aquí se incluyen a los *origin server* y un conjunto de *replica servers*, que son los que distribuyen o entregan copias del contenido a los usuarios finales.

Comúnmente CDN incluye información dinámica sobre las condiciones de la red y la carga sobre los *cache servers*, para redirigir las solicitudes y balancear las cargas entre los *surrogates servers*. Dependiendo del tipo de contenido o servicio que CDN provea, es como variará su estructura, es por esto que principalmente se empleará un conjunto de *surrogates* para formar la estructura *content-delivery* como tal y para esto se emplearán ciertos mecanismos para encaminar los requerimientos del cliente a los *surrogates* y protocolos de interacción entre los elementos que conforman todo el sistema CDN (Pathan & Buyya, 2009).

La composición de CDN tiene los siguientes parámetros dentro de su estructura (fig. 13):

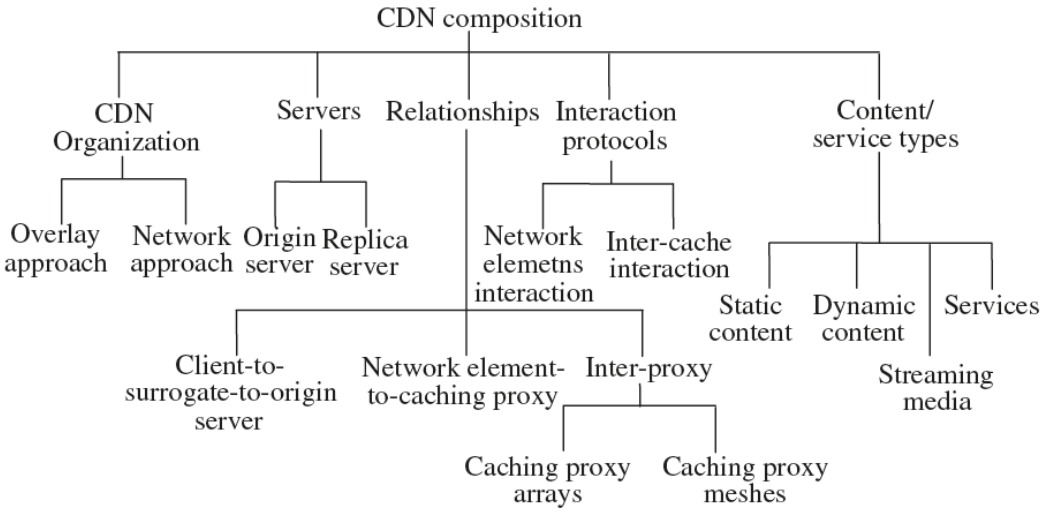


Figura 13. CDN composition taxonomy. (2008) (Fuente: R. Buyya)

2.2.2.1 Organización de CDN

Existen 2 enfoques generales en la constitución de CDN, que son: *overlay* y *network*. La primera es mediante una red *overlay network* (o red de capa de aplicación), que corresponde al encaminamiento global y está formada por un conjunto de servidores y servidores caché que atienden las peticiones y distribución de ciertos tipos de contenido (web, *media streaming*, vídeo en tiempo real) (Buyya et al, 2008). En este esquema los componentes de conectividad básica de red como *routers* y *switches*, que no desempeñan ningún papel activo en la distribución de contenido, sino simplemente proporcionar conectividad y en ciertos casos un cierto nivel de QoS para tráfico de requerimiento específico. La gran mayoría de proveedores comerciales emplean este esquema *overlay* en la organización de la estructura de su CDN.

Como segundo enfoque está el de *network*, que corresponde a un encaminamiento local, donde los componentes de la red, incluyendo *routers* y *switches* están equipados con código para identificar los tipos de aplicaciones específicas y poder transmitir las solicitudes basadas en políticas predefinidas. Los ejemplos de este enfoque incluyen dispositivos que redirigen solicitudes de contenido a cachés locales o tráfico de conmutación que vienen a los

centros de datos a servidores específicos optimizados para servir tipos de contenido específico. En este caso existen algunos proveedores de CDN como Akamai que utiliza los dos tipos de enfoque para la organización (Pathan & Buyya, 2009).

2.2.2.2 Servidores

En esta sección se describen 2 tipos de servidores: *origin server* (o el servidor origen) y *replica servers* (o los réplicas, conocidos también como *surrogates*).

Los *origin servers* son aquellos que albergan el contenido que fue introducido o actualizado por el proveedor de contenido y para esto, está en constante comunicación con los *distributed replica servers*. Mientras que el *surrogate* operaría como un *media server* (respondiendo a las consultas de audio y video específicas), o un *Web server* (el cual contiene los enlaces para la transmisión del medio o contenido específico de CDN), finalmente está como un *cache serve*r (Pathan & Buyya, 2009). (el cual realiza réplicas del contenido en el borde de la red con el fin de evitar la necesidad de tener que estar accediendo a los *origin server* para satisfacer cualquier solicitud de contenido realizada, manteniendo una réplica total o parcial del contenido que guarda el *origin server*.

2.2.2.3 Relaciones

Dentro de CDN, para comprender este tipo de relaciones entre equipos, sería entender el comportamiento del entorno entre un *replication server* y un *caching server* dentro del sistema CDN, es por esto que al hablar de lo compleja y amplia arquitectura que abarca, implicaría varias relaciones entre todos sus componentes, como: clientes, *surrogates, origin server, proxy caches* y otros elementos de red. La mayoría de estos componentes se comunican entre sí para replicar o hacer *caching* de contenido dentro del ambiente CDN (Pathan & Buyya, 2009).

El proceso de replicación, entra en operación cuando se realiza un "*pushing*" de contenido desde el *origin server* hacia los *surrogates*. Por otro lado, el proceso de *caching* implica almacenar respuestas para futuras peticiones. Cada cliente dentro de la CDN, si la red emplea *surrogates*, la comunicación la hace directa y transparentemente a estos servidores, en lugar de que uno o más *origin servers* atiendan las peticiones (a).

CDN puede conformarse utilizando un enfoque desde el punto de vista de la red, por ejemplo desde la lógica operacional de un *router o switch*, reenviando el tráfico a los *servers o proxies* especializados dentro del sistema y que tienen la capacidad de servir las peticiones que hagan los usuarios, formando en este caso la relación entre los usuarios del sistema, elemento de la red y cacheo de los equipos *servers o proxies* o conjuntos de los mismos formando *proxy arrays* (b) (Pathan & Buyya, 2009), así como la comunicación entre *caching proxies*, que es un servicio de red en la capa de aplicación con capacidad para almacenar en caché los objetos Web (fig. 14). Los *CDN proxy caches* entregan el contenido sólo para ciertos proveedores de contenido, conocidos como los *CDN customers*.

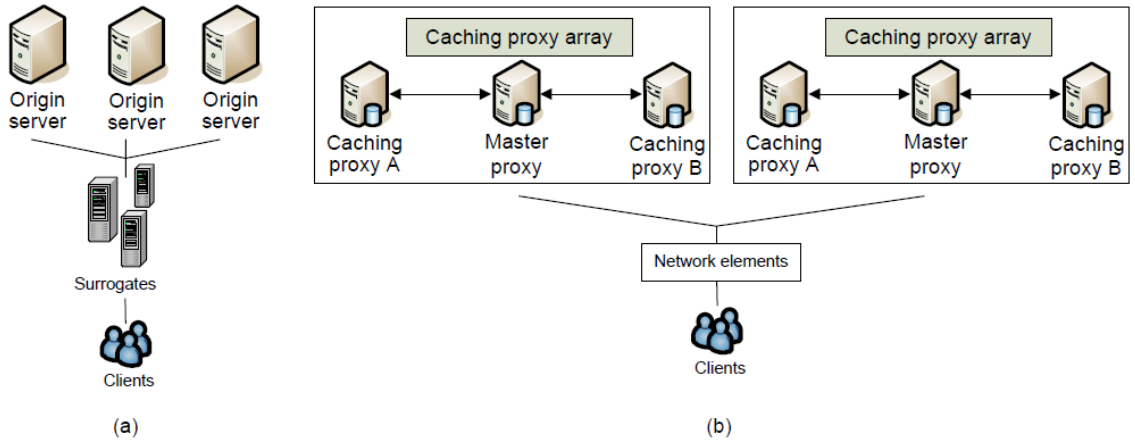


Figura 14. (a) Client-to-surrogate-to-origin server; (b) Network element-to-caching proxy (2009) (Fuente: A. Pathan y R. Buyya)

Basándose en la comunicación *inter-proxy*, los *caching proxies* pueden conformar *proxy arrays* (clúster) (c) y *proxy meshes* (d), donde un *proxy* autoritativo actuará como

equipo maestro en la comunicación con otros *caching proxies* (fig. 15). A su vez, la comunicación puede tener lugar al mismo nivel (*peers o siblings*) o a un nivel superior (*parents*) (Pathan & Buyya, 2009).

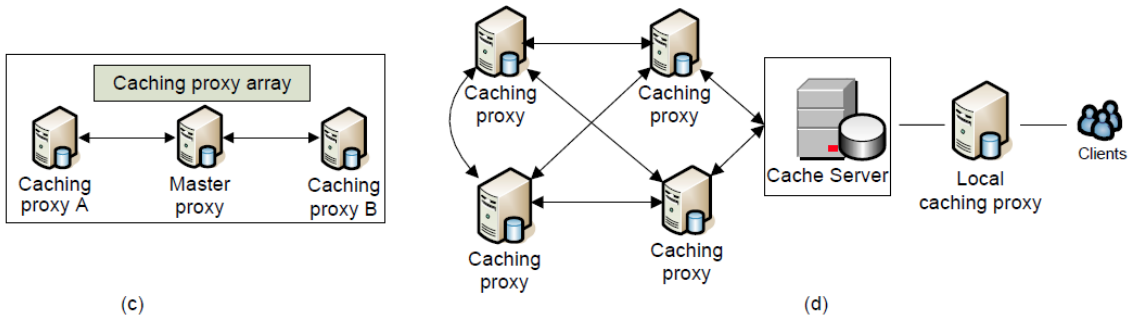


Figura 15. (c) *Caching proxy arrays*, (d) *Caching proxy meshes* (2009) (Fuente: A. Pathan y R. Buyya)

2.2.2.4 Protocolos de Interacción

CDN puede utilizar varios tipos de protocolos de interacción: entre los elementos de la red y entre los cachés. Dentro del tipo de protocolos para la interacción entre los elementos de red están: El *Network Element Control Protocol* (NECP), *Web Cache Coordination Protocol* y *SOCKS*. Por otro lado, como ejemplo de los protocolos de interacción *inter-cache*, están el *Cache Array Routing Protocol* (CARP), el *Internet Cache Protocol* (ICP), así como el *Hypertext Caching protocol* (HTCP) y el *Cache Digest*. Los cuales se describen a continuación:

- **NECP**

Network Element Control Protocol (NECP), es un protocolo ligero empleado para la señalización entre los servidores y los elementos de red (que son *content-aware switches* and

load-balancing routers. Este protocolo permite a los elementos de red realizar balanceo de carga a través de las granjas de servidores y la redirección a los *interception proxies*. NECP proporciona métodos a los elementos de red para aprender acerca de las capacidades del servidor, disponibilidad y sugerencias en cuanto a que los flujos puedan o no ser atendidos (Cieslak et al, 2013). NECP se utiliza en una amplia variedad de aplicaciones de servidor, incluso para los *origin servers, proxies e interception proxies*, por lo tanto emplea TCP como protocolo de transporte. Al emplear este modelo de protocolo, los mensajes *Keepalive* a nivel de aplicación se utilizan para detectar cuando un *peer* ha colapsado.

- **WCCP**

Cache Coordination Protocol (WCCP), este protocolo especifica la interacción entre un router, como un elemento de red de redireccionamiento y equipo para *interception proxies*; como también se tiene la interacción con un grupo de routers para establecer y mantener la redirección transparente de determinados tipos de flujo de tráfico. El siguiente grupo de relación es hacia uno o más *Web-caches*, donde el tráfico seleccionado es redirigido a un grupo de *Webcaches* con el fin de aumentar la utilización de recursos y para reducir al mínimo el tiempo de respuesta (Cieslak et al. 2013).

- **SOCKS**

Este protocolo es conceptualmente un *"shim-layer"* entre la capa de aplicación y la capa de transporte, debido a que está diseñado para proporcionar seguridad a través de un túnel autenticado al utilizar los servicios entre un *firewall* y un *caching proxy*; entregando una estructura dentro de las aplicaciones *client-server* para TCP y UDP (Leech et al, 1996). Cuando un sistema o un cliente basado en TCP, trata de conectarse a un contenido que puede ser cacheable pero que está protegido por un *firewall,* tiene que abrir una conexión TCP al puerto SOCKS dentro del servidor SOCKS, empleando las rutinas de encapsulación adecuados registrados en la biblioteca SOCKS y así establecer la conexión o ser rechazado.

- **CARP**

Cache Array Routing Protocol (CARP) es un protocolo de almacenamiento en caché distribuido, se basa en una lista conocida de *proxy servers* débilmente acoplados y una función *hash* para dividir el espacio URL entre esos proxies. Un cliente HTTP que implemente CARP, puede rutear peticiones a cualquier miembro del *Proxy Array*, cuya tabla está definida en un archivo ASCII en texto plano (Valloppillil & Ross, 1998). Para conocer quién es el contenedor de los recursos almacenados, la función *hash* y el algoritmo de encaminamiento de CARP toman a unos de los miembros más apropiados definidos en la tabla *proxy array*, organizando el sistema de tal manera que la duplicación del contenido en caché es eliminada y las tasas de almacenamiento en cache mejoran considerablemente.

- **ICP**

Internet Cache Protocol (ICP), es un formato de mensaje ligero utilizado para la comunicación *inter-cache*. Los equipos caches intercambian mensajes ICP para localizar y seleccionar de la ubicación más adecuada y así recuperar un objeto desde un repositorio. Se lo utiliza en un entorno *caching proxy mesh* para localizar objetos Web específicos en equipos caches vecinos. Por lo general, la PIC se implementa en la parte superior de la UDP (Wessels, 1997) con el fin de proporcionar características importantes para las aplicaciones de almacenamiento en caché Web y podría contar con características de balanceo de carga entre el resto de equipos caches.

- **HTCP**

Hypertext Caching Protocol (HTCP), es un protocolo desarrollado para descubrir equipos HTTP caches, datos almacenados en caché, es compatible con HTTP 1.0, sirve para monitorear y gestionar sus actividades, como permitir a las cabeceras que se incluyan en las solicitud. HTCP permite peticiones y respuesta completas de cabeceras y que serán utilizadas en la gestión del caché, haciendo que los agentes HTCP no deban ser aislados de fallo en la

red o retardos, tratando de mantenerse activos. Los Mensajes HTCP pueden ser enviados a través de UDP o TCP (Vixie & Wessels, 2000).

- ***Cache Digest***

Es un protocolo de intercambio y formato de datos. Su funcionamiento puede operar de tal manera que puede eliminar la redundancia y mejorar el uso de los *Internet server* y de todos los recursos de ancho de banda del sistema. El *cache digests* proporciona una solución a los problemas de tiempo de respuesta y congestión asociados con otros protocolos de comunicación *inter-cache* como ICP y HTCP (Hamilton et al, 1998). Cuando se lo emplea es posible determinar con precisión si un servidor concreto almacena en caché a una URL determinada, haciendo que el protocolo como tal pueda ser intercambiado por FTP y anunciar resúmenes indicando cuales URLs son almacenados por un determinado servidor.

2.2.2.5 Tipos de Contenido/Servicio

Una CDN se enfoca principalmente en forjar su infraestructura de red para servir los siguientes servicios y funcionalidades: Almacenamiento y administración del contenido, distribución del contenido entre los *edge servers*, administración de la memoria en cache, distribución de contenido estático, dinámico y de *streaming*, entrega también soluciones de contingencia para la recuperación de contenido ante perdidas por desastres, así como administración de temas de contabilidad para clientes, como monitoreo, estadísticas de rendimiento del servicio, reportes de uso y todo este contenido puede ser distribuido y alojado a través de la red de *cache servers* de la CDN (fig. 16) (Buyya et al, 2008).

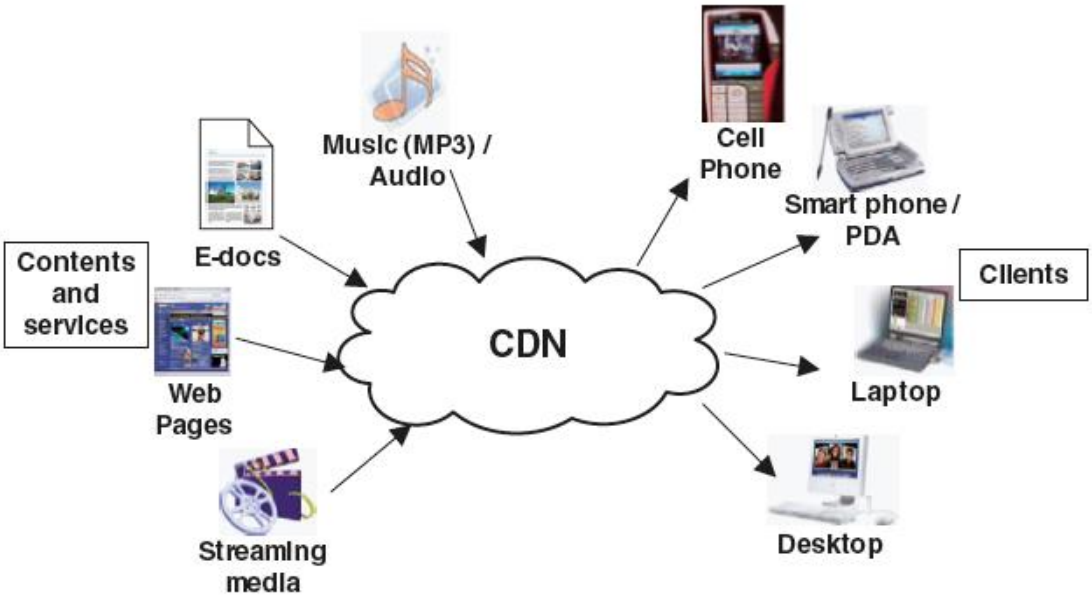


Figura 16. Contenidos y servicios de CDN. (2008) (Fuente: Buyya)

Ya para la práctica, se podría indicar los siguientes tipos de elementos por cada uno de los servicios que se puede proveer en CDN:

- Servicios de contenido: pueden ser los servicios de directorio, servicios de *e-commerce* o servicios de transferencia de archivos.

- Distribución de contenido desde un origen: este servicio puede ser dedicado para grandes empresas y exigencia de distribución del contenido, proveedores de servicios Web, compañías de medios de información y transmisores de noticias.

- Los clientes y consumidores típicos de un servicio CDN, son aquellas compañías que emplean medios de comunicación o empresas de publicidad sobre Internet, *Data Centers*, proveedores de servicios de Internet (ISP), tiendas de música en línea, operadores móviles o de electrónica de consumo.

- **Servicios Basados en Web Estático**

Cuando se habla de contenido estático, este tipo de servicio es utilizado para acceder a contenido estático como (páginas HTML estáticas, imágenes, documentos, parches de software, audio y/o archivos de vídeo), así como contenidos que cambian con poca frecuencia o dentro de un tiempo definido como (páginas web volátiles, Bolsa de cotización). Dentro de esta categoría se podría encontrar a casi todos los proveedores de CDN, los cuales apoyan este tipo de entrega de contenidos, ya que son fácilmente almacenado en caché y la frescura de su contenido mantiene a los equipos de borde utilizando tecnologías tradicionales de almacenamiento en caché (Bartolini et al, 2004).

- **Servicios de Almacenamiento Web**

Este servicio puede basarse en las mismas técnicas utilizadas para la entrega de contenido estático. Se deben agregar características adicionales para gestionar el registro y la transferencia segura de archivos. Dentro de este tipo de aplicaciones, se puede requerir el procesamiento en el lugar de origen (*origin server*) o en el borde (*edge server*) (Bartolini et al, 2004).

- **Servicios para Transferencia de Archivos**

Este servicio trata sobre la distribución mundial de software como por ejemplo (parches para sistemas operativos, bases de datos para la definición de virus, etc.), material de estudio (*e-learning*) que las empresas ofrecen a todos sus empleados a nivel mundial, películas bajo pedido a grandes compañía en el medio, imágenes médicas de alta definición que son compartidas entre médicos y hospitales, etc. Este tipo de contenido es esencialmente estático y puede utilizando las mismas técnicas adoptadas para los servicios *static web based* (Bartolini et al, 2004).

- **Servicios para E-commerce**

La semántica de la consulta utilizada en la navegación de un catálogo de productos no es compleja, por lo que para resultados más frecuentes se pueden utilizar técnicas de almacenamiento en caché. Ciertas negociaciones comerciales pueden ser almacenadas y mantenidas en los *replica server*, así como los pedidos y transacciones de las tarjetas de crédito pueden ser procesadas en los equipos de borde (*edge*), esto requiere que se cuente con *replica servers* de confianza y que estén aptos para realizar transacciones seguras. Algunos autores proponen que para manejar sitios donde se realice e-commerce, se emplee el almacenamiento dinámico del contenidos (Bartolini et al, 2004).

- **Servicios para Manejo de Directorios**

Se utiliza para acceder a los servidores de bases de datos. Por ejemplo, en el caso de un servidor *Lightweight Directory Access Protocol* (LDAP), los resultados de las consultas frecuentes o un subconjunto de directorios pueden almacenarse en el caché de los equipos de borde (*edge*) (Bartolini et al, 2004).

- **Aplicación Web**

Una página web dinámica utiliza transacciones Web, procesamiento de datos, acceso a base de datos, calendarios, horarios de trabajo, etc. Por lo que dentro de este servicio, para utilizar una infraestructura CDN se debería llenar los *replica servers* con el contenido que más frecuentemente componen la estructura dinámica de una página web y manteniendo las aplicaciones y su procesamiento en el *origin server*. Otro enfoque sería el de replicar la aplicación tanto parcial o total y el contenido en el *origin server*, de esta manera todo el proceso de generación de contenidos (lógica de la aplicación y recuperación de contenidos) serían manejados por el *replica server* liberando de carga al *origin server* (Bartolini et al, 2004).

- **Transmisión en Vivo o Bajo Demanda *(Streaming)***

En este caso el *edge server* debe tener la capacidad de soportar *streaming* (Bartolini et al, 2004). Están los archivos de audio o el video en tiempo real, repositorios de videos como *User Generated Videos* (UGV). Este tipo de servicio se revisará en detalle durante el capítulo de Video.

2.2.3 Distribución y Gestión de Contenidos

Son aquellos componentes que se encargan de mover el contenido desde el *origin server* hacia los *CDN edge servers* y asegurar la consistencia de información del contenido en los *cache servers*. Aquí se describen los alcances de CDN para la distribución y el manejo del contenido, en términos de la ubicación de servidores (*surrogate placement*), selección y distribución de contenido, externalización de contenido y organización de equipos cachés y réplicas.

La distribución y la administración del contenido son muy importantes para todo el desempeño del sistema CDN, ya que implica la reducción del tiempo de respuesta que percibe un usuario. La distribución del contenido incluye: técnicas basadas en el posicionamiento estratégico para que un *edge server* este lo más cerca al usuario y para esto se emplea el *placement of surrogates*; se tiene también el *content selection y delivery*, que se basa en el comportamiento del tipo y frecuencia con que se realizan los requerimiento del usuario, y finalmente se tiene el método de *content outsourcing,* que es utilizado para decidir qué tipo de metodología de externalización se seguirá (Buyya et al, 2008) (fig. 17). La gestión de contenido depende en gran parte de las técnicas conocidas como *cache organization*, las cuales incluye: *caching techniques, cache maintenance y cache update* (CDN Market, 2005).

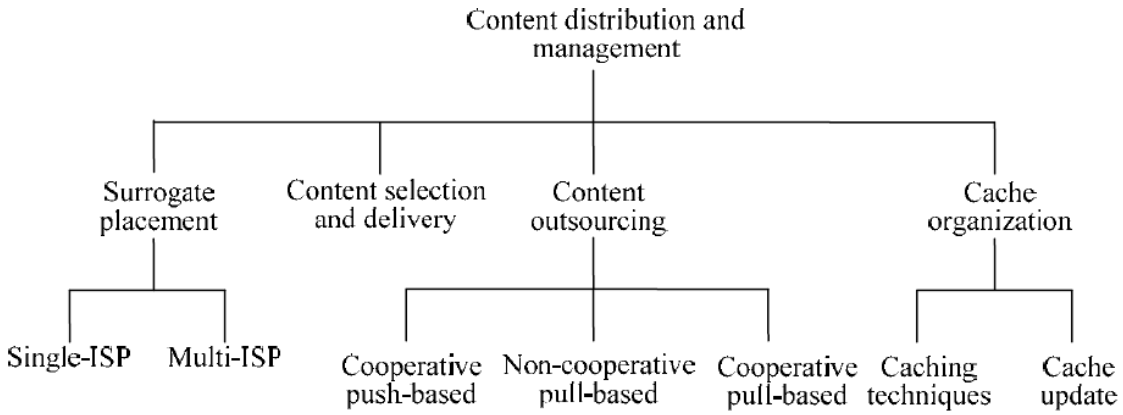


Figura 17. Content distribution and management taxonomy (2010) (Fuente: A. Pathan y R. Buyya)

2.2.3.1 Ubicación o Colocación de Surrogates

La correcta ubicación de un servidor *surrogate*, está estrechamente relacionado con el proceso único dentro del contexto de realizar una distribución del contenido; siempre tratando de conseguir la mejor ubicación del *surrogate*, para reducir notoriamente la latencia percibida por el usuarios al momento de estar en el proceso de adquisición de un contenido, así como disminuir efectivamente al máximo el ancho de banda de una red en el proceso de la transferencia del contenido replicado (Pathan & Buyya, 2009). El principal beneficio será el de conseguir el mayor rendimiento dentro del negocio de CDN, lo que a la final representará que se ofrezcan servicios CDN de calidad y al menor precio. Y es este proceso detallado de ubicación donde nos enfocaremos principalmente al hablar de los beneficios de la integración CDN/Cloud.

Este mecanismo se utiliza para decidir la correcta ubicación de los servidores de réplica y para que adaptativamente sean llenados con el contenido apropiado antes de que les llegue la petición (*pre-fetching*), durante el cambio en la condición de tráfico (Bartolini et al, 2004). Es así que, estos servidores no completan su capacidad una vez que les llega la petición como lo hacen tradicionalmente los *proxy caching*, sino que son actualizados de forma

proactiva, provocando una sola sobrecarga al momento de la descarga, la cual no se repite nuevamente en cada acceso al *origin server*.

Las estrategias teóricas, como el problema de la métrica k mínima *(minimum k-center problem*) o los k- HST (*k-Hierarchically well-Separated Trees*) (Bartal, 1996) modelan el problema de la ubicación de servidores como un problema de ubicación centrada (*center placement problem*) que se define de la siguiente forma: para la ubicación de un número concreto de servidores centrados, minimizar la máxima distancia entre un nodo y el centro servidor más cercano. El algoritmo k-HST (Jamin et al, 2000) resuelve el problema empleando teoría de grafos. En este método, la red se representa como un grafo G (V,E) donde V es el conjunto de nodos y E ⊆ V x V es el conjunto de enlaces. El algoritmo tiene dos fases (fig. 18).

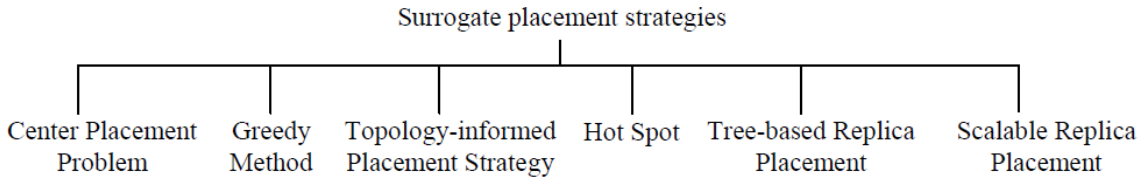


Figura 18. Surrogate placement strategies (2009) (Fuente: A. Pathan y R. Buyya)

En la primera fase, se selecciona un nodo arbitrario del grafo completo llamada partición padre (*parent partition*) y todos los nodos que están dentro de un radio aleatorio desde ese nodo arbitrario forman una nueva partición llamada partición hija (*child partition*) (Molina, 2013). El radio de la partición hija es k veces más pequeño que el diámetro de la partición padre. Este proceso continúa hasta que cada uno de los nodos se encuentre en una partición propia. De esta forma, el grafo se divide de forma recursiva y se obtiene un árbol de particiones siendo el nodo raíz la red entera, mientras que los nodos hoja serían los nodos individuales dentro de la red (Pathan & Buyya, 2009). En la segunda fase, se asigna un nodo virtual a cada una de las particiones en todos los niveles. Cada nodo virtual en una partición padre desempeña la función de 'padre' de los nodos virtuales en las particiones hija y juntos los nodos virtuales también forman un árbol. A continuación, se aplica una estrategia de tipo

voraz (*greedy*) para encontrar el número de nodos centralizados necesarios para el árbol k-HST obtenido cuando la distancia máxima entre un centro y un nodo está acotada por un valor D. Por otro lado, la segunda estrategia teórica, asociada al problema de la métrica k-mínima se plantea como un problema basado en grafos bi-dimensionales (Jamin et al, 2000).

Debido a la complejidad de los algoritmos teóricos y a la carga computacional que ello implica, se han introducido métodos heurísticos, como los métodos (*greedy replica placement*) y topológicos (*topology-informed placement*) (Krishnan et al, 2000). Estos algoritmos sub-óptimos tienen en cuenta la información disponible en la CDN, como los patrones de carga y la topología de la red, proporcionando una solución adecuada con una carga computacional reducida. Por un lado, el algoritmo *greedy* (Buyya et al, 2008) selecciona M servidores entre N ubicaciones (*sites*) disponibles. En la primera iteración, se calcula el coste asociado con cada ubicación, teniendo en cuenta el acceso de todos los clientes a dicha ubicación. En la segunda iteración, el algoritmo *greedy* busca una segunda ubicación (la siguiente con menor coste) en conjunción con la primera ubicación seleccionada. La iteración continúa hasta que se hayan seleccionado M servidores. El algoritmo *greedy* es efectivo incluso con datos imprecisos, pero requiere el conocimiento de las ubicaciones de los clientes en la red y todas las distancias entre nodos (Pathan & Buyya, 2009).

En el método topológico (Jamin et al, 2001), los servidores se ubican en hosts candidatos en orden descendiente de grado (el grado viene determinado por el número de enlaces que conectan un nodo con otro). En este método se asume que los nodos de mayor grado pueden alcanzar otros nodos con una menor latencia. El algoritmo emplea las topologías de los sistemas autónomos (*AS, Autonomous System*) donde cada nodo representa un AS y un enlace corresponde con una relación de *peering* en el protocolo BGP (*Border Gateway Protocol*). Si se desea mejorar la estrategia basada en la topología (Radoslavov et al, 2001), se puede emplear la topología a nivel de router en lugar de la topología a nivel de AS; en este caso, cada red de área local (*LAN, Local Area Network*) asociada con un *router* corresponde con una ubicación potencial para colocar un servidor, en lugar de que cada AS se convierta en un *site*.

Otros algoritmos de ubicación de servidores son *Hot Spot* (Qiu et al, 2001) y los basados en árbol (*Tree-based*) (Li et al, 1999). El algoritmo *Hot spot* ubica a los servidores réplica (*surrogates*) cerca de los clientes que generan mayor carga de tráfico. Básicamente, lista las N ubicaciones potenciales en base a la cantidad de tráfico generada por los clientes, y luego ubica los *surrogates* en las M ubicaciones que generan mayor tráfico. El algoritmo de ubicación de los *surrogates* basado en árbol (*Tree-based*) asume que la topología subyacente son árboles, y modela la estrategia de ubicación de los replica server como un problema de programación dinámico. De esta forma, un árbol T se divide en varios árboles *Ti* y la ubicación de *t* proxies se consigue al colocar *t´i proxies* de la mejor manera en cada árbol *Ti*, siendo $t = \sum_i t'_i$.

En cuanto a los métodos escalables en la ubicación de servidores, un ejemplo lo constituye SCAN (*Scalable Content Access Network*) (Chen et al, 2002), que es un marco de gestión de réplicas (*surrogates*) escalable que genera réplicas bajo demanda y las organiza dentro de un árbol *multicast* de nivel de aplicación. Esta aproximación minimiza el número de réplicas a la vez que se cumplen los requisitos de latencia de los clientes y los requisitos de capacidad de los servidores. Una vez vistos los algoritmos o estrategias de ubicación de servidores, y retomando la Figura 19, los administradores de CDNs también determinan el número óptimo de *surrogates* empleando una aproximación *mono-ISP* o *multi-ISP* (Vakali & Pallis, 2003). En el caso *mono-ISP*, el proveedor de CDN típicamente despliega al menos 40 *surrogates* en el extremo de la red del ISP para dar soporte a la distribución de contenido (Douglis & Kaashoek, 2001). La política empleada en este caso es ubicar uno o dos *surrogates* en cada ciudad ampliamente poblada donde entregue servicio dicho ISP. La desventaja de este método es que los *surrogates* pueden estar ubicados lejos de los clientes a los que el proveedor de CDN desee dar servicio.

En el método *multi-ISP*, el proveedor de CDN ubica varios *surrogates* en el mayor número de puntos de presencia (PoP, *Point of Presence*) (Akamai Technologies, 2007) de ISPs como sea posible, con la finalidad de estar lo más cerca posible de los clientes y mantener el contenido confiable. Las grandes CDNs comerciales como Akamai disponen de más de 25000 servidores repartidos por todo el mundo de esta forma (Dilley et al, 2002). Más que el coste y la complejidad en la configuración, el mayor inconveniente de este método

multi-ISP es que cada *surrogate* pueda recibir un número reducido de peticiones (o incluso ninguna petición), lo que conlleva a una infrautilización de la infraestructura y a una posible baja valoración del rendimiento de la CDN en su conjunto (Pallis & Vakali, 2006). Algunas estimaciones de rendimiento indican que el método *mono-ISP* funciona mejor para tráficos de red bajos o moderados, mientras que el método *multi-ISP* funciona mejor para tráficos de red elevados, por lo que son adecuados para sitios web con gran carga.

2.2.3.2 Selección de Contenido y Distribución

Durante la selección y distribución total (*Full-site*), los *surrogates* replican todo el contenido de los *origin server*. La eficiencia en la entrega del contenido recae sobre la correcta selección del contenido a ser entregado a los usuarios y puede ayudar en la reducción del tiempo de descarga del cliente y evitar una excesiva carga en el servidor. Este mecanismo debe de agregarse a los *routers* de acceso cooperativo, para localizar con precisión el mejor y más cercano *edge server* disponible sobre el cual los usuarios finales pueden recuperar el contenido requerido (Bartolini et al, 2004). A más de esto, debe mantenerse un servicio robusto para evitar que los servidores se sobrecarguen por medio del control de acceso y el balanceo de carga. El servicio de negociación sirve para cubrir los requerimientos concretos de usuarios específicos (o grupos de usuarios), mientras que, el servicio de distribución sirve para replicar o cachear contenido desde un *origin server* a los *surrogates* dispersos en la Internet (Buyya et al, 2008).

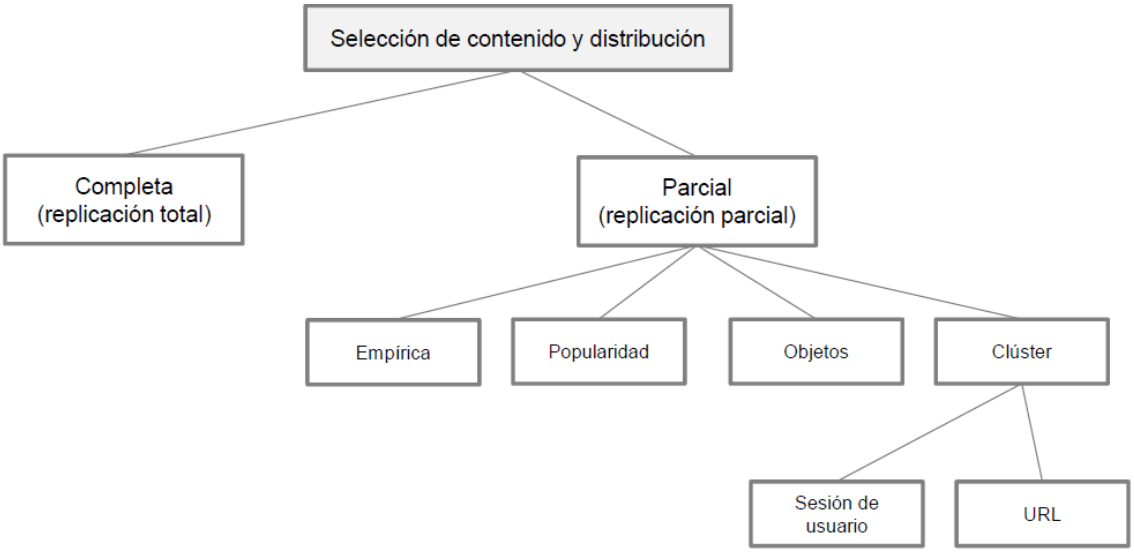


Figura 19. Selección y distribución de contenido (2010) (Fuente: R. Buyya)

Para que *full-site* funcione, el proveedor de contenido configura su servicio DNS de tal forma que todas las peticiones son resueltas por un servidor de la CDN o *surrogate*, quien distribuye el contenido, ganando algo de sencillez con este esquema. Sin embargo, dentro del diseño no representaría una solución práctica, ya que se debe considerar el crecimiento dinámico de Internet y la enorme colección que se puede formar de los objetos web.

Durante la selección y distribución parcial (*partial-site*), los *surrogates* realizan una especie de replicación parcial para distribuir únicamente objetos embebidos como imágenes desde la CDN, de esta manera el cambio del contenido embebido es menos frecuente, por lo que este método resulta tener un mejor rendimiento (fig. 19). Este a su vez se subdivide en modalidad empírica, de objetos y basada en popularidad del contenido o *cluster*. De esta forma, un proveedor de contenido modifica su contenido de tal forma que los enlaces a objetos específicos embebidos referencian nombres de host de los cuales el proveedor de CDN es autoritativo. Pero también se requiere de una estrategia de gestión adecuada para replicar dicho contenido web. Así, la página web HTML básica se obtiene del *origin server*, mientras que los objetos embebidos se obtienen de los *proxy caches* o servidores cache de CDN (Molina, 2013). Siendo la más compleja de las modalidades la de *Cluster*, que se realiza o bien fijando el número de clústeres o fijando el diámetro máximo del clúster, lo que permite

reducir el tiempo de descarga y la carga en el servidor, ya que los objetos web más populares son replicados en unidades de clústeres donde la distancia de correlación entre cada par de URLs está basada en una métrica de correlación determinada. Por otro lado, el *clústering* de contenido puede estar basado en la sesión de usuario o en la URL (Buyya et al, 2008).

2.2.3.3 Externalización de Contenido

La externalización de contenido (*content outsourcing*), se deriva una vez que se ha seleccionado correctamente el posicionamiento de los *surrogates serves* y se ha seleccionado el contenido a ser distribuido dentro de la infraestructura CDN, haciendo referencia al mecanismo de distribución desde el *origin server* hacia los *surrogates* (Papagianni et al, 2013). Existen tres mecanismos de distribución:

- **Push Cooperativo**

De primera instancia, en este sistema el contenido se envía desde el *origin server* hacia los *surrogates* y posteriormente, dichos *surrogates* cooperarán entre sí para reducir costos de replicación y actualización (Kangasharju et al, 2002). Este mecanismo de CDN mantiene un esquema tipo mapeo entre el contenido y los *surrogate servers*, donde cada petición de clientes es direccionada al *surrogate server* más cercano o en último recurso la petición se enviará al *origin server*. Desde esta perspectiva, resulta adecuado emplear un algoritmo heurístico de tipo *greedy-global* debido a que se ajusta para realizar las decisiones de replicación entre los *surrogates* cooperativos (Chen et al, 2003). En tal caso se trata de un esquema teórico puesto que no es empleado por ninguna CDN comercial.

- **Pull no Cooperativo**

Durante este enfoque, los requerimientos de clientes son dirigidos hacia sus *surrogates* más cercanos (ya sea mediante la redirección de DNS o la reescritura de URL) (Fujita et al, 2004). Si se produce un fallo (*cache miss*), los *surrogates server* obtienen el contenido del *origin server*. Los proveedores de CDNs comerciales más populares (*Akamai*, *Mirror Image*) emplean este mecanismo. El principal inconveniente de este esquema es que no siempre se toma un *surrogate* óptimo para servir contenido (Johnson et al, 2001). Sin embargo, la mayoría de CDNs acertadamente emplean esta solución mientras que el esquema *push* cooperativo permanezca en etapa experimental (Katsaros et al, 2009).

- **Pull Cooperativo**

En este esquema, los requerimientos de cliente son direccionados a los *surrogates* más cercanos a través de la redirección de DNS y se diferencia con el esquema *pull* no cooperativo, en que los *surrogate servers* cooperan entre ellos para obtener el contenido solicitado, en caso de presentarse un fallo (*cache miss*) (Freedman, 2004). Mediante el uso de un índice distribuido, los *surrogates* son capaces de localizar copias de contenido solicitado en *surrogates* cercanos y almacenarlos en caché. A este esquema en particular se lo considera reactivo en base a que desde el punto de vista, un objeto es cacheado sólo cuando un cliente lo solicita. Como ejemplo, la CDN académica Coral, emplea este esquema de externalización de contenido, utilizando una variación del *Distribution Hash Table* (DHT).

Dentro del enfoque de la externalización de contenido, resulta extremadamente importante dentro de los problemas que se puedan presentar en la distribución de contenido, el determinar en cuales *surrogate servers* se debe replicar qué contenido. Se pueden encontrar varios trabajos dentro de la literatura científica que demuestran la efectividad de diversas estrategias de replicación del contenido externalizado. Como por ejemplo, en la literatura de *Kangasharju* et al (2002) se describen cuatro métodos heurísticos, como son aleatorio (*random*), basado en popularidad (*popularity*), *greedy-single* y *greedy-global*. Mientras que el autor *Tse* (2005) presenta otras estrategias de tipo *greedy* donde el contenido se distribuye

balanceando la carga y el tamaño de la capacidad de los *surrogates*. Por otra parte *Pallis* et al. (2005) describen un algoritmo autoajustable sin parámetros (de entrada) denominado *lat-cdn* para ubicar de forma óptima el contenido en una CDN, este algoritmo emplea la latencia de los objetos para tomar las decisiones de replicación. La latencia de un objeto se define como el retardo experimentado desde que se solicita un objeto hasta que se obtiene completamente. Una mejora sobre este algoritmo la constituye *il2p* (Pallis et al, 2006), que ubica el contenido en los *surrogates* dependiendo de la latencia y la carga de los objetos.

2.2.3.4 Organización de Caché

La organización de Caché está compuesto por las técnicas de almacenamiento en caché (*caching*) utilizados y la frecuencia de actualización, para asegurar la frescura, la disponibilidad y la fiabilidad de los contenidos (Stamos et al, 2006). Una adecuada gestión del contenido es fundamental para el buen rendimiento y está principalmente basado en el esquema de organización de caché empleado dentro de una CDN. Además, es posible el uso de forma integrada en la infraestructura de la CDN, a través de políticas de *caching* y replicación con la finalidad de introducir mejoras potenciales en la latencia percibida, *hit ratio* y *byte hit ratio*. Es más, el uso conjunto de ambas técnicas permite aumentar la robustez del sistema frente a eventos de tipo *flash-crowd*. Para esto, en la autoría de *Stamos* et al (2006) se describe un método heurístico no paramétrico que integra ambas técnicas de *Web caching* con *content replication* y evalúa el nivel de integración, a lo cual se lo conoce como *Similarity Replication Caching* (SRC) (fig. 20). Otro ejemplo, denominado *Hybrid* (Bakiras & Loukopoulos, 2005), el cual combina *static replication* con *web caching*, usando un modelo analítico basado en LRU. La integración híbrida llena gradualmente los cachés de los *surrogate servers* con contenido estático en cada interacción, tanto tiempo como contribuya a la optimización de los tiempos de respuesta.

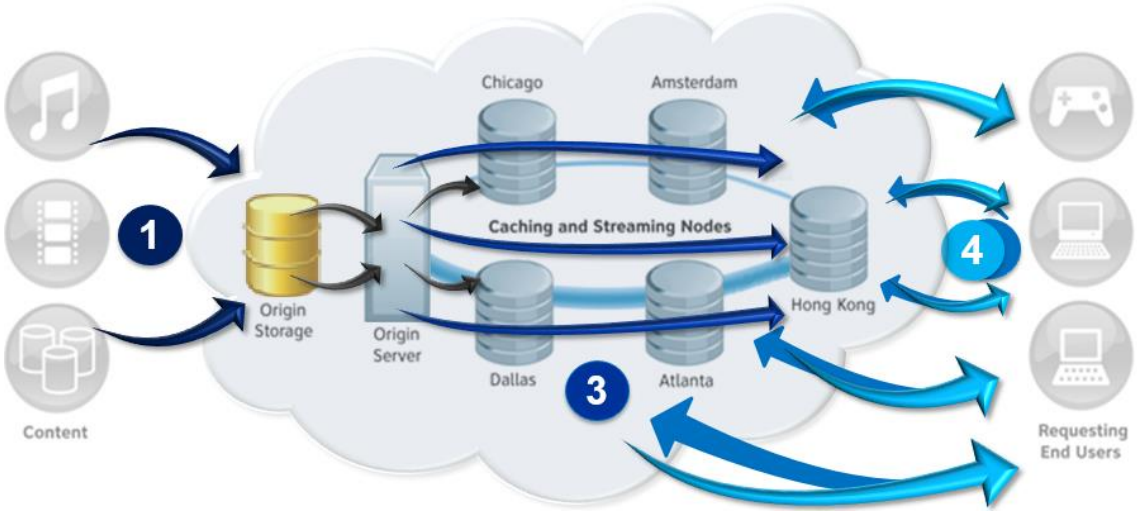


Figura 20. Origin Storage Service Details (2014) (Fuente: Level3.com)

2.2.3.4.1 Técnicas de Almacenamiento en Caché (*Caching*).

La replicación del contenido es un fenómeno común y ampliamente aceptado a gran escala en entornos de distribución como CDN, donde el contenido se almacena en más de un lugar por razones de rendimiento y fiabilidad. Diferentes estrategias de replicación son adecuados para diferentes aplicaciones. La replicación del contenido para CDNs de tipo comercial se realiza alrededor de todo el mundo y esto para clientes exigentes que necesitan entregar de manera oportuna grandes cantidades de información. El almacenamiento en caché del contenido de CDN puede ser *intra-clúster* e *inter-cluster* como se muestran en la figura 21:

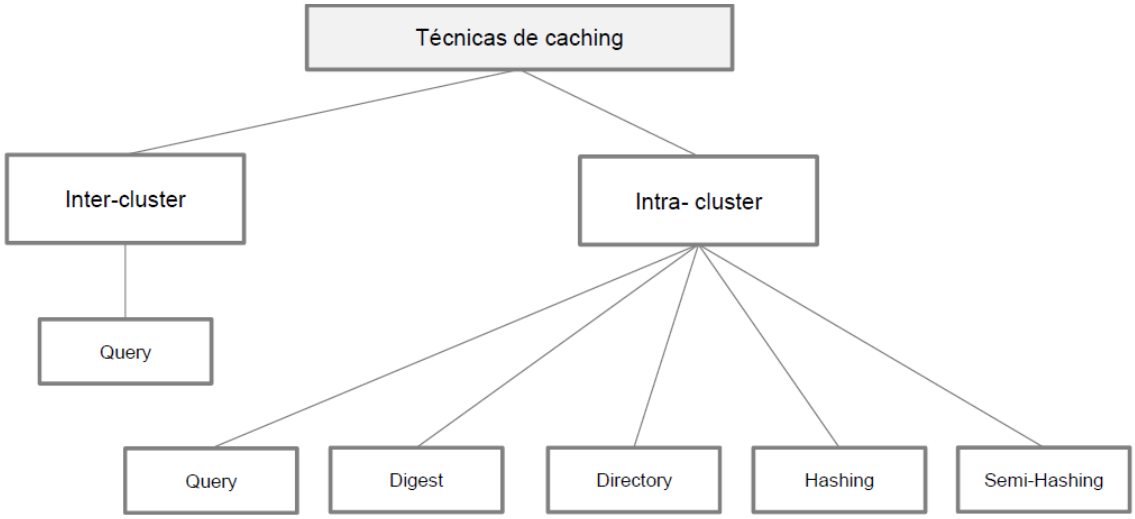


Figura 21. Taxonomía de técnicas de caching (2009) (Fuente: R. Buyya)

- ***Inter-cluster Caching***

Esta técnica sólo dispone de esquemas basados en peticiones (*query-based*) (Jian et al, 2003). En este caso, cuando un clúster no es capaz de servir un cierto contenido solicitado, realiza la petición a un clúster cercano y si no consigue el contenido de este clúster, puede preguntar a otro clúster vecino. Los *surrogates* dentro del clúster suelen emplear un esquema de tipo *hashing*, y el servidor representativo de ese clúster (*front end*) solamente contacta con el *surrogate* designado por la función hash para servir al contenido solicitado. Un sistema *hashing* no resultaría apropiado para el *caching* cooperativo *inter-clúster*, debido a que los *surrogate servers* de diferentes *clusters* están típicamente distribuidos geográficamente y el impacto en la latencia sería considerable. Los esquemas de tipo *digest* y *directory* tampoco son adecuados porque el tamaño de la información a almacenar puede ser considerable entre tantos *surrogates* dispersos en diferentes *clústers* (Molina, 2013). Por lo tanto, este sistema utiliza el esquema basado en *hash* para el enrutamiento de contenido *intra-cluster* y el esquema basado en consultas para el enrutamiento de contenido *inter-cluster*. Este enfoque mejora el rendimiento, ya que limita la inundación de tráfico de consultas y supera el problema de los retrasos al recuperar contenido desde servidores remotos a través de la

utilización de un número con valor de tiempo de espera y TTL (*Time-to-Live*) con cada mensaje de consulta.

- ***Intra-cluster Caching***

Es un esquema basado en petición (*query-based*) (Wessels & Claffy, 1997), que se presenta al momento de un error de cache (*cache miss*), donde un *surrogate* de la CDN realiza una operación de requerimiento a través de un *broadcast* a otros *surrogates* con los que coopera dentro de la CDN. El problema principal de este esquema es el exceso de tráfico en la red durante la petición, así como el retardo incurrido, ya que el *surrogate server* de CDN debe esperar hasta la última contestación de fallo para determinar cuál de sus *peers* tiene el contenido solicitado, debido a estas desventajas, se debe considerar que el sistema basado en peticiones (*query*) tiene problemas de sobrecarga durante la implementación.

- ***Digest-based***

En un esquema de este tipo, cada uno de los *surrogates* dispone de un resumen con la información disponible en el resto de *surrogates* cooperativos, que se actualiza periódicamente. Por lo que supera los inconvenientes de sobre flujo en las peticiones o consultas del esquema basado en peticiones. En este esquema, cada uno de los servidores CDN mantiene un compendio de contenidos en poder de los otros *surrogates* cooperantes (Rousskov & Wessels, 1998). Los *surrogates* cooperativos son informados acerca de cualquier tipo de actualización del contenido por el servidor CDN encargado de las actualizaciones. En el registro del contenido, un servidor CDN puede tomar la decisión de enrutar una solicitud de contenido a un *surrogate* en particular. El principal inconveniente es que sufre de sobrecarga de tráfico de actualización, debido al cambio frecuente para asegurarse de que los *surrogates* cooperativos mantengan la información correcta de los otros (Halpert, 2011).

- ***Directory-based***

El esquema basado en directorio (Gadde S et al, (1997) no es más que una versión centralizada del esquema *digest*, donde un servidor que ahora estará centralizado almacena la información de localización de contenido de todos los *surrogates* que son cooperativos dentro del *cluster*. Los *surrogates* sólo notifican al servidor de directorio cuando se producen actualizaciones locales, y lo consultan siempre que se produce un fallo local (*cache miss*). Este mecanismo representa un cuello de botella potencial y un punto centralizado de fallo, debido a que recibe la información de todos requerimientos y respuestas.

- ***Hashing***

En un esquema basado en *hashing* (Karger et al, 1999), los *surrogates* cooperativos mantienen la misma función *hash*. Cada uno de los *surrogates* almacena un cierto contenido en base a la URL del contenido, las direccionamiento IP de los *surrogates y* la función *hash*. Los esquemas basados en funciones *hash* son los más eficientes al tener menor sobrecarga de implementación y alta eficiencia en la compartición de contenido, sin embargo, este esquema no escala bien con peticiones locales y contenido multimedia ya que las peticiones pueden ser encaminadas a un *surrogate* lejano, en vez de uno más cercano (Valloppillil & Ross, 1998).

Para solventar este problema, se puede emplear un esquema *semi-hashing* (Ni & Tsang, 2005). En este caso, un *surrogate* divide su espacio de almacenamiento en dos zonas: una primera zona donde se aloja el contenido más popular para sus usuarios locales (incrementando el *hit rate* local), y una segunda zona donde se aloja el contenido indicado por la función *hash* en la porción cooperativa (Ni et al, 2003).

Por esta razón el enrutamiento *Inter-cluster content*, es necesario cuando el enrutamiento *intra-cluster content* falla.

2.2.3.4.2 La Actualización de Caché (*Cache Update*).

Este mecanismo sirve para comprobar automáticamente el sitio de alojamiento para los cambios y así poder recuperar contenido actualizado que pueda ser entregado a los equipos de bordes en las diferentes redes (Bartolini et al, 2004). Los mecanismos estándar adoptados en *proxy caching*, no garantizan un contenido actualizado, ya que el contenido almacenado en los *cache servers* estándar, no cambia cuando el contenido de origen cambia. Los objetos almacenados por caché en los *surrogate servers* de un CDN asocian tiempos de expiración después de lo cual se consideran obsoletos. Asegurar la frescura de los contenidos es necesario para servir a los clientes con la información actualizada. Para evitar retrasos involucrados en la propagación de los contenidos, un proveedor de CDN debe ser consciente de que el contenido puede ser inconsistente y/o ha expirado. Para gestionar la consistencia y la frescura de los contenidos en las réplicas, CDN despliega diferentes técnicas de actualización de memoria caché, como se muestra a continuación (fig. 22):

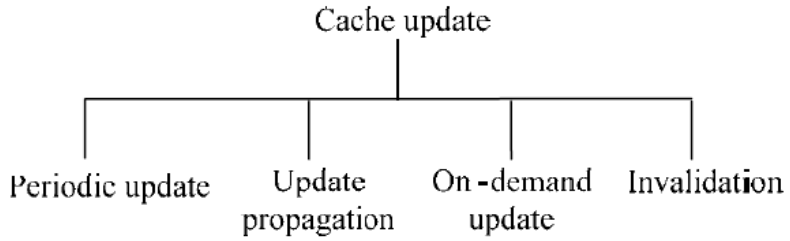


Figura 22. Cache Update (Fuente: R. Buyya)

- **Actualización Periódica**

Las actualizaciones periódicas son el método más común para actualizar la caché. Para garantizar la consistencia del contenido, el proveedor de contenido configura sus *origin Web servers* y proporciona instrucciones a los cachés acerca de qué contenido es cacheable, durante cuánto tiempo un contenido se considera válido, cuándo se debe interrogar al *origin server* por contenido actualizado, etc. (Gayek et al, 2004) De esta forma, las cachés se

actualizan en un modelo regular. Sin embargo, este mecanismo incurre en niveles significativos de tráfico generado en cada intervalo de actualización.

- **Propagación de la Actualización**

Este esquema de propagación de la actualización, sólo se genera cuando hay un cambio en el contenido, y se notifica a los *surrogates (pushing*) desde el servidor origen. El problema en este caso es que en una situación de cambio frecuente de contenido en el servidor origen, se genera demasiado tráfico de actualización de datos hacia los *surrogates*.

- **Actualización Bajo Demanda**

En este esquema de actualización bajo demanda, sólo se actualiza el contenido en los *surrogates* cuando se lo solicita explícitamente. La desventaja de este mecanismo es el tráfico generado entre *surrogate y origin server* es para garantizar que el objeto solicitado por el cliente sea el último actualizado.

- **Invalidación**

Este último método de actualización envía un mensaje de invalidación a todas las cachés cuando se realiza un cambio en el *origin server*, y donde los *surrogates* bloquean dicho contenido hasta que no se obtenga nuevamente del servidor (Halpert, 2011).

Generalmente, las CDNs comerciales ofrecen a los proveedores de contenido el control sobre el grado de actualidad (*freshness*) y garantizan la consistencia entre todos los múltiples *sites* de la CDN. Por otro lado, los proveedores de contenido pueden crear sus propias políticas de *caching* específicas, en un formato único de cada proveedor de CDN, quien propaga el conjunto de reglas a sus *surrogates.*

2.2.4 Sistema de Encaminamiento y Redirección (*Request-routing*).

Son los responsables de direccionar las solicitudes de los clientes a los *edge servers* apropiados e interactuar con el componente de distribución para mantener una vista actualizada del contenido almacenado en los *cache servers* de CDN. Trata a los algoritmos de solicitud de enrutamiento, considerando los mecanismos de redirección de clientes y las metodologías para las peticiones hacia los *surrogates* más cercanos,

Este sistema es responsable de enrutar las peticiones de los clientes al *surrogate server* más apropiado para entregar el contenido. La aproximación consiste de una colección de elementos de red que se encargan de redirigir las peticiones a los *surrogates* más cercano al cliente. Sin embargo, el *surrogate* más cercano no siempre es el mejor servidor para satisfacer al cliente (Chen et al, 2005), por lo cual el sistema considera una serie de métricas antes de tomar su decisión de enrutamiento y algunos ejemplos de las métricas que considera son: la proximidad a una red, el nivel de latencia percibida por el cliente, la distancia y la carga en memoria, procesamiento que tenga en ese momento el *surrogate* a ser elegido e identificar el de mejores prestaciones que sirva el requerimiento. La selección de contenido y distribución en una CDN, sea esta por *full-site o partial-site* tiene un impacto directo en como se diseña el sistema el encaminamiento (fig. 23).

El sistema *request-routing* consta básicamente de dos partes: el algoritmo de redirección y el mecanismo de redirección (Sivasubramanian et al, 2004). El algoritmo de redirección especifica cómo seleccionar un *edge server* adecuado en respuesta a un requerimiento y se invoca cada vez que se recibe una petición. Mientras que el mecanismo de redirección es una forma de notificar al cliente el *surrogate server* que debe contactar.

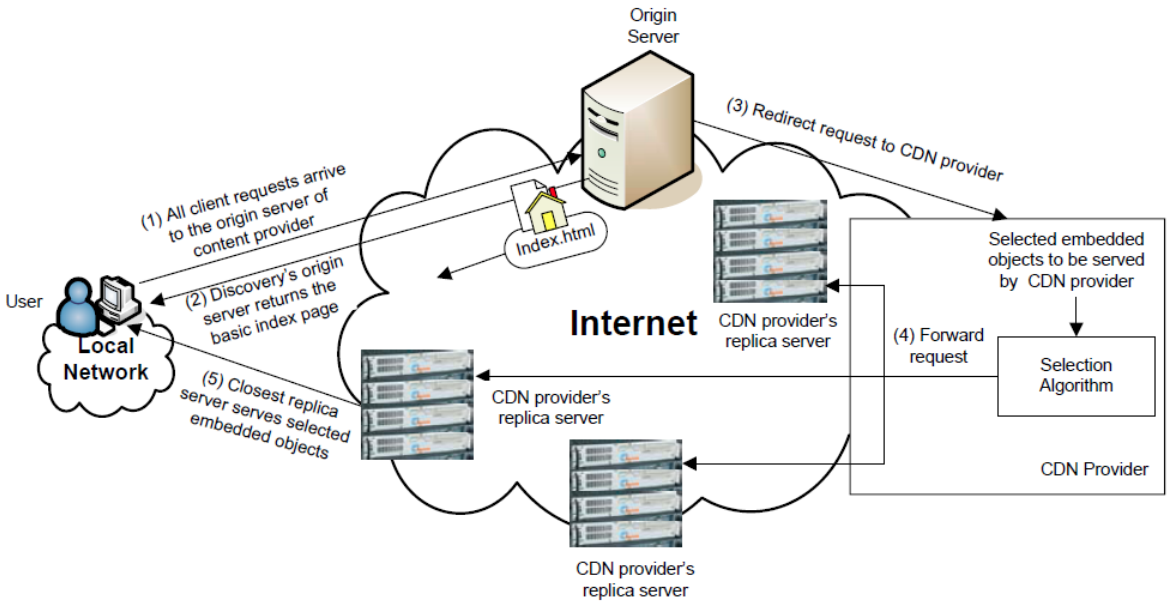


Figura 23. Request-routing in a CDN environment (Buyya R., 2010)

El proceso de encaminamiento y redirección dentro del ambiente de CDN (interacción cliente-servidor) se puede describir en los siguientes pasos:

- El cliente solicita el contenido del proveedor de contenidos, indicando la URL en el navegador Web, el requerimiento del cliente, es redirigido en primera instancia hacia su *origin server*.

- Cuando el *origin server* recibe un requerimiento, toma la decisión de proporciona únicamente contenido como información básica (objetos de poco tamaño, como el *index* del *web site*).

- Mientras que para servir o distribuir los objetos de gran tamaño y que se solicitan con mayor frecuencia, las solicitudes son redirigidas por parte del *origin server* del proveedor a un espacio de nombres dentro de la CDN. El tipo de contenido descrito aquí, es susceptibles de consumir un elevado ancho de banda, como lo hacen los (objetos embebidos, barras de navegación, banners, es decir el contenido más fresco)

- El proveedor de la CDN emplea su algoritmo de redirección para decidir cuál es el *replica server* más adecuado y más próximo al cliente para servir los objetos embebido que sean solicitados.

- El *surrogate server* que ha sido seleccionado en el proceso, obtiene los tipos de objetos embebidos del *origin server*, con lo cual sirve el contenido solicitado y lo cachea para servir peticiones subsecuentes.

2.2.4.1 Algoritmos para el Request-routing

Los algoritmos del mecanismo de redirección son de dos categorías (fig. 24):

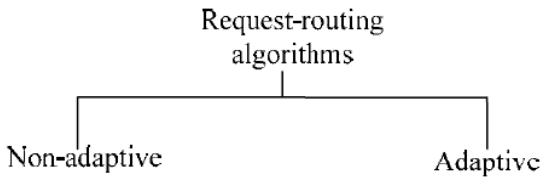


Figura 24. Request-routing algorithms (Buyya R., 2010)

- **Algoritmos No Adaptativos**

Esta categoría distribuye todos los requerimientos a los *cache server* de CDN e intenta balancear la carga entre ellos, emplea cierta heurística para tomar la decisión en lugar de considerar las condiciones del sistema, esto debido a que asume que todos los *cache server* tienen las mismas capacidades de procesamiento y que todos los servidores pueden servir todos los requerimientos que realice un cliente. Un tipo de algoritmo básico como este, se lo considera un *Round-robin (Sivasubramanian et al, 2004)*, pero no se lo podría considerar para sistemas de distribución extensos tipo WAN, donde los *cache servers* están localizados en distintos lugares.

Los algoritmos no adaptativos son más sencillos de implementar, por ejemplo estos tienen gran eficacia al ser implementados en *clusters*, donde los *replica server* están localizados en el mismo lugar. Por otro lado, los adaptativos suelen ser más complejos al tener que adaptarse a las condiciones cambiantes de la red y/o de los servidores en tiempo real. Esto los hace más robustos (Wang, 2002) que los no adaptativos, especialmente en los eventos de tipo *flash-crowd*, basado en el cálculo de una función hash considerando la URL del contenido, ya que estos últimos (no adaptativos) sólo son eficientes cuando se cumplen las condiciones o asunciones heurísticas.

En otro ejemplo de algoritmo no adaptativo, los *surrogates* se ordenan dependiendo de la carga esperada en cada uno de ellos. Dicha predicción está basada en el número de peticiones que cada servidor ha satisfecho, y considera también la distancia cliente-servidor. De esta forma, se balancea la carga entre los servidores en base a estos dos parámetros. Si bien el algoritmo resulta eficiente (Aggarwal & Rabinovich, 1998), la latencia percibida por el cliente es mejorable. Sin embargo, en la mayoría de estos algoritmos se puede producir una redirección a un servidor sobrecargado, lo que degrada el rendimiento percibido por el usuario. Varios otros algoritmos no adaptativos interesantes son implementados en el *Cisco Distributed Director* (Delgadillo, 1997). Este cálculo determina un encaminamiento eficiente a un anillo lógico de *cache servers*. Algunas variaciones de este algoritmo se han empleado en contextos de *caching intra-clúster* (Chen, 2005) y sistemas P2P de compartición de ficheros (Balakrishnan & Kaashoek et al, 2003).

- **Algoritmos Adaptativos**

En esta categoría se consideran las condiciones actuales y en tiempo real de la red y/o los servidores para decidir el *cache server* más adecuado para distribuir el contenido, teniendo en cuenta ciertas métricas (Wang et al, 2002) (congestión en los enlaces dentro de la red, carga de los *replica servers*).

Globule (Pierre & Van Steen, 2006) emplea un algoritmo de encaminamiento adaptativo que selecciona el *replica server* más cercana a los clientes en términos de proximidad de red (Szymaniak et al, 2003). La estimación de la métrica en Globule se basa en la longitud del

trayecto (*path*) que se actualiza periódicamente y el servicio de estimación de métrica es pasivo, con lo cual no se introduce tráfico adicional en la red. Esto a pesar que en (Huffaker et al, 2002) se muestra que dicha estimación de la métrica no es muy exacta.

Akamai (Dilley et al, 2002) emplea un algoritmo complejo de encaminamiento adaptativo y que tiene gran eficacia en evitar *flash-crowds*. El algoritmo considera una serie de métricas, como son la carga de los *surrogates*, la fiabilidad de las cargas entre cliente y *surrogates* y el ancho de banda disponible en cada *surrogate*. El algoritmo es propietario de Akamai y los detalles tecnológicos no están disponibles.

Cisco Distributed Director (Delgadillo, 1997) ha implementado un algoritmo de encaminamiento adaptativo. El algoritmo desplegado en este sistema tiene en cuenta una combinación ponderada de tres métricas, llamadas, *inter-AS distance*, *intra-AS distance* y latencia *end-to-end.*

Otros algoritmos de encaminamiento adaptativos (Andrews et al, 2002) (Ardaiz et al, 2001) están basados en la latencia cliente-servidor. En este mecanismo, se tiene en cuenta los logs de acceso de clientes o medidas pasivas de latencia en los servidores y de esta forma, se encamina una petición al *replica server* (*surrogate*) que reporta lo mínimo de latencia hacia el cliente. Si bien estos algoritmos resultan eficientes, se requiere el mantenimiento de una base de datos central con integración de medición continua, lo que limita la escalabilidad del sistema donde se emplean estos algoritmos (Sivasubramanian et al, 2004).

2.2.4.2 Mecanismos para el Request-routing

Los mecanismos de redirección informan al cliente el resultado de la selección del *surrogate server*, que se obtuvo durante el proceso del algoritmo de encaminamiento. Sirven para dirigir una solicitud al *Surrogate* más cercano y ser capaz de re-enrutar rápidamente las solicitudes de contenido, en respuesta a las ráfagas de tráfico y la congestión que se revelen en la actividad de medición, utilizando mecanismos que eviten el efecto *Flash-Crowd* o efecto *SlashDot* (Buyya et al, 2008). Se los puede clasificar dependiendo de varios criterios, aunque

típicamente dependerán de la variedad de procesamiento en las solicitudes. A continuación se muestra su clasificación (fig. 25):

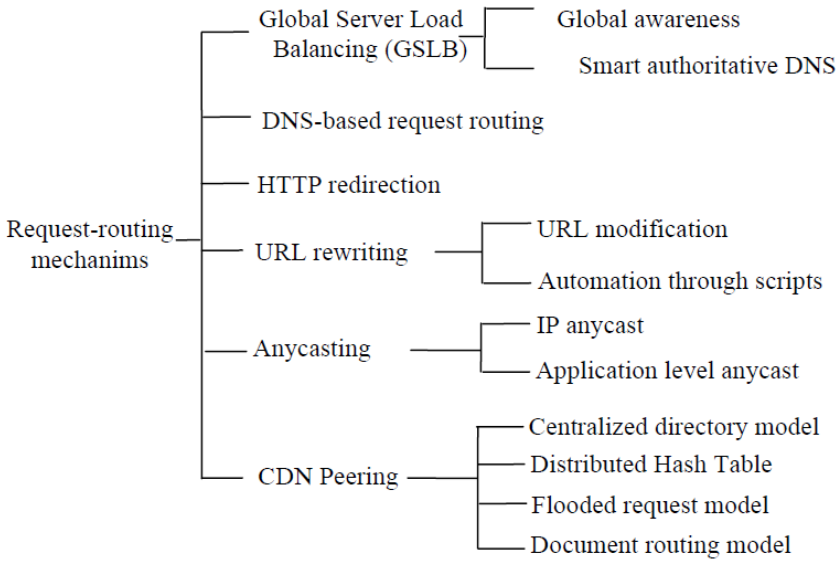


Figura 25. Request-routing mechanisms (Buyya R., 2010)

- ***Global Server Load Balancing* (GSLB)**

En este mecanismo de balanceo global de carga, los *service nodes* (encargados de servir el contenido a los usuarios finales), están conformados por un *Web switch* global habilitado con GSLB y un conjunto de *Web servers* desplegados por todo el mundo (Pathan & Buyya, 2009). Que pueda soportar balanceo de carga global dependerá de dos capacidades especiales de los *service nodes* (*surrogates*). La primera de ellas es el *global awareness* y la segunda es el *smart authoritative DNS* (Hofmann & Beaumont, 2005). En el balanceo de carga local, cada nodo (servidor) es consciente del estado y rendimiento de los servidores directamente conectados. En GSLB, existe un nodo (*switch*) que es consciente de la información disponible en los otros nodos e incluye sus direcciones IPs virtuales en su lista de servidores conocidos (Molina, 2013). En la configuración GSLB del *site* de la CDN dispone de un *web switch* que tiene un conocimiento global del resto de los *service nodes* así como de su información de rendimiento. Para lograr este conocimiento global, los switches GSLB actúan a modo de

smart authoritative DNS para ciertos dominios. Una de las ventajas dentro del GSLB es que el administrador de red puede añadir capacidades GSLB sin necesidad de introducir dispositivos adicionales en la red. La gran ventaja del mecanismo GSLB es la de ser capaz de seleccionar el *surrogate* más adecuado para cada petición, ya sea local (de un grupo de servidores conectados directamente al switch) o global (conectados a un *service node* remoto), todo esto gracias a que cada nodo dispone de un conocimiento global del resto de nodos (Barbir et al, 2003). En gran parte la mayor desventaja del mecanismo es la configuración manual de GSBL sobre los *service nodes* o switches.

- ***DNS-based Request-routing***

En este mecanismo, los servicios de distribución de contenido confían en servidores DNS modificados, que son capaces de realizar un mapeo entre el nombre simbólico de un *surrogate server* y su direccionamiento IP, considerando que un nombre de dominio puede tener múltiples IPs (Molina, 2013). Cuando un usuario final realiza una petición de contenido, el *DNS server* del proveedor de servicios, devuelve las direcciones IPs de servidores (*surrogates*) que disponen de una réplica del contenido u objetos solicitados y es donde el *DNS resolver* del usuario final debe seleccionar un *surrogate* de los disponibles. Para lograr dicha decisión, el *DNS resolver* puede enviar mensajes de prueba hacia los *surrogate servers* y elegir el de menor tiempo de respuesta, al igual que podría tomar información histórica de los clientes basándose en el acceso previo a dichos *surrogates* (Buyya et al, 2008). Es importante conocer que los proveedores de servicios CDN, sean estos globales (*full-site*) como parciales (*partial-site*), ambos emplean redirección DNS (Shaikh et al, 2001). La ventaja de este mecanismo de redireccionamiento es la transparencia con la que los servicios y contenidos quedan referenciados por sus nombres de DNS (Mao et al, 2002), y no por su direccionamiento IP y esto se ha examinado en varios estudios, donde se ha probado en detalle el rendimiento y la efectividad del encaminamiento basado en el uso de DNS (fig. 26).

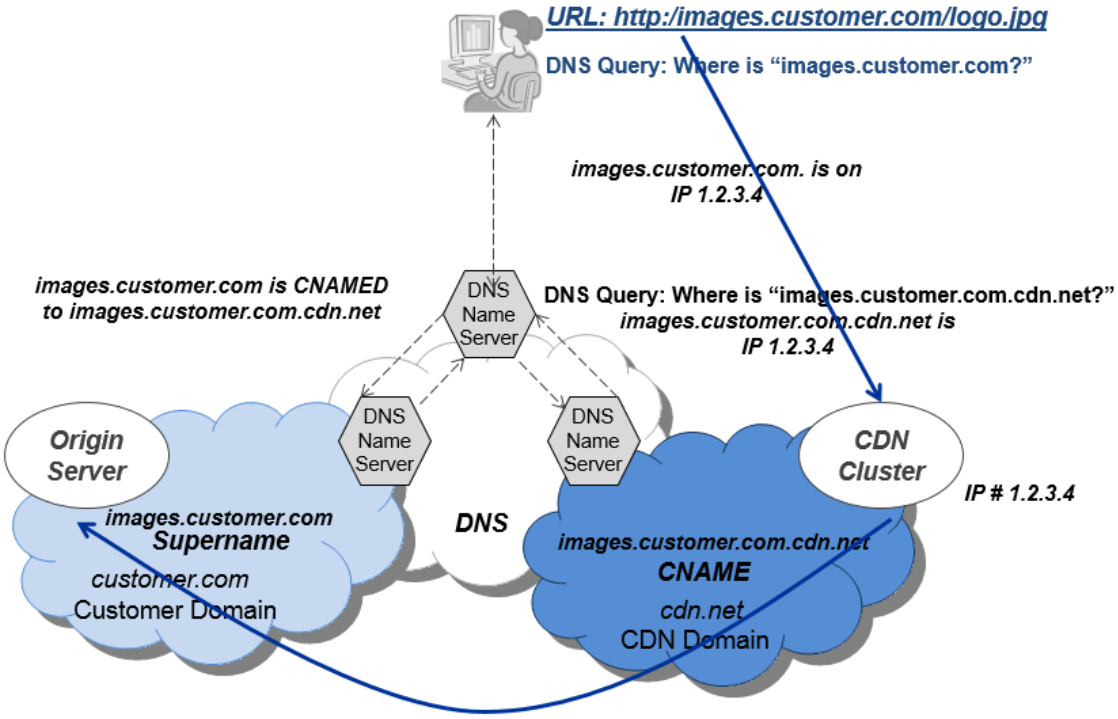


Figura 26. DNS - Object Delivery (Caching) (Fuente: Level3.com)

El esquema DNS es extremadamente popular debido a su simplicidad e independencia de cualquier servicio de replicación, puesto que está incorporado en el servicio de resolución de nombres (como un directorio), y puede ser empleado por cualquier aplicación de Internet (Sivasubramanian et al, 2004). La gran desventaja de la redirección DNS es que se incrementa la latencia de red porque se requiere un tiempo adicional en la resolución de nombres (*DNS lookup*). En su mayoría los administradores de CDNs abordan esta problemática dividiendo el servicio DNS en dos niveles (*low-level DNS y high-level DNS*) como método de distribución de carga (Krishnamurthy et al, 2001). Otra limitación del sistema basado en DNS, es que en la resolución del *surrogate* se dispone de la dirección IP del servidor DNS local, y no de la dirección IP del cliente, por lo que se puede producir una mala resolución si el cliente y el servidor DNS local no están próximos y por lo tanto no le permite ser escalable (Pathan & Buyya, 2009). Pero, la desventaja más significativa en la redirección DNS es que no se pueden controlar todas las peticiones debido al sistema de *caching* de datos de DNS internos, tanto del lado del ISP como a nivel de cliente. Resulta

muy conveniente evitar el *caching* en el cliente ya que en caso de fallo para alcanzar el *surrogate server*, el sistema no puede hacer adecuadamente.

- **Redirección HTTP**

Este mecanismo propaga la información acerca de los *surrogates* en las cabeceras HTTP. El protocolo HTTP permite a un *Web server* responder a un cliente para que realice un nuevo requerimiento hacia otro servidor (Molina, 2013). La redirección HTTP se puede utilizar tanto para *full-site* como *partial-site* para la selección y distribución del contenido. Este mecanismo puede ser utilizado para construir un *Web server* especial, el cual pueda aceptar solicitudes de clientes, elegir *replica servers* y redirige a los clientes a esos servidores. Requiere cambios en ambos servidores Web y de cliente para procesar cabeceras extras. Los clientes también deben modificarse para implementar la selección del servidor de réplica. La ventaja de este mecanismo es su flexibilidad y simplicidad, además de poder gestionar la redirección con una granularidad fina (Peng, 2003). Sin embargo, la redirección HTTP ofrece poca transparencia, al tener que introducir cambios en cada página web; adicionalmente, la sobrecarga en el servidor al gestionar las redirecciones es elevada.

- ***URL rewriting - Hyperlink***

Pese a que la mayoría de las CDNs en uso actuales utilizan DNS como mecanismo de encaminamiento, algunos sistemas emplean la modificación o rescritura de URLs. Normalmente se emplea en la modalidad de replicación parcial donde los objetos embebidos se envían con una URL modificada. En este esquema, el *origin server* redirige a los clientes a diferentes *surrogates* al modificar de manera dinámica la URL de los enlaces que direccionan a los objetos embebidos (la página principal la sirve el propio servidor origen). Para automatizar el proceso, las CDNs actuales proporcionan *scripts* especiales que analizan de forma transparente el contenido web y modifican las URLs embebidas (Krishnamurthy et al, 2001). El mecanismo de modificación de URLs puede ser proactivo o reactivo. En la modalidad proactiva, se crean las URLs de los objetos embebidos de la página principal antes

de cargar el contenido en el *origin server*. En la modalidad reactiva, las URLs se modifican cuando la petición del cliente llega al *origin server*. La gran ventaja de la modificación de URLs es que los clientes no están asociados a un solo *surrogate,* ya que las URLs contienen nombres DNS que apuntan a un grupo de *surrogates*. Además, se puede conseguir un elevado nivel de granularidad, ya que cada objeto embebido se puede gestionar de forma independiente. La gran desventaja de este mecanismo es el retardo adicional debido al *URL-parsing*, así como el posible cuello de botella que puede experimentar alguno de estos elementos embebidos.

- ***Anycasting***

Este mecanismo se divide en IP a*nycasting* y *Application-level anycasting*. En IP *anycasting* se asume que la misma IP se asigna a un conjunto de *hosts* (servidores) y cada router almacena una ruta en su tabla de enrutamiento al *host* más cercano a dicho router (Partridge et al, 1993). De esta forma, routers diferentes disponen de rutas distintas a la misma dirección IP. Una desventaja de *anycasting* IP es una parte del espacio de direcciones IP donde se asigna direccionamiento *anycast*, lo cual no se llevó a cabo en IPv4 e implicaría modificar los routers y clientes, lo que repercutiría en costos de operación. En el mecanismo *anycasting* de nivel de aplicación el servicio consiste en un conjunto de *anycast resolvers*, que realizan el mapeo de los nombres de dominio *anycast* a sus direcciones IP. Los clientes interaccionan con los *anycast* resolver generando una consulta *anycast*, para lo cual el *resolver* procesa la petición y genera una respuesta (Fei et al, 1998). Una base de datos de métricas, asociadas con los *anycast resolvers* contiene información de rendimiento sobre cada *surrogate*, el cual se estima en base a la carga y la capacidad de procesamiento de cada servidor. Una ventaja del mecanismo *anycast* a nivel de aplicación es su gran flexibilidad, puesto que es altamente configurable y no implica modificación alguna en los routers de la red.

- ***CDN Peering***

Cuando se habla de las redes de contenido basadas en tecnología *peer-to-peer*, se entiende a las redes que están formadas por conexiones simétricas entre *hosts* de computadoras (servidores) y que distribuyen contenido entre sus *peers*. En estos casos, una CDN puede alcanzar a un mayor número de usuarios si emplea servidores y *forward proxies* cercanos de otra CDN aliada con las que puede realizar interconexiones hacia otras CDNs (*peering* CDNs), de cierta manera similar a los puntos neutros entre ISPs (Pathan et al, 2007). Las CDNs con funcionalidad de *peering*, son más tolerante a fallos debido a que la información de la red de recuperación necesaria puede ser desarrollada por los mismos miembros del *peering* en lugar de depender de una infraestructura dedicada como lo hacían las CDN tradicionales. Para localizar el contenido en una CDN de tipo *peering*, se puede emplear *centralized directory model*, un esquema DHT (*Distributed Hash Table*), *flooded request model* o un *document routing model* (Milojicic et al, 2002).

- **Modelo de Directorio Centralizado:** En este modelo los *peers* contactan de un directorio centralizado donde todos los *peers* publican el contenido que quiere compartir con el resto de *peers*. Cuando se realiza un requerimiento, se responde con el *peer* que dispone de dicho contenido solicitado (Pathan & Buyya, 2009). Si hay más de un *peer* que mantengan el contenido, se selecciona aquel *peer* con mejores prestaciones de métricas basándose en la proximidad de red, disponibilidad alta de ancho de banda, un enlace sin congestión o gran capacidad de almacenamiento. Una vez que se recibe la respuesta por parte del director, se contacta al *peer* que ha sido referido y que mantiene el contenido. La mayor desventaja de este mecanismo es que al tener un solo directorio centralizado representa punto de fallo sensible, perdiendo la escalabilidad del sistema ya que su limitación será la capacidad. Dos ejemplos de directorios centralizados para adquirir archivos FTP son *Archi* y *WAIS*.

- ***Distributed Hash Table* (DHT):** En este sistema los *peers* se indexan con claves *hash* dentro de un sistema distribuido (Stoica et al, 2003). De esta forma, el *peer* que contenga el contenido deseado puede ser localizado empleando solicitudes complejas

de descubrimiento. La ventaja de este mecanismo es que permite el balanceo de carga para aliviar el exceso de los *peer* más cargados (Byers et al, 2003), siendo un ejemplo de este sistema *Chord*.

- **Modelo Flooded-request*:*** En este sistema la petición de un *peer* se la realiza a todos los *peers* conectados a través de un *broadcas*t, a su vez los que reciben la petición lo difunden a los *peers* directamente conectados y el proceso continúa hasta que la respuesta es contestada o se alcanza el límite del *broadcast* (Berer, 2002). Entonces conociendo esto, la desventaja es que el mecanismo genera tráfico de red innecesario y requiere un enorme ancho de banda, alcanzando inconvenientes de escalabilidad y obviamente el tamaño de la red será limitado. Un ejemplo de este mecanismo lo constituye el protocolo *Gnutella* (Wang et al, 2007).

- **Modelo Document-routing*:*** En este modelo se emplea un *peer* autoritativo para obtener el contenido. Cada peer resulta útil en el modelo, ya que todos completan parcialmente la información de referencia (Hofmann & Beaumont, 2005). En este esquema, cada *peer* es responsable de un rango de identificadores o IDs. Cuando un *peer* desea obtener un fichero, envía una solicitud de requerimiento del identificador de fichero. La petición se reenvía al *peer* cuyo ID sea idéntico al identificador del fichero, ya con esto, cuando se localiza el fichero, se transfiere al *peer* solicitante. Ya en el desempeño, una gran ventaja de este mecanismo es que puede realizar una búsqueda comprensiva dentro de un número acotado de pasos O (log n), además de mostrar un buen rendimiento de operatividad y escalabilidad suficiente para crecer significativamente.

CDN	Técnica de redirección
Akamai	Algoritmos de redirección adaptativos que tienen en cuenta la carga del servidor y otras métricas Combinación de redirección basada en DNS y URL rewriting
Edge Stream	Redirección HTTP
LimeLight Networks	Redirección basada en DNS
Mirror Image	Redirección con balanceo global de carga (GSLB), empleando tanto conocimiento global como DNS inteligente
CoDeeN	Algoritmo de redirección que tiene en cuenta ubicación de la petición, carga sistema, fiabilidad e información de proximidad Redirección HTTP
Coral	Algoritmos de redirección con estimación de la ubicación al explotar medidas de red y almacenar detalles de red Redirección basada en DNS
Globule	Algoritmos de redirección considerando proximidad proporcionada por AS Redirección basada en DNS

Figura 27. Encaminamiento-redirección. Correspondencia principal CDNs (Fuente: R. Buyya)

2.2.5 Medición del Desempeño

Aquí se hace hincapié en la medición del desempeño de CDN, las metodologías de evaluación de prestaciones, las métricas de rendimiento y las técnicas de adquisición de estadísticas de red utilizadas para CDN. Son aquellos componentes que mantienen los logs de registro de acceso y estadísticas de utilización de los servidores de CDN, esta información es crucial para el reporte de uso y facturación del cliente.

Este mecanismo debe de agregarse a los *routers* de acceso cooperativo, para tener acceso inmediato a la información real del volumen de tráfico de internet, con el fin de mantener el panorama con la mejor ruta que solicitará un usuario al *replica server* (Bartolini et al, 2004), mientras que el servicio de gestión sirve para administrar los componentes de red, gestionar

la contabilidad, monitorizar, reportar el uso y acceso al contenido (Buyya et al, 2008). Las medidas de rendimiento de un CDN fueron concebidas para medir de alguna manera su habilidad o desempeño para servir al cliente con el contenido o servicio deseado. En gran frecuencia se consideran cinco tipos principales de métricas por parte de los *content providers* para evaluar el rendimiento de una CDN:

- ***Cache Hit Ratio***

Se define como la tasa de ratio entre el número de contenidos cacheados frente al total de contenidos solicitados (Douglis & Kaashoek, 2001). Un valor elevado de este ratio indica que la CDN está empleando una técnica efectiva de *caching* para administrar sus *proxy caches*.

- **Ancho de Banda Reservado *(Bandwidth)***

Hace referencia al ancho de banda empleado por el *origin server*, la cual es medida en bytes y se recupera desde el *origin server*.

- **Latencia**

Se refiere al tiempo de respuesta registrado por el usuario. Un valor de latencia reducido significa el decremento en las reservas del ancho de banda del *origin server* (Gadde et al, 2000).

- **Utilización del Surrogate Server**

Se refiere a la fracción de tiempo en la cual el *surrogate server* permanece ocupado. Este tipo de medición es empleado por los administradores, para calcular la carga del CPU, el número de peticiones atendidas y el uso de almacenamiento de entrada o salida I/O (Douglis & Kaashoek, 2001).

- **Confiabilidad**

La medición de las pérdidas de paquetes es utilizado para determinar la fiabilidad de una CDN. Contar con una alta disponibilidad representará que la CDN incurra en una menor pérdida de paquetes y este siempre disponible para ser utilizada por los usuarios (Krishnamurthy et al, 2001).

Una correcta evaluación del rendimiento puede ser obtenido basándose en medidas de rendimiento interno, así como también se puede basar en la perspectiva del usuario final, a través de las siguientes métricas:

2.2.5.1 Medidas Internas

La medición de los servicios de entrega de contenido a través de toda la red, puede realizarse mediante la recopilación y el análisis de los registros de los servidores de *caché y streaming*. También podría ser que los servidores CDN estén equipados con la capacidad de recopilar estadísticas con el fin de obtener una medición de extremo a extremo del rendimiento (Buyya et al, 2008). Adicionalmente, se pueden desplegar sondas en hardware o software, a través de toda la red que recolecten esta información con los *logs* de los servidores. Este tipo de mediciones también se puede realizar mediante el uso de algunas herramientas de terceros que puedan producir representaciones gráficas de los datos de rendimiento.

2.2.5.2 Medidas Externas

Adicionalmente a las medidas internas, las medidas externas llevadas a cabo con herramientas de terceros, como *Keynote Systems* (2008) permiten a los clientes validar con mayor confiabilidad el rendimiento de una CDN. Este proceso es eficiente y fiable puesto que estos terceros soportan el *benchmarking* de sistemas de cómputo conectados a través de

grandes redes de *backbone* distribuidas por toda la red de Internet en varias ciudades. Estos sistemas por ejemplo miden el rendimiento de un *Web site* en particular desde la perspectiva del cliente, considerando varias métricas significativas en el rendimiento de aplicaciones Web dentro de áreas críticas (Vakali & Pallis, 2003). *Giga Information Group* es uno de esos evaluadores que miden el rendimiento mediante el análisis de un CDN basándose en varios parámetros como la escalabilidad y disponibilidad, capacidad de gestión, el tipo de contenido soportado, facilidad de uso, la viabilidad de la información, los costos, la cobertura y la presentación de informes.

2.3 CDN Y EL ECOSISTEMA DE VIDEO

Como el vídeo comenzó a ser integrado en los sitios web, CDN surgió para distribuirlo, pero inicialmente el contenido era totalmente pre-grabado y entregado bajo demanda. En cierto sentido, esto significaba que el vídeo era sólo un gran archivo de datos. Pero con la gran diferencia, de que no se trataba solo de un trozo de texto de poco peso, como para que el usuario lo pueda descargar en su totalidad, lo que el usuario necesitaba era reproducirlo y dejar de verlo en algún momento. Por lo tanto, incluso el *video on-demand* se requiere de *streaming* y esto estimuló el desarrollo de CDNs optimizados para una entrega con calidad (Schwarz, 2012).

Algunas grandes empresas del mercado como Netflix todavía están construyendo su propio CDN, pero la mayoría de las capas *over-the-top* (OTT) utilizan la infraestructura proporcionada por un proveedor especializado de servicios CDN. Estos servicios se acoplaron a las necesidades de los propietarios del contenido, que pagaron por un servicio CDN para entregar los archivos de vídeo bit a bit y por lo tanto cumpliendo la exigencia del usuario que era de visualización a su gusto.

Pero luego vino la creciente demanda del *live video streaming* y en un principio parecía que este sería el final del camino para el almacenamiento en caché. Parecía que el

live video no podría ser almacenado como el contenido que se grababa previamente, y que por lo tanto, la infraestructura de la CDN debe ser modificada para dar cabida a las transferencias punto a punto con elevado anchos de banda entre la fuente y el usuario. El problema era que los costos involucrados en la construcción y el mantenimiento de una transmisión en vivo para eventos populares serían extremadamente costosos y técnicamente exigente. Con este fin, una serie de tecnologías y la evolución de los modelos de negocio centrados en el vídeo y el cambio de estilo de vida de los consumidores, impulsaron el auge del video en línea, lo que fundamentalmente transformaría el medio de distribución, la televisión, la publicidad y los métodos de entrega de contenido.

Durante esta sección, se analizará en detalle los beneficios que CDNs trajo sobre los servicios que ofrece a las compañías de medios digitales, emisoras de contenido, empresas de marketing y publicidad a través de Internet o medios digitales experimentales y como su desempeño mejora la calidad de servicio sobre el mercado del *progressive download, linear, Video on Demand (VoD)* y del *live streaming*; sea este a través de dispositivos móviles, televisores inteligentes, consolas de juegos online o con dispositivos *set-to-box* (Pathan et al, 2014). El mercado de CDN es amplio y es de gran ayudan a entregar el contenido de vídeo a gran escala o continuo, así como al tráfico exclusivo para eventos especiales o temporales como un campeonato o una premiación.

El contenido de video fue sin duda el motor en el incremento de la utilización de los servicios de CDN. De hecho, más del 48% de los ingresos mundiales del sector CDN se asocia con el *video content delivery*. Cuando el consumo de contenidos en línea y de vídeo son ampliamente distribuidos, la calidad de la experiencia (*Quality of Experience QoE*) de los usuarios finales se vuelve significativamente importante, aún más cuando el contenido tiene que viajar por la Internet, mayor es la latencia en la entrega de contenido y el más pobre se vuelve la experiencia del usuario final (Wang et al, 2011).

Un CDN con todas sus ricas características técnicas, capacidades y *value-added services* (VASs) está en el centro del ecosistema de vídeo, mejorando la experiencia del usuario final mediante la aceleración de la forma de entrega de contenido (descarga/velocidad de transmisión) a través de los usuarios finales geográficamente distribuidos (fig. 28).

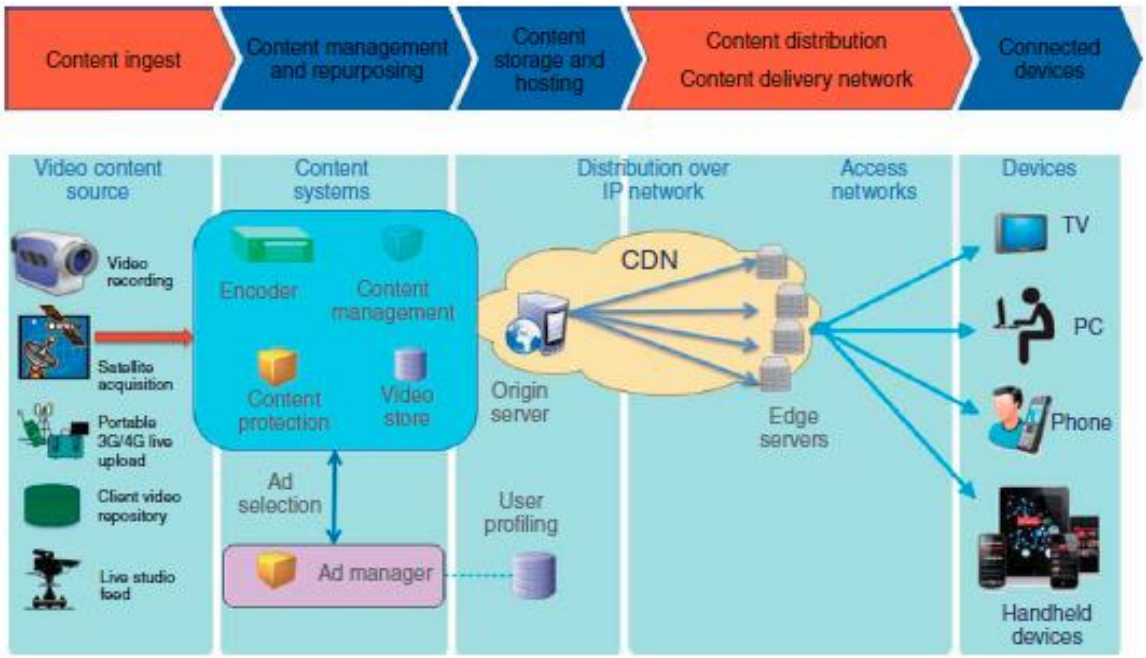


Figura 28. CDN en la cadena de suministro del ecosistema de vídeo de extremo a extremo. (2013) (Fuente: Ponce G.)

El contenido de video se inyecta desde la fuente, empaquetado en el adecuado formato (*format*) contenedor, mediante la codificación (*encoding*) de la plataforma y a continuación es indexado en el almacenamiento por parte del sistema de gestión de contenidos. CDN como el vehículo de entrega distribuye el contenido entre los *edge servers* para la entrega a los dispositivos conectados en IP. Dependiendo de los perfiles del usuario a través de informes y análisis se afirma que más oportunidades de generación de ingresos se crean a través de la inserción de anuncios. Estos anuncios se muestran incrustados en el *stream* de vídeo, en las páginas Web o dentro de la guía electrónica de programas (*electronic program guide* - EPG) de decodificadores y *set-top boxes*.

2.3.1 Transmisión en Vivo (*Live Streaming)*

Para entender cómo es que se va a enviar el contenido es necesario conocer una serie de conceptos que se utilizan en *streaming*. En primer lugar se encuentran los codecs de video, luego están los protocolos y cómo estos se desenvuelven en el *live streaming*. También es necesario comprender todos los modos de transmisión en la red IP, entre los cuales se puede identificar el modo *unicast* o *multicast*. Un caso de uso principal para CDN es proporcionar cobertura de *live streaming* de eventos deportivos, culturales y eventos corporativos, haciéndose disponible para su visualización en dispositivos conectados en IP. El escoger una tecnología de *live streaming* implica varias consideraciones entre las cuales está el entender las ventajas y desventajas de los diversos protocolos y los formatos de entrega de contenido durante el envío de la información (fig. 29).

Figura 29. Codecs y Protocolos de Streaming. (2013) (Fuente: Ponce, G)

2.3.2 Protocolos

Varios protocolos fueron creados, entre los cuales se encuentra RTP siendo el primero en este campo, esta HTTP con sus variaciones actuales y también RTMP que es un protocolo impulsado por el grupo técnico de Adobe (Torres, 2009). A continuación se describen algunos de los protocolos mencionados:

2.3.2.1 RTMP *(Real Time Messaging Protocol)*

Es un protocolo propietario de Adobe Systems para el *streaming* de audio, video y datos a través de Internet entre un *Flash Player* y un servidor *Flash Media*, por ejemplo *Wowza*, *Flash Media Server* o *Red5*. Además trabaja bajo el protocolo en capa de transporte TCP, el cual le permite hacer un *tracking* de las acciones del cliente desde la primera vez que se conecta hasta que se desconecta. Funciona transmitiendo bits de datos de video gradualmente hacia los *video players* de los usuarios que pueden estar embebidos en una página web. Una ventaja es que puede ser utilizado por los *smart phones* y permite que el video no ocupe espacio en el disco duro del cliente al no ser almacenado en su carpeta temporal (CISCO, 2009). El incremento y la influencia de la popularidad de iPhone han contribuido a la desaparición del estándar de Adobe y el aumento de la transmisión HTTP, ya que Steve Jobs insistió en que no sería compatible con *Flash*.

2.3.2.2 RTP *(Real Time Protocol)*

Es el protocolo nativo de Internet diseñado para llevar datos en tiempo real sobre redes IP y es recomendado utilizarlo cuando se maneja un control plano mediante el protocolo RTSP (*Real Time Streaming Protocol*). Además, maneja cada flujo de *stream* por separado

mediante la encapsulación sobre el protocolo UDP (*User Data Protocol*) que permite una mayor velocidad de transmisión de paquetes gracias a su menor cantidad de cabeceras. Finalmente se tiene que la sincronización viaja en cada *stream* de tal manera que, por parte del lado del cliente se pueda sincronizar el *stream* cuando este sea necesario (Ponce, 2013).

En los primeros días de vídeo o Internet, RTSP (*Real Time Streaming Protocol*) se alió con RTP (Protocolo de Tiempo Real) surgiendo como el protocolo más común para la transmisión de vídeo de los sitios web y que funcionó razonablemente bien, explotando el almacenamiento en caché para apoyar la distribución de fragmentos para mejorar la fiabilidad y el rendimiento. A veces se utilizan varias velocidades de bits para optimizar la experiencia del usuario para el dispositivo receptor. Para el protocolo tradicional RTSP basado *streaming*, los cachés realmente no ayudan. Pero RTSP/RTP resultó tener algunas fallas graves, algunas de las cuales fueron identificadas primero por Apple, ya que desarrolló su protocolo de *streaming*. Uno de los temas principales fue que RTSP tiende a ser bloqueado por los routers o configuraciones de firewall, previniendo que un dispositivo acceda al *stream*.

Tabla 3.

Comparación entre el protocolo RTMP y HTTP (2013)

HTTP vs. RTMP – Quick Reference		
	HTTP	**RTMP**
Server Components	Open	Specialized
Security/IP Protection	Basic	Enhanced
Caching/Scale	Native	Specialized
Player Interactivity	Basic	Extensive
Multicast Support	N/A	Supported
Plug-in Penetration	Nascent	Extensive
Firewall (port/protocol)	No restrictions	Some restrictions
Variable Bitrate Support	No impact	Susceptible to data spikes

2.3.2.3 HTTP *(Hypertext Transfer Protocol)*

Por otro lado HTTP, habiendo sido desarrollado específicamente como el protocolo estándar para la Web, es generalmente accesible. Es un protocolo usado para manejar la entrega de imágenes y páginas web, se lo emplea también para mandar *pseudo-streamings* de video a través del Internet. Se tiene que es soportado por la mayoría de servidores web, ya sean propietarios o de licencia libre. Una ventaja que ofrece está en el acceso ya que utiliza el puerto 80 y con esto permite saltarse reglas del firewall en el bloqueo de puertos debido a que este es el que se utiliza para la salida y conexión a Internet, por lo que no puede estar bloqueado (Macaulay et al, 2005).

HTTP tiene otras ventajas, como exigir sólo un servidor HTTP estándar, y la participación de expertos y una mayor sencillez en el desarrollo del lado del cliente. HTTP es también un estándar abierto a diferencia de *Flash* vídeo, que se le dio un lugar sobre el cual domina cuando RTSP muestra sus limitaciones. Pero ahora incluso Adobe reconoce que HTTP es un importante paso adelante y después de haber desarrollado su protocolo HTTP *Dynamic Streaming* propia, Adobe está firmemente horneando la iniciativa de la industria DASH. En este contexto *HTTP Live Streaming* ha alcanzado gran popularidad entre los proveedores de CDN. MPEG DASH se perfila como un fuerte candidato para unificar las normas con el apoyo de Microsoft y Adobe. La actitud de Apple todavía no está claro, pero parece probable que su propio estándar HLS seguirá siendo por un tiempo.

Todos los contextos de la transmisión HTTP implican software de servidor que divide el flujo de transporte MPEG en trozos pequeños guardados como archivos separados, junto con una lista de reproducción llamada un archivo de manifiesto, para decirle al software cliente de reproducción multimedia dónde obtener los archivos que componen el *streaming* completo. El cliente del reproductor multimedia simplemente descarga y reproduce los pequeños trozos en el orden especificado en la lista de reproducción. En el caso de una transmisión en vivo, el cliente actualiza periódicamente la lista de reproducción para ver si ha habido algún nuevo trozo añadido al *stream*. El protocolo también es compatible con ABR y cambia automáticamente a la velocidad de bits óptima de acuerdo a las condiciones de red

que prevalecerán para una experiencia de reproducción de la calidad (fig. 30). De esta manera el usuario final siempre experimenta, al menos en teoría, la mejor calidad posible que la red y el dispositivo en conjunto pueden producir, sujeto a variar dependiendo de las condiciones de ancho de banda.

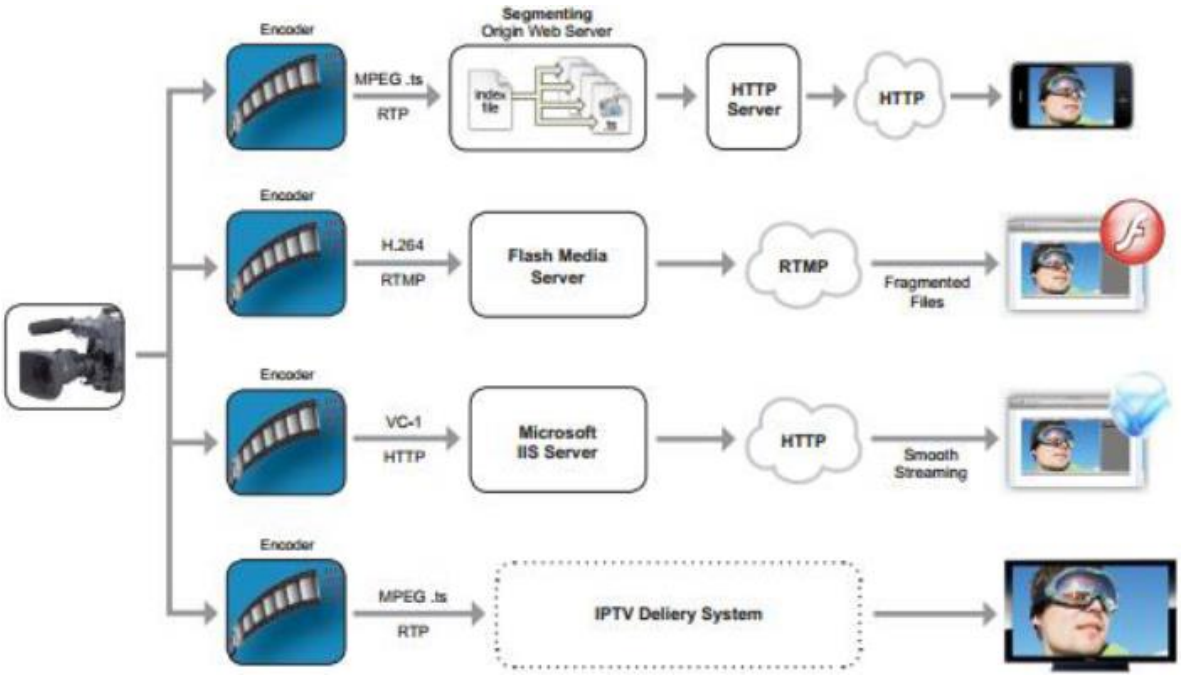


Figura 30. Entrega de Video a Múltiples Plataformas. (2013) (Fuente: Ponce, G)

2.3.2.4 *Adaptive Bitrate* (ABR)

El más reciente, que es la tecnología de *streaming Adaptive Bitrate* (ABR) basado en HTTP (ABR), algunos otros como Microsoft *Smooth*, Apple HLS, y Adobe HDS, se han utilizado mucho en la industria de la CDN para la transmisión en vivo. *Adaptable Bit-Rate* ha traído una nueva vida para el almacenamiento en caché y CDN. *Adaptable Bit-Rate* (ABR) ha revitalizado el almacenamiento en caché y pone de nuevo en el centro del escenario la calidad de la experiencia *Quality of Experience*. Permite a un CDN la entregar de media *streaming*,

incluyendo contenido en vivo, escalable al replicar el contenido a través de múltiples *Edge cache servers*. Un usuario final que solicita el *stream* es redirigido entonces al *Edge server* "más cercano".

También hay una importante ventaja de coste. El uso de *HTTP adaptive streaming* permite que el servidor *Edge* para ejecutar software de servidor HTTP sencillo, cuya licencia de costos es libre o de bajo costo, en comparación con las costosas licencias de servidor de medios de comunicación para el *Adobe Flash Media Streaming Server* (Pathan et al, 2014). El costo de CDN para los medios de *streaming* HTTP han llegado a ser similares al tradicional almacenamiento HTTP *web caching*. ABR *streaming* es una tecnología de *multibitrate video streaming* que permite al reproductor de medios del cliente a que ajuste dinámicamente la calidad del flujo de vídeo, dependiendo de las condiciones de la red imperantes durante la sesión de transmisión.

Para asegurar la mejor experiencia del espectador sea posible, el cliente reproductor multimedia de un usuario final es capaz de seleccionar una tasa de bits de transmisión que mejor se adapte a las condiciones de la red en el momento, y puede cambiar dinámicamente la velocidad de bits durante la sesión si las condiciones de red mejoran o se deterioran, sin terminación o *rebuffering* de la presentación del video.

2.3.3 Algoritmos de Compresión

Se llama algoritmo de compresión, a un conjunto de procedimientos secuenciales que permiten representar un flujo de información con una menor cantidad de ésta. Realmente la compresión consiste en sustituir una cadena de datos por otra más corta cuando se guarda un archivo. Ciertos métodos son los reversibles y que son conocidos como "*lossless data compression*", porque permiten la reconstrucción exacta del original (Olivé, 2008). Pero por otro lado, están los métodos que permiten que la información original sólo se recupera aproximadamente, ya que se descarta una parte de los datos a cambio de relaciones de compresión mucho mayor que este y son conocidos como "*lossy data compression*".

- **Compresión sin Pérdidas (*Lossless*)**

Esta compresión permite recuperar exactamente la calidad original de la imagen. Mayormente es utilizada para comprimir archivos o información que contienen datos que no pueden ser degradados o perdidos, como pueden ser documentos de texto, archivos ejecutables, etc. Según se tengan en cuenta o no las características del archivo a comprimir, esta compresión se distingue entre sistemas adaptativos, no adaptativos y semiadaptativos (Olivé, 2008). Como ejemplos de agoritmos *Lossless*, se tienen a: *Flac-audio, PNG-imagen, TIFF-imagen, Huffyuv-video, MSU-video* y *LCL-video*.

- **Compresión con Pérdidas (*Lossy*)**

Esta compresión con pérdidas se define al tipo de compresión que degrada en mayor o menor medida la calidad de la imagen. Dentro de esta categoría es universalmente conocido por su eficacia el formato JPEG. Por otro lado, los efectos negativos de este método compresión son el empobrecimiento del tono y la nitidez global, que notaríamos más bien en una impresión, y la aparición de artefactos a nivel local visibles sobre todo en pantalla, aunque JPEG sea un formato habitual en Internet (Olivé, 2008). Como ejemplos de agoritmos *Lossy*, se tienen a: *Fractal compression-imagen, JPEG-imagen, JPEG 2000-imagen, Flash-video, H.261-video, H.263-video, H.264/MPEG-4/AVC-video, Motion JPEG-video* y *MPEG 1, 2 ,4-video.*

2.3.4 Técnicas de Compresión

Los principales objetivos que buscan las diferentes técnicas de compresión son: Reducir lo mayor posible la información a transmitir, reducir los costos de conexión y reducir el tiempo de enlace o demora. Sin embargo, hay que considerar el código de compresión debe ser lo más compacto posible que el original y eliminar toda o casi toda la redundancia existente en la imagen original (Olivé, 2008). Por esta razón se tienen 2 principales técnicas de compresión, que son:

- **Compresión Espacial**

La compresión se hace sobre la información contenida en un *frame*, y no se relaciona con los otros *frames* de la secuencia. La redundancia espacial tiene lugar dentro de cada fotograma. Este proceso es ocupado en formatos de imágenes como, JPEG, GIF y TIFF. Esta técnica viene asociada al hecho de que la naturaleza y todo lo que la rodea está llena de objetos sólidos con superficies y texturas uniformes; lo que permite entender que las estructuras, los paisajes, e incluso los rostros no varían significativamente la información en termino de pixel, sin embargo por lo general encontramos grandes superficies sin variación (EDII, 2005). El estandar de video digital "Motion JPEG" es un ejemplo de compresión espacial de video.

- **Compresión Temporal**

La relación entre los píxeles homólogos de imágenes sucesivas, estará siendo deternimada por la redundancia temporal. Todo esto permite a que esta dimensión temporal busque información redundante a través de la secuencia de *frames*, conociendo que la redundancia aparece porque la vida no cambia significativamente desde un fotograma a otro fotograma y su funcionamiento consiste en tomar como referencia un *frame* base y almacenar sus diferencias con los *frames* dentro de la misma secuencia, sin tomar en cuenta la información redundante. Por ejemplo, suponiendo una frecuencia de cuadro de 25 f/s y dentro de un espacio de 40m, no se evidencian que ocurran cambios significativos en las cosas desde el punto de vista del espectador (EDII, 2005). Lo que el espectador espera como resultado al contemplar una secuencia de vídeo estándar, es una continuidad en la acción y no un cambio continuo en los planos, definiendo el frame de partida, el resto de los frames se calcula a partir de las diferencias. Este tipo de compresión es siempre con pérdidas (lossy).

Se conoce que la codificación diferencial de pulsos modulados (*Differential Pulse-Code Modulation* - DPCM), es una técnica de compresión que se puede usar, tanto para eliminar la redundancia temporal, como la espacial y es la cual codifica el valor de diferencia de una muestra con respecto a la anterior.

2.3.5 Tendencias de Compresión

Se debe considerar ciertos elementos básicos en la tecnología de *broadcast* y video, los estándares técnicos son definidos por la SMPTE (*Society of Motion Picture & Television Engineers*), existen 2 interfaces para el video digital en Banda Base, que son: Interfaz SD-SDI a 270 Mbps e Interfaz HD-SDI a 1.5 Gbps ó 3 Gbps, comprendiendo que ambos pueden llevar embebido audio (Level 3, 2012). Los rangos de compresión dependen de varios factores, como el Formato (SD vs HD), las Tecnologías de compresión sobre las cuales se emplearán técnicas y codecs de compresión para la resolución como los más importantes que son: MPEG-2, MPEG-4 y JPEG 2000 (fig. 31); así como entender el sistema de muestreo que involucra la *Luminancia* (brillo) y *Crominancia* (color): 4:2:2 vs 4:2:0.

Al momento de transportar el contenido comprimido, el estándar DVB-ASI se usa para llevar señales de video comprimido, sabiendo que se pueden multiplexar varios *streams* de video en una señal ASI y que es fácilmente encapsulable en transmisiones sobre IP.

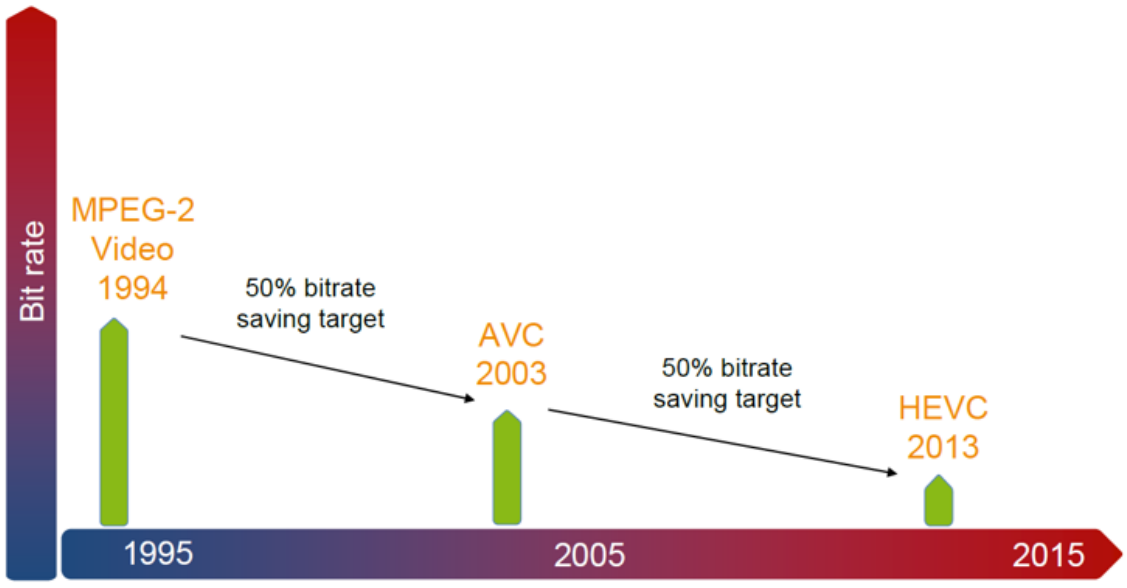


Figura 31. Ahorro de Bit rate vs. Evolución. (2014) (Fuente: Level 3.com)

2.3.6 Codecs para Streaming

Los codecs se describen como un tipo de especificación desarrollada en software, hardware o de ser el caso una combinación, capaz de transformar un archivo con un flujo de datos (*stream*) o una señal de una fuente, con componentes de *Encoder* y *Decoder* (Torres, 2009). Son los encargados de que se pueda enviar video a través de Internet, ya que permiten a través de sus técnicas de compresión disminuir el tamaño del espacio ocupado comprimiendo y descomprimiendo video digital, por esto hay que recalcar que son la base en el mercado del *streaming* de contenido, debido a las facilidades que presta sus servicios para que el contenido pueda ser enviado, visualizado a través de la red y/o almacenarlo en algún repositorio. Comunmente dichos algoritmos de compresión o codecs son de 2 tipos, los que conllevan una pérdida de información durante la transformación, con el propósito en base a la calidad de imagen y cantidad de espacio ocupado en una unidad de almacenamiento, o el tipo de códec que no presenta perdidas. Como agregado del codec se tiene un contenedor, como por ejemplo MP4, MOV, AVI, que provee información de que codec de video y audio que se utilizó (SyT. Códec, 2010). Se describirán algunos de los estándares de compresión más usados para video de alta resolución, como los son JPEG y MPEG.

2.3.6.1 MPEG

MPEG está dedicado a las secuencias de vídeo, donde su grupo de estándares MPEG incluye a los formatos MPEG-1, MPEG-2 y MPEG-4.

MPEG-2 es la versión mejorada de MPEG-1 la cual define dos tipos de formato de contenedor: flujo de transporte y flujo de programa (Torres, 2009). Al momento de transmitir video y audio por canales se utiliza el primer contenedor. Introduce la posibilidad de la codificación interlineada, *broadcast* TV, DVD, 2-15 Mbps (vídeo+audio), con basa en *Discrete Cosine Transform* (DCT), pero con compresión basada en correlación temporal y encapsulado (*transport stream* TS) para enviar el vídeo sobre la red, agrega la posibilidad de

codificar más de 2 canales de audio utilizando codificación MPEG-2 part 7 *advanced audio codec* (AAC) (UPNA, 2009). Se puede decir que tiene como desventaja un *bitrate* elevado frente a nuevos y más eficientes *codecs* que se verán durante esta etapa (fig. 32).

	MPEG-1	MPEG-2	MPEG-4
Aprobación	Nov-91	Nov-94	Oct-98
Calidad	VHS	DVD	Calidad Escalable
APLICACIONES	Video CD	TV Digital (HDTV)	Basado en el formato QuickTime
	CD-ROM	Digital Versatile Disk (DVD)	Codificación de datos multimedia, adaptable - desde teléfonos celulares hasta televisión por satélite
	Audio mp3	Canales de sonido envolvente	Adecuado para la WEB
Máx. ratio de bits (Mbps)	1.86	15	15
Ancho de imagen (píxeles)	352	720	720
Alto de imagen (píxeles)	288	576	576
Ratio de imágenes (fps)	30	30	30

Figura 32. Standard MPEG. (2014) (Fuente: Level 3.com)

2.3.6.2 H.264

Es conocido también como *Advanced Video Coding for generic audiovisual services* (AVC) del estándar MPG-4 y técnicamente idéntico al codec MPEG-4 parte 10. El objetivo era conseguir que sea mayor o en tal caso la misma calidad que el *codec* de vídeo de MPEG-2 a uno menor, mitad o un tercio del *bitrate* que sus formatos predecesores. Se lo utiliza con interoperabilidad en formatos de disco HD-DVD y Blu-ray, como también en transmisiones

de alta definición bajo redes CDN (Torres, 2009). A pesar de su gran dominio dentro del mercado de video, en el mes de Mazo del 2013, fue estandarizado su sucesor, el H.265, por la ITU el cual permitirá ser 50% más eficiente que el H.264 cuando se comercialice.

Tabla 4.

Resolución y escenarios. (Fuente: UPNA) (2009)

Use Scenario	Resolution & Frame Rate	Example Data Rates
Mobile Content	176x144, 10-15 fps	50-60 Kbps
Internet/Standard Definition	640x480, 24 fps	1-2 Mbps
High Definition	1280x720, 24p	5-6 Mbps
Full High Definition	1920x1080, 24p	7-8 Mbps

2.3.6.3 JPEG

JPEG está asociado a imágenes digitales estáticas. Entre los formatos de imágenes tradicionales, están: JPEG y JPEG 2000. Existen variantes que resultan apropiadas para vídeo: Motion JPEG y Motion JPEG 2000.

2.3.6.4 Motion JPEG

Una secuencia de video puede ser representada como una serie de imágenes JPEG, logrando la ventaja de ofrecer una alta calidad de la imagen, pero limitándolo, con la desventaja de que sólo utiliza una serie de imágenes estáticas sin hacer uso de técnicas de compresión de vídeo (JPEG, 2010). El resultado es un ratio de compresión inferior para secuencias de vídeo en comparación con las técnicas de compresión de vídeo temporales.

2.3.6.5 JPEG 2000

Es un sistema de codificación de imágenes que utiliza un estado del arte de las técnicas de compresión basadas en la tecnología de ondas. También se incluye soporte para audio asociado. Su arquitectura se presta a una amplia gama de usos, desde el uso de cámaras digitales portátiles hasta usos en sectores profesionales como el de *pre-press*, para imágenes médicas y otros. JPEG 2000 se refiere a todas las partes del standard (JPEG, 2010). A continuación se muestra la lista de las piezas actuales que conforman la suite completa JPEG 2000 de las normas:

JPEG2000 ofrece ratios de compresión más altos (aproximadamente 25% menos en requerimientos de AB que JPEG para similares calidades de imagen). JPEG2000 permite una completa compresión sin pérdidas.

2.3.6.6 Motion JPEG2000

MJ2 no implica codificación entre cuadros: cada trama se codifica de forma independiente mediante JPEG 2000. Aplicaciones esperadas incluyen: Almacenar clips de vídeo tomadas con cámaras digitales, grabación de vídeo basado en cuadros de alta calidad y edición, cine digital, imaginería médica y satelital (JPEG, 2010). Esto hace que sea apto para su almacenamiento y compresión de imágenes cuando la retención fidelidad es importante. Todas las imágenes guardadas en el formato JPEG2000 sin pérdidas serán guardadas principalmente como archivos .JPF.

Tabla 5.

Typical Compression Rates Used for Different Applications (Fuente: Level 3.com) (2014)

Compresión	Formato	Aplicación	Velocidad de Compresión
MPEG-2	SD	Contribución de Noticias	8 Mbps
	HD	Contribución de Deportes	80 Mbps
MPEG-4	SD	Distribución de Programas	12 Mbps
	HD	Contribución de Deportes	40 Mbps
JPEG-2000	HD	Contribución de Deportes	150 Mbps

2.3.7 Modos de Videostreaming

Existen 2 formas de transmisión de video a través de las redes: *Unicast* y *Multicast* (fig. 33):

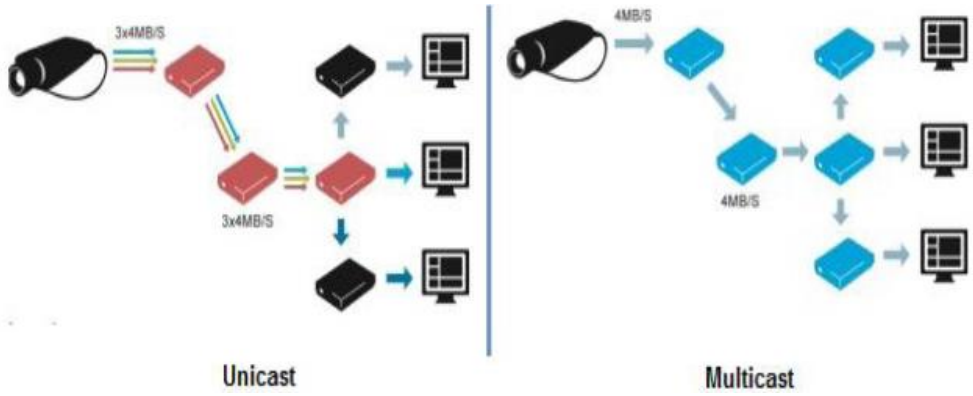


Figura 33. Diagramas de Streaming en unicast y multicast. (Fuente: Ponce, G)

- ***Unicast***

Es definido como la forma de comunicación uno a uno entre la fuente y el destino, su arquitectura centralizada proporciona escalabilidad limitada, esto quiere decir que si se tiene un determinado servidor de *streaming* este tiene que ser capaz de soportar tantas conexiones

como usuarios soliciten el contenido o video (1M usuarios → 200K/día @ 2Mb/s & 20%) y es soportado tanto en el protocolo de transporte TCP o UDP. Un ejemplo de equivalencia de transmisón *Unicast* es el *video on Demand* (VoD). La mayoría de redes comerciales CDN utilizan este modo ya que los routers de Internet aún no soportan multicast (GENETEC, 2010).

- ***Multicast***

En este modo no existe conexión directa entre la fuente y los destinos, ya que se crean grupos *multicast* que manejan un determinado *stream* con lo cual los destinos acceden a un determinado contenido registrándose en el grupo *multicast*, identificado por una dirección independiente de la localización, por esta razón la necesidad de contar con red una mejorada para ofrecer un adecuado *multicast* y QoS. Lo que se logra es realizar tantas copias como usuarios se tenga en el grupo y así ahorrar ancho de banda ya que se maneja a nivel IP, fácil de utilizarlo dentro de LANs que soportan *broadcast* y complicado de ser utilizado al atravesar redes (GENETEC, 2010). Es muy efectivo en el ahorro de ancho de banda, sin embargo es necesario crear una red privada para su uso y por lo tanto no es posible implementarlo si es que los clientes o la red pública aún no lo soportan. Por esta razón el *multicast* es ampliamente utilizado para la difusión por TV (fig. 34), ya que dada su escalabilidad sobre un gran número de canales y un número ilimitado de usuarios a entregar el servicio, este emplea un óptimo uso del ancho de banda (UPNA, 2009).

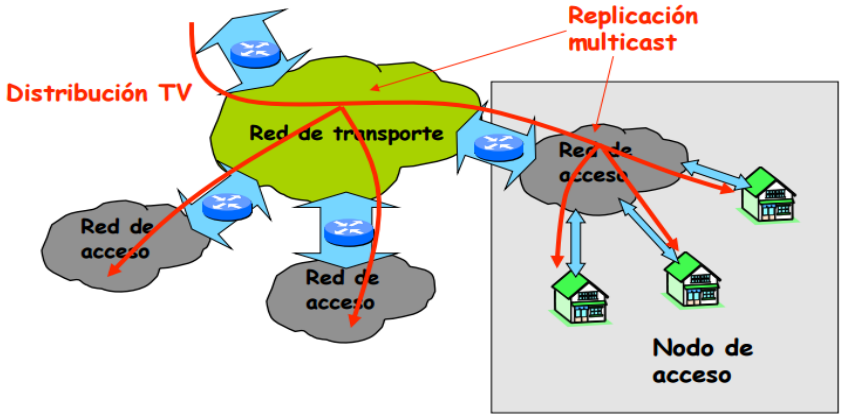


Figura 34. Multicast para difusión de TV. (2009) (Fuente: UPNA)

2.3.8 Transmisión Bajo Demanda (*On-Demand Streaming*)

El *streaming on-demand* permite a los usuarios finales seleccionar, ver y/o escuchar contenido de vídeo o audio bajo demanda, por ejemplo, *video on Demand* VoD, audio y vídeo *on Demand* (AVOD) y el *suscription video on demand* (SVOD). Tecnología IPTV se utiliza a menudo para llevar los servicios de VoD a los televisores y computadoras personales. Los sistemas de Televisión VoD realizan un *stream* de contenido a través de un decodificador o *set-top box*, un ordenador u otro dispositivo ofreciendo su visualización en tiempo real, teniéndolo disponible para descargarlo a otro dispositivo o computador, tenerlo disponible en un *personal video recorder* (PVR) o simplemente verlo en cualquier momento a través de un reproductor multimedia portátil (Pathan et al, 2014).

La mayoría de los proveedores de cable y por televisión de paga, ofrecen ambos métodos de disponibilidad de contenido *VoD streaming*, que incluyen contenido gratuito o *pay-per-view*. El usuario puede comenzar a reproducir el contenido de vídeo en el aparato de televisión de forma casi instantánea, descargarlo a un *digital video recorder* (DVR) o finalmente descargarlo a un PC para visualizarlo en el futuro. La IPTV cada vez más se convierte en una popular forma de *VoD service*, ya que a parte de todas sus ventajas, este permite también la transmisión de múltiples canales lineales con una EPG, lo que permite al espectador mirar toda la programación que viene más adelante. Esto presenta una experiencia tipo televisión digital en cualquier dispositivo IP conectado.

CLOUD COMPUTING

2.4 INTRODUCCIÓN

La computación como la conocemos ha evolucionado significativamente en los últimos 60 años. Durante sus inicios, la carga computacional de toda una empresa la manejaba un solo computador central de grandes características (*mainframe*). Esta estructura fue desarrollándose gradualmente a lo que se conocía como los equipos departamentales en la década de 1970, hasta las épocas de los años 1980 y 1990 donde aparecieron las computadoras personales. Aunque la computación en nube (*Cloud Computing*) es un término prácticamente nuevo, su fundamento como concepto inicial se predijo en los años 1960 por algunos científicos informáticos como *John McCarthy* y el Ingeniero canadiense *Douglass Parkhill* en 1966. McCarthy afirmó: "La Computación podría estar algún día organizada como un servicio público" (Halpert, 2011) y por otro lado Parkhill que publicó su libro: *"The Challenge of the Computer Utility*" (Parkhill, 1996), en el que describe la idea de la computación como un servicio público, formado por una instalación de computación centralizada a la que muchos usuarios remotos se conectan a través de las redes computacionales.

Con el cambio en la industria de semiconductores (o potencia de cálculo), predicha por el cofundador de Intel, Gordon E. Moore, las computadoras personales se hicieron asequibles, y las empresas fueron abandonando la infraestructura de los mainframes. A continuación, un nuevo desafío se introdujo: El de cómo compartir los datos ?.

Se suponía que los sistemas cliente-servidor harían frente al desafío del intercambio masivo de datos, proporcionando servidores centralizados de gestión de datos y procesamiento. Como las necesidades informáticas de las empresas crecían y la Internet llegó a ser ampliamente adaptada, lo que en un principio era una sencilla arquitectura cliente-servidor, se transformaba en algo mucho más complejo como los sistemas de arquitecturas de *two-tier*, *three-tie*, y *four-tier* (Gorelik, 2013). Como resultado, los costos de la

complejidad y de gestión de la infraestructura de IT se incrementaron exponencialmente, incluso los costos del desarrollo de software actual dentro de grandes organizaciones son generalmente más bajos que los costos de software y mantenimiento de la infraestructura.

En 2007 varias organizaciones importantes como: Google, IBM, *Carnegie Mellon*, el MIT, la Universidad de Stanford, la Universidad de Berkeley, la Universidad de Maryland, y la Universidad de Washington, colaboraron para iniciar la investigación sobre la conceptualización del *Cloud Computing*. Al poco tiempo después, muchos grupos de analistas comenzaron a informar sobre la significativa compartición de mercado que se estaba estableciendo por el *Cloud Computing* (Halpert, 2011). Mientras que su regularización se está dando a través de varias organizaciones y consorcios de estandarización como el *Open Group*, OASIS, y DMTF. Aunque el efecto de la adopción de *Cloud Computing* aún no se ha visto, muchas empresas creen que la Computación en Nube puede ofrecer un modelo alternativo viable que puede reducir costos y complejidad al mismo tiempo que aumenta la eficiencia operativa.

2.4.1 Definiciones

La computación en la nube (*Cloud Computing*) es un paradigma de la computación y en el entorno existen un sinnúmero de definiciones e interpretaciones acerca de esta y que se encuentran en múltiples fuentes. El término *Cloud Computing* como tal, probablemente proviene de esquemas de red que basaron su estructuración con un estilo de servicio en la nube y que se utilizaron para describir ciertos tipos de redes, ya sea por Internet o redes internas. Algunas fuentes se refieren al *Cloud Computing* como un conjunto de aplicaciones distribuidas como un servicio, que habilitan las aplicaciones al ser combinadas con el hardware y software de un *Datacenter*. Otros dicen que la computación en la nube es un modelo de negocio en lugar de una tecnología o servicio específico.

Si bien no existe una única definición de este término, se suele hacer referencia a lo expuesto por el *National Institute of Standards and Technology* (NIST) (Mell & Grance,

2011). “El *Cloud Computing* es un modelo para permitir un acceso conveniente y bajo demanda a la red de un conjunto compartido de recursos informáticos configurables (por ejemplo, redes, servidores, almacenamiento, aplicaciones y servicios) que pueden ser rápidamente aprovisionados a un usuario o sistema y liberados con mínimo esfuerzo de administración o interacción por parte del proveedor de servicios”. El objetivo principal es permitir a las organizaciones TIC alcanzar una mejora sustancial en la provisión elástica de servicios, tanto en términos de coste como en la propia gestión del servicio (Mendoza, 2007). Esta mejora sustancial en aprovisionamiento inmediato de servicios, cobra un mayor significado en gestiones críticas (como por ejemplo, una gestión de emergencia), donde la disponibilidad debe ser prácticamente del 100%.

Se podría decir que el *Cloud Computing* consiste en ambos componentes tanto tecnológicos como de negocios (fig. 35). Ciertas tecnologías de tipo *cloud-enabling* como el software de código abierto (*open-source*), la virtualización, el almacenamiento distribuido, las bases de datos distribuidas y los sistemas de vigilancia, son la base de la infraestructura *Cloud* y que ayudaron considerable a formar lo que hoy se conoce como el servicio en la nube (Gorelik, 2013).

La computación en nube asume que cada aplicación de software o componente del sistema se convierte en un servicio o parte de un servicio. Por lo tanto, la arquitectura de la mayoría de sistemas nuevos o existentes, tendrían que empezar a adaptarse para ser compatibles con un entorno en la nube, así como los usuarios deben cambiar de paradigma y adaptarse a nuevos conceptos o mecanismos de servicio. Es por esto que el modelo en la nube (*Cloud*) se compone de cinco características esenciales, tres modelos de servicio y cuatro modelos de despliegue o distribución.

Figura 35. Conceptos de Cloud Computing. (Fuente: http://www.comusoft.com/aspectos-juridicos-del-cloud-computing)

Varios grupos le han entregado una definición adecuada, como por ejemplo el del grupo *Yankee Group*, donde define al *Cloud Computing* como "un servicio de información dinámicamente escalables, virtualizados y entregados bajo demanda a través de Internet con capacidad *multi-tenant* y acuerdos de nivel de servicio o SLAs, con la fijación de precios basada en el uso"

El grupo *Enterprise Strategy Group* define al *Cloud Computing* como "un modelo de servicio, en el cual la carga de trabajo es desarrollada y ligeramente ejecutada, para lo cual se expande el negocio de paga lo que consumes" (Skillsoft, 2011).

El grupo *Burton Group* define al *Cloud Computing* como "el set de disciplinas, tecnologías y modelos de negocio, utilizados para hacer rendir las capacidades de un servicio *on-demand*".

Esencialmente el *Cloud Computing* transfiere tareas computacionales a la Internet y es el hecho que deriva su nombre del uso común de la imagen de una nube, para representar la red distribuida en Internet (fig. 36).

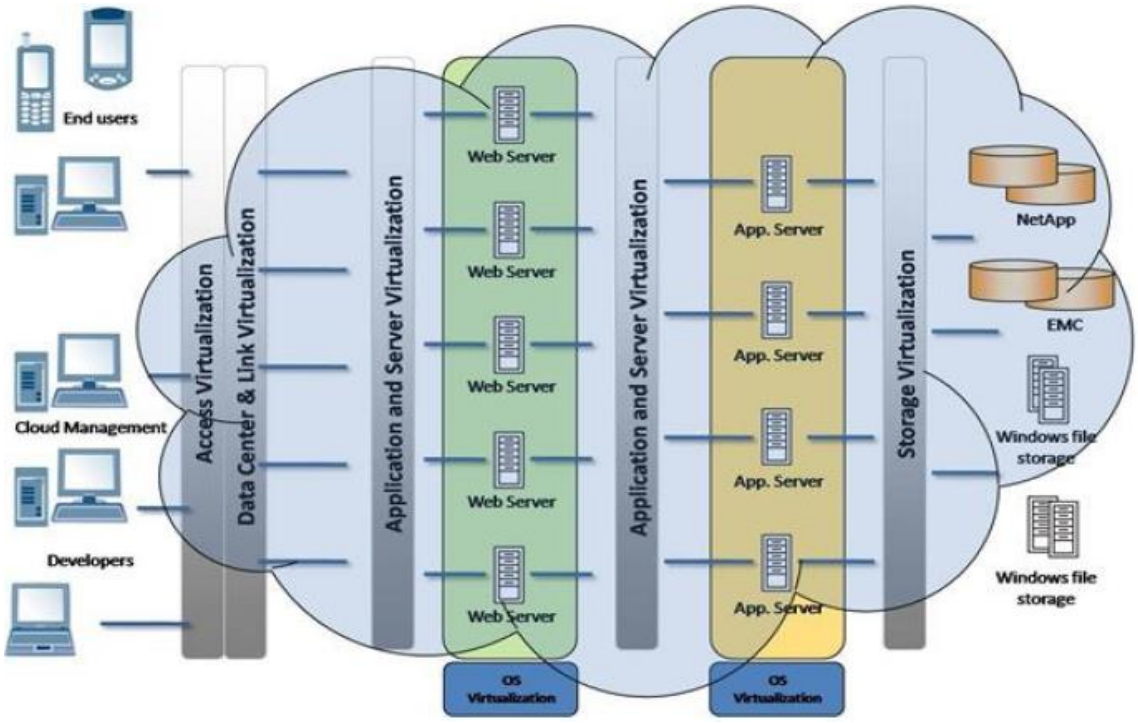


Figura 36. Conceptualización general de Cloud Computing y servicios (Fuente: Hernansanz J.)

2.4.2 Tecnologías Similares.

Las tendencias hacia una computación en la nube comenzaron a finales de 1980 con los primeros conceptos de *Grid Computing*, donde por primera vez una gran cantidad de sistemas se aplicaron a un solo problema. Luego en la década de 1990 los conceptos de virtualización se ampliaron más allá de los servidores virtuales a niveles más altos de abstracción; en primer lugar están las plataformas virtuales y en segundo lugar la aplicación virtual, dichas aplicaciones se encontraban embebidas dentro de la infraestructura de los *Datacenters*,

siendo estas escalables y con recursos asignados dinámicamente. Lo que ha permitido que surjan modelos emergentes donde los usuarios puedan acceder a las aplicaciones desde cualquier lugar, logrando constituir el modelo de *Cloud Computing* el cual no solo es práctico, sino que también permitió la reducción de la complejidad de TI (Quan, 2008).

Teniendo en cuenta una carga de trabajo en particular y conociendo las diferentes formas de administrar los recursos, las personas se preguntan cuál es la diferencia entre *Cloud Computing* y las tecnologías similares. *Utility Computing* proporciono un clúster de plataformas virtuales para la computación. A partir de los conceptos que se muestran a continuación, como: *Grid, Servicios Web-based, Utility Computin*g y el SaaS propio del *Cloud Computing*, el concepto de computación en la nube evolucionó.

2.4.2.1 Servicios Web-based

Existen algunos tipos de servicios *web-based* como el servicio de mail por internet, el alojamiento de una página web (*hosting*) o las tiendas online, han estado en el mercado ya por varios años y su uso se ha vuelto casi generalizado, sin embargo, el *cloud computing* le ha provisto a este tipo de servicios nuevas oportunidades, en cómo estas aplicaciones serán alojadas (*hosted*) y administradas con muchas diferencia a cómo se las hacia anteriormente (Skillsoft, 2011). Cuando se habla sobre los servicios basados en la web (*Web-based*), se sabe que estos soportan *Cloud Computing* y hacen uso de algunas de las mismas tecnologías, pero hay que tener en cuenta que no todas las funcionalidades o aplicaciones *web-based* utilizan *Cloud Computing* para logra su objetivo. Por ejemplo, se puede acceder a una aplicación *web-based* a través de la Internet, pero esta puede estar alojada en un tradicional *datacenter* empresarial.

2.4.2.2 *Cloud vs. Grid Computing (Distributed Computing)*

Muchos expertos argumentan que el *Cloud Computing* viene del *Grid Computing* y del *Parallel Computing*. Sin embargo, aunque hay muchas similitudes entre el *Cloud y Grid Computing*, sus metodologías no son las mismas. La principal diferencia de las redes *grids* es que no fueron creadas originalmente como un servicio de computación pública bajo demanda, si no que se las utiliza típicamente dentro de la misma organización para ejecutar arduas tareas computacionales, dividiendo un carga de procesamiento entre múltiples computadoras, ya teniendo implementado toda una infraestructura completa en la misma sede o dicha infraestructura puede estar dispersa geográficamente y ser totalmente heterogenia, siendo esta la principal diferencia también ante un *Cluster* tradicional. Mientras que el *Cloud Computing*, está asociado con un servicio específico y dicho servicio es utilizado como punto de acceso, al proveer resultados a los servicios que se entrega al consumidor u otra aplicación, como por ejemplo una aplicación *Business-to-Business* (B2B) (Gorelik, 2013).

Por esta razón las redes de *Cloud Computing* tienen una gran ventaja, ya que puede escalar bajo pedido. La nube ofrece más elasticidad, de tal manera que un ambiente de *Cloud* puede partir de tan solo un poco de infraestructura y crecer rápidamente a cientos de servidores o luego si se tiene la necesidad o se requiere ahorro, se adapta para retornar hasta el tamaño inicial.

2.4.*2.3* *Clustering*

El término se aplica a los conjuntos o conglomerados de computadoras construidas mediante la utilización de componentes de hardware comunes, trabajando en paralelo y que se comportan como si fueran una sola computadora, ya que comparten recursos para realizar tareas de cómputo complejas. Se clasifica en *Clusters* de: Alto Rendimiento, Alta Disponibilidad y Balanceo de Carga.

2.4.*2.4* *Datacenters*

Conocidos en nuestro idioma como centros de datos, los *Datacenters* son ambientes especializados en donde los datos son almacenados, tratados y distribuidos al personal o son puestos bajo procesos autorizados para consultarlos y/o modificarlos (Revelo, 2013). Los servicios de *Cloud Computing* aprovechan las ventajas que tiene la utilización de recursos de hardware agrupados como uno solo, ya que automatiza los servicios y generalmente involucra un gran acuerdo de virtualización. Los servidores que componen el *Datacenter* salvaguardan el activo más valioso de una empresa que es la información, estos mantiene albergados los datos en un entorno de funcionamiento óptimo y es por esta razón que, aunque el *Cloud Computing* requiera siempre del hardware de un *Datacenter*, el concepto de múltiples servidores en un solo *Datacenter* no califica como *Cloud*. Por esta razón dentro de *Cloud Computing*, siempre se empleará el hardware alojado en múltiples *Datacenters*, pero la localización física de los servidores será irrelevante y transparente para el usuario. A través de la tecnología de virtualización, ahora es posible ejecutar no sólo una aplicación, sino también un servidor completo dentro de la nube, reduciendo así el costo de la base de hardware en un modelo de servidor de un servicio típico, donde el hardware es a menudo infrautilizada (Skillsoft, 2011).

El tipo de *Datacenter* que despliega el hardware y software necesarios es la esencia del servicio en la nube, esto debido a que el *Cloud Computing* ofrece una amplia gama de servicios, incluyendo los procesos computacionales (*computing*), el almacenamiento (*storage*) y la entrega (*delivery*), a más de los diferentes modos de prestación de servicios, como son, *Software / Plataforma / Infraestructura* como servicio (S/P/IaaS) (Christodoulou et al, 2012). Se debe entender que un solo servicio o aplicación puede estar alojado o su procesamiento puede estar siendo operado por un número indefinido de servidores físicos dentro de los *Datacenters* con funcionalidad para servicios en la nube, en vez de que un servidor específico o un *cluster* de servidores sirva a la operación, como funcionaba en los *datacenters* tradicionales.

2.4.*2.5* *Utility Computing*

Se refiere a como los recursos son administrados, consumidos y utilizados, teniendo en cuenta las cantidades adecuadas y en el momento adecuado para implementarlos. Se podría decir que es algo simple, al entender que al momento de implementarlo se transforma en una especie de sueño utópico para brinda servicios en las organizaciones de TI, donde sus costos de fiabilidad y rendimiento se reducen, al mismo tiempo que su eficiencia se incrementa de manera responsable y confiable (Revelo, 2013). Siendo este un lineamiento anhelado por todos, pero que en la práctica es difícil de lograrlo.

2.4.*2.6* *Cloud Networking*

Es un término emergente que surgió debido a que los modelos de virtualización y ambientes *Cloud* requieren un alto nivel de disponibilidad en su red, las cuales son mucho más exigentes que las redes tradicionales. Siendo una infraestructura de red, requerida para solventar las necesidades de *Cloud Computing*, como la escalabilidad, fiabilidad y retardo.

La plataforma principal de *Cloud Networking* es desarrollada por una compañía como Arista que brinda soluciones eficientes de *Cloud Networking*, ya que puede proveer *Datacenters* con soporte de *Cloud Computing* de alta escalabilidad, robustez y costos efectivos para alcanzar los requerimientos de calidad (Revelo, 2013).

2.4.3 Características del Cloud Computing

El servicio de *Cloud Computing*, cuenta con varios importantes factores que resaltan el tipo de servicio que ofrecen y que hacen atractiva la labor del usuario (Rosales, 2010) (fig. 37):

- ***On-Demand Capabilities***

Utiliza el concepto de *pay-as-you-grow* ya que tiene la capacidad de ofrecer servicios por adelantado o nuevos recursos informáticos de forma dinámica, según sea la necesidad del usuario, como un modelo para pequeñas empresas, que no obliga a comprometer recursos financieros mientras el negocio está creciendo.

- ***Elasticity***

Tiene la característica de elasticidad, que en si da la facultad para entregar un servicio escalable y bajo demanda, por ejemplo, facultar a que un usuario final utilice un servicio solo cuando lo requiera o por una temporada, en vez de contratar todo el servicio y no incurrir en gastos innecesarios, con poca o ninguna interacción por parte del cliente. En algunos modelos de entrega de *Cloud Computing*, la elasticidad a menudo se facilita a través de la virtualización, aunque no requiere virtualización.

- ***Wide Network Access***

Otra característica importante de la computación en nube es que los usuarios pueden acceder a los archivos y documentos importantes desde su estación de trabajo de oficina, computadoras portátiles o mientras está en movimiento a través de los teléfonos inteligentes y tabletas (Cloudoye, 2015). La información se puede leer o recuperar desde cualquier lugar con un simple acceso en línea, a través de una Interfaz de Programa de Aplicación, conocida como API, sobre el cual los clientes acceden a los recursos o correo electrónico a través de más de un sistema operativo.

- ***Multi-Tenancy***

Esta característica fue sugerida por la *Cloud Security Alliance* y hace referencia a la exigencia de políticas sobre la aplicación, la gobernanza de los recursos, la segmentación, los acuerdos de nivel de servicio (SLA) y los modelos de facturación/devolución de cargo para diferentes empresas. Algunos proveedores públicos de servicios en la nube, pueden ofrecer acuerdos de

nivel de servicio y administrar responsablemente los recursos de IT de la empresa, siendo un gran ahorro al dejar de contratar un especialista de tiempo completo, entregando confiabilidad a la red. La idea de que muchos usuarios compartan la infraestructura y los recursos permite aprovechar una infraestructura en la nube de alta seguridad y de manera escalable con otras unidades de negocio.

- ***Resource Pooling***

Un modelo de servicio en la nube de tipo pool, junta recursos informáticos del proveedor de servicios para servir a varios usuarios utilizando el modelo *multi-tenacity*, con diferentes recursos virtuales y físicos asignados y reasignados dinámicamente en base a la demanda del cliente. Según el Cuadrante mágico de *Gartner*, la puesta de recursos en comunidad (*pooling*) lleva a cualquier empresa a tener una economía de escala (Cloudoye, 2015). Empleando la centralización de aplicaciones, servidores, datos y recursos de almacenamiento, con auto aprovisionamiento y administración en la nube.

- Maneja la Virtualización de componentes, lo que ha permitido la densidad en la utilización de hardware, donde se podría incluir: servidores, aplicaciones, conmutadores, *routers*, firewalls, etc.

- Manejo organizado y automatización de la mayor cantidad de tareas, tanto como sean posibles, como por ejemplo, el manejo de la configuración, regular el aprovisionamiento y deaprovisionamiento de recursos y manejar tareas de *troubleshooting*, así como la automatización de procesos etc.

- Tiene la característica de evitar altos costos de licencias de software y esto es uno de los factores que permiten a las empresas proporcionar servicios en la nube asequibles. Adicionalmente promueve la competencia basada en líneas de negocio, en lugar de una competencia basada en fortalezas tecnológicas. Utiliza el desarrollo de estándares que permiten la federación de infraestructuras *Cloud.*

- El cambio más significativo con el *cloud computing* es el de la abstracción. Donde la mayoría de los proveedores de servicios *Cloud* ofrecen una o más capas de servicios (*Services Layers*) a sus consumidores. El aspecto operativo de las capas que están soportando la carga del servicio, está aislado del cliente. Por lo tanto, cuando hablamos de *Software as a Services* (SaaS), el cliente interactuará con la aplicación en sí, pero no con el sistema operativo o el hardware de la respectiva nube. Esta diferencia clave permite a las organizaciones que no cuentan con las habilidades necesarias de administración del sistema o instalaciones de cómputo para aprovechar las aplicaciones empresariales alojadas por otros (Halpert, 2011).

Figura 37. Principales características de Cloud Computing (Fuente: Cloudoye.com)

2.4.4 Ventajas, Desventajas y Riesgos de Cloud Computing

A continuación se resumen algunas de las ventajas (Prabhakar, 2008) (Hamilton, 2008):

- Permite la disminución de los presupuestos y esfuerzos dirigidos al área de TI, permitiendo la redistribución de los mismos en áreas de mayor interés para el negocio,

logrando una gran ahorro en costos de arquitectura de red y compatibilidad de hardware, así como el empleo eficiente del software libre (*open source*), siendo estos los mayores facilitadores para un ambiente *Cloud Computing.*

- El empleo de Software de código abierto (*open source*) y el *hardware* de los productos básicos son los principales facilitadores de un servicio *Cloud Computing*. El sistema operativo Linux, en particular se ha convertido en una pieza fundamental en el corazón de los mayores entornos en la nube. Del mismo modo, el software de virtualización *Xen* es utilizado por Amazon para albergar el mayor conjunto de máquinas virtuales en el mundo (aproximadamente medio millón) (Amazon, 2014).

- También está *Hadoop* que distribuye una plataforma informática que ayuda a miles de empresas a ejecutar masivos cálculos computacionales paralelos dentro de la nube (a través del servicio de *Amazon Elastic MapReduce*). La capacidad de evitar altos costos de licencias de software es uno de los factores que permiten a las empresas proporcionar servicios accesibles de *Cloud*.

- Disminución de los riesgos asociados a proyectos TI, beneficiando el retorno de la inversión (ROI) y disminuyendo los tiempos necesarios para el aprovisionamiento de infraestructura propia.

- Tiene una gran ventaja al momento de entregar accesibilidad de la información, ya que sistema puede estar ubicado en infraestructuras computacionales de alta disponibilidad, generalmente administradas por terceros. Esto incluye las facilidades para compartir y manipular información en un esquema multiusuario.

- Dado el crecimiento de la tecnología a nivel mundial, aporta de manera rentable y ecológica con la disminución del consumo de electricidad (*Green Computing*), ya que es muy rentable con el uso del espacio físico y mantenimiento de ambientes de temperatura constante y controlada, debido a su tipo de infraestructura computacional sobredimensionada.

Dentro de las principales desventajas, están:

- Una de las principales de un servicio de *Cloud* es la seguridad, debido a la transacción de información por infraestructura de terceros, ya que la información podría ser interceptada y manipulada.

- Dado el nivel de servicio, podrían presentarse riesgos con la privacidad de la información del usuario, con propósitos de mercadeo.

- Al ser un nuevo concepto de servicios, la falta de competitividad del mercado produce una gran dependencia con el proveedor de servicios *cloud* y las leyes que legislen el país por donde pase su infraestructura.

- Con este nivel de servicio, la competitividad entre empresas tiene una brecha muy pequeña como diferenciador en la entrega de la calidad de servicio.

2.4.5 Situación Actual de la Tecnología y Mercados

El enorme crecimiento del comercio electrónico (*e-commerce*), la expansión de las redes sociales y diversos servicios de la Web 2.0, han permitido que se incremente rápidamente la demanda de recursos computacionales. Grandes empresas como Google, Amazon y Microsoft se dieron cuenta rápidamente de que para sus necesidades, financieramente es más factible construir grandes centros de datos, que varios de tamaño pequeño y capacidad, ya que representa mucho más rentable al comprar recursos como la electricidad, el ancho de banda y almacenamiento en grandes volúmenes. En los centros de datos más grandes, se hace más factible maximizar la cantidad de trabajo por cada dólar invertido: de esta manera el usuario puede compartir componentes de una manera más eficiente, mejorar la densidad física y virtual de los servidores, reducir los tiempos de inactividad de los servidores y la relación del margen entre administrador y servidor (Tabla 6).

El sistema operativo y en particular Linux, se ha convertido en el engranaje clave de los mayores entornos de *Cloud Computing*. Al igual que, la popularidad de varios softwares de virtualización como *VMware* o *Xen,* que en especial este último es utilizado por Amazon cuyo conglomerado forma el mayor conjunto de máquinas virtuales en el mercado.

Tabla 6.

Relación de inversión económica entre Datacenter de mediana y gran escala (Fuente: Gorelik E.)

	Medium-sized DC	Very Large DC	Ratio (Large-to-Small DC)
Network	$95 per Mbit/sec/month	$13 per Mbit/sec/month	7.1
Storage	$2.20 per GByte/month	$0.40 per GByte/month	5.7
Administration	~ 140 Servers/ Admin	>1000 Servers/ Admin	7.1

Algunas compañías se dieron cuenta sobre los márgenes de inversión expuestos y desarrollaron una gran experiencia en el ámbito de la implementación de grandes *datacenters*, ya que se requiere de mucha inversión y amplios conocimientos técnicos para construir un *datacenter* con integración de servicios de *Cloud Computing* (Cloudoye, 2015). Una vez que estas empresas tuvieron una alta experiencia en la construcción y desempeño de toda la infraestructura para uso en su nube interna, aprovecharon todo la experiencia adquirida y el manejo óptimo de la tecnología existente, para construir *datacenters* con servicios hacia nubes públicas y ofrecer tecnología de *Cloud Computing* a otras empresas. Como resultado, las empresas como Amazon y Google se convirtieron en los principales proveedores de servicio con estructura de nube pública.

El 84% de los Directores de Informática (CIOs) tienen que recortar los gastos de aplicación al pasar de la infraestructura interna hacia la aplicación en la nube. Se tiene previsto que los ingresos por servicios en la nube a través de todos los segmentos de computación, crezca a una tasa compuesta de crecimiento anual del 33,2% entre 2012 y 2017.

Los servicios de *Cloud Computing* han fomentado dentro de una empresa políticas de economía de escala, ya que con el flujo continuo del *Big Data*, las cargas de trabajo están creciendo sustancialmente a un ritmo increíblemente rápido. Por ejemplo, al utilizan el modelo de *Cloud Infrastructure as a Service* (IaaS) les ha permitido a las empresas a superar este problema al momento de cambiar la propiedad y la gestión de su servidor físico a los servicios gestionados (Cloudoye, 2015). Esto permite a las empresas recortar sus costos de implementación del sistema y actualización, reduciendo la intervención humana, lo que ahorra más tiempo y dinero al tiempo que mejora la eficiencia del negocio.

Muchas de las tecnologías que ayudan a impulsar los servicios de *Cloud*, ya han estado presentes durante muchos años fortaleciendo su desarrollo. La Virtualización y la respuesta autonómica son áreas de la computación que han sido entrenadas por décadas, al igual que el desarrollo de la Internet y es por esto que los proveedores de *Cloud Computing* fueron capaces de montar esta tecnología a través de la experiencia adquirida (fig. 38).

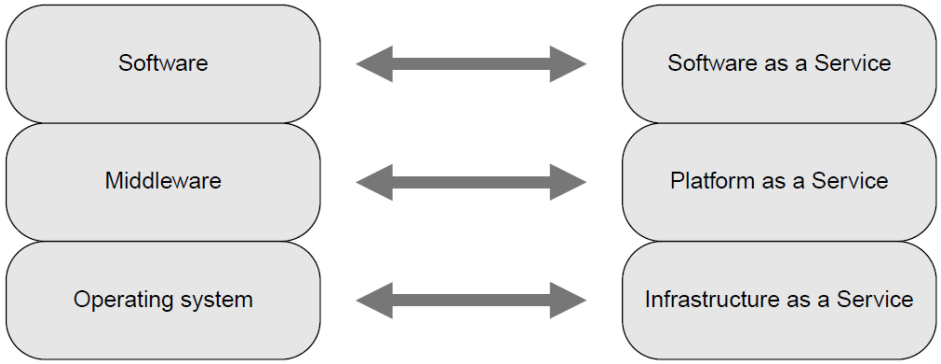


Figura 38. Modelo tradicional vs. Modelo de Cloud Computing. (Fuente: Gorelik E.)

Los acuerdos de servicios en la nube son más simples en comparación con las complicadas y tradicionales infraestructuras *in-house* (Cloudoye, 2015), que requieran altos gastos en la instalación del sistema, negociación de precios, aprobaciones de los gerentes de área, y un recortado plazo de entrega. El servicio *Cloud Hosting* sólo requiere una solicitud de servicio al proveedor para instalar la plataforma (fig. 39).

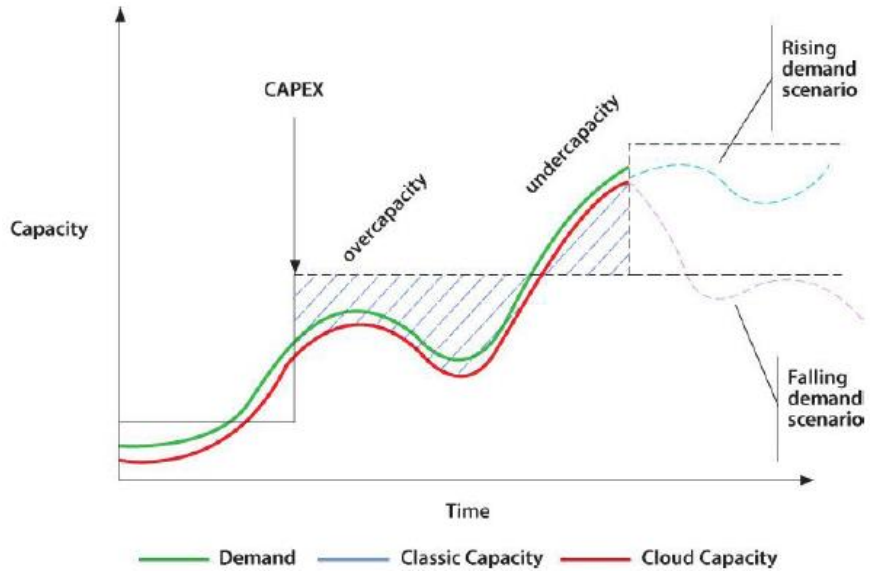


Figura 39. Modelo de crecimiento en Cloud Computing (Fuente: Cloudoye.com)

Desde que inició la adopción masiva de los servicios de *Cloud Computing* en 2005 y 2006, varios proveedores *Cloud* lograron una ventaja competitiva temprana del mercado. A continuación de esto, el mercado comenzó a florecer en torno a los servicios en la nube y muchas compañías vieron un gran potencial de negocio entrando en el mercado. Actualmente, a pesar de varios representantes importantes lideran el mercado, ninguno de ellos mantiene un bloqueo del mercado en términos de estándares de tecnología y características. Según *James M. Utterback* (1994) en su publicación *Mastering the Dynamics of Innovation*, dice que este efecto es una respuesta típica del mercado frente a la introducción de un nuevo e innovador producto. Razón por la cual se evidencia productividad en el próximo crecimiento del mercado del *Cloud Compunting*, donde el pico máximo en el número total de participantes del mercado en competencia alcanzarán un estado del producto de "diseño dominante" y se verá un número significativo de salidas de otros competidores del mercado, así como consolidaciones entre empresas.

2.5 INTRODUCCIÓN A LA ARQUITECTURA DE CLOUD COMPUTING

La revolución de los servicios de *Cloud Computing* comenzó a principios de la década de 2000 con la introducción de los servicios web, que utilizaban especificaciones abiertas como el lenguaje de programación, la plataforma y sin dependencia del proveedor.

Para lograr una adecuada implementación de la arquitectura, es necesario comprender que uno de los factores más importantes en la arquitectura de cualquier infraestructura es la capacidad de ser escalable. En la infraestructura tradicional *non-cloud*, los sistemas suelen estar diseñados para mantener el potencial de crecimiento futuro y de demanda de recursos. Cientos de organizaciones deben invertir considerables recursos financieros por adelantado al crecimiento futuro. Debido a que las infraestructuras *non-cloud* no proporcionan elasticidad, los recursos del sistema no pueden de forma inmediata expandirse y reducirse; esto conduce a exceso de aprovisionamiento constante de recursos y por lo tanto los sistemas están siendo ineficientemente utilizados la mayor parte del tiempo.

Por el contrario, la infraestructura *Cloud* es *multi-tenant*, para lo cual los recursos de computación se comparten a través de múltiples aplicaciones y se basa en la suposición de que todas las aplicaciones alojadas normalmente no pueden ser ocupadas al mismo tiempo o cuando una aplicación está inactiva. De este modo, los proveedores de servicios *Cloud* puede asignar recursos bajo demanda y mejorar la eficiencia en la utilización de los recursos computacionales. La infraestructura *Cloud* se compone de un conjunto de recursos compartidos de *hardware* (que incluyen servidores, almacenamiento, redes y un software de gestión *Cloud* que supervisa automáticamente la utilización de recursos, asignando dichos recursos según sea necesario (Gorelik, 2013). El proveedor de servicios *Cloud* debe asegurar la disponibilidad de los recursos informáticos para servir a todos los clientes, incluso en los momentos de alto tráfico, donde se genera gran cantidad de información y es parte de la razón de los beneficios económicos del *Cloud Computing*.

Antes de esta revolución, el sistema operativo se instalaba directamente en el hardware y se alojaban en el la múltiples aplicaciones, sin proporcionar aislamiento físico o virtual (fig. 40). Debido a que era difícil mover recursos rápidamente o re-balancear

aplicaciones a través de servidores, todo ese hardware y recursos costosos del servidor no se utilizaban eficientemente.

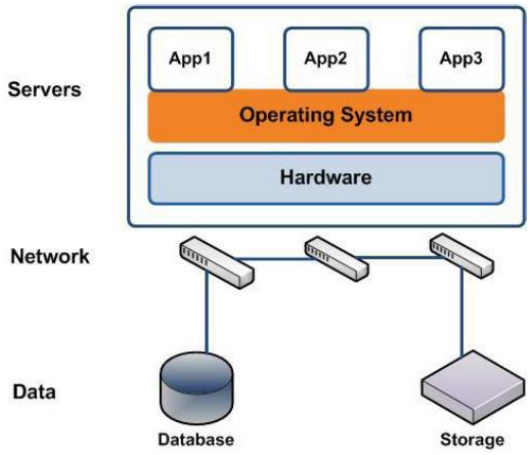


Figura 40. Servidores sin virtualización (Fuente: Gorelik E.)

2.5.1 Virtualización

Conceptualmente la virtualización es una tecnología de apoyo utilizada en todos los modelos de servicio en *Cloud Computing* para asignar recursos y proveer servicios en forma eficiente, dinámica y elástica (Rosales, 2010). La virtualización de sistemas se define como una herramienta que tiene como objetivo integrar varios elementos de software, ajenos e independientes unos de otros, dentro de un solo sistema o sobre una arquitectura de hardware compartida, el cual validará compatibilidad al momento de la instalación. Estas ejecuciones se llevan a cabo mediante instancias de archivos, formalmente denominados máquinas virtuales. Una máquina virtual es el encapsulamiento de un sistema operativo y un conjunto de aplicaciones en un archivo que es ejecutado por un *Hypervisor* (VMware, 2015).

Esta tecnología lleva ya varios años aplicándose en computadoras centrales o mainframes, desde que lo aplicó IBM en la década de los 60’s. Tiene la capacidad de separar el software del hardware en el que están instalados, siendo prácticamente la emulación de

una o más estaciones de trabajo o servidores, en una sola computadora física "emulación de hardware mediante software" (Franco, 2009). Este tipo de virtualización algunas veces se la conoce como "virtualización nativa" y permite que una sola máquina física aloje y comparta los recursos físicos entre varias máquinas virtuales.

Esta característica aplicada al *Cloud Computing* se materializa en que el usuario no tiene que preocuparse por la implementación concreta de los servicios de *Cloud*, ni tener en cuenta el hardware asociado a ellos. La virtualización es un paso importante hacia la infraestructura *Cloud*; sin embargo, el componente de servicio todavía estaba faltando en la cadena de procesos. La virtualización ha sido muy importante para el *Cloud Computing*, ya que ha posibilitado una optimización respecto al aprovechamiento de los recursos comunes, porque ha permitido que las aplicaciones sean independientes del hardware en el que se ejecutan y así proporciona un mecanismo para reasignar rápidamente aplicaciones en otros servidores (ORSI, 2010), dependiendo de las demandas computacionales que requiera la aplicación (fig. 41).

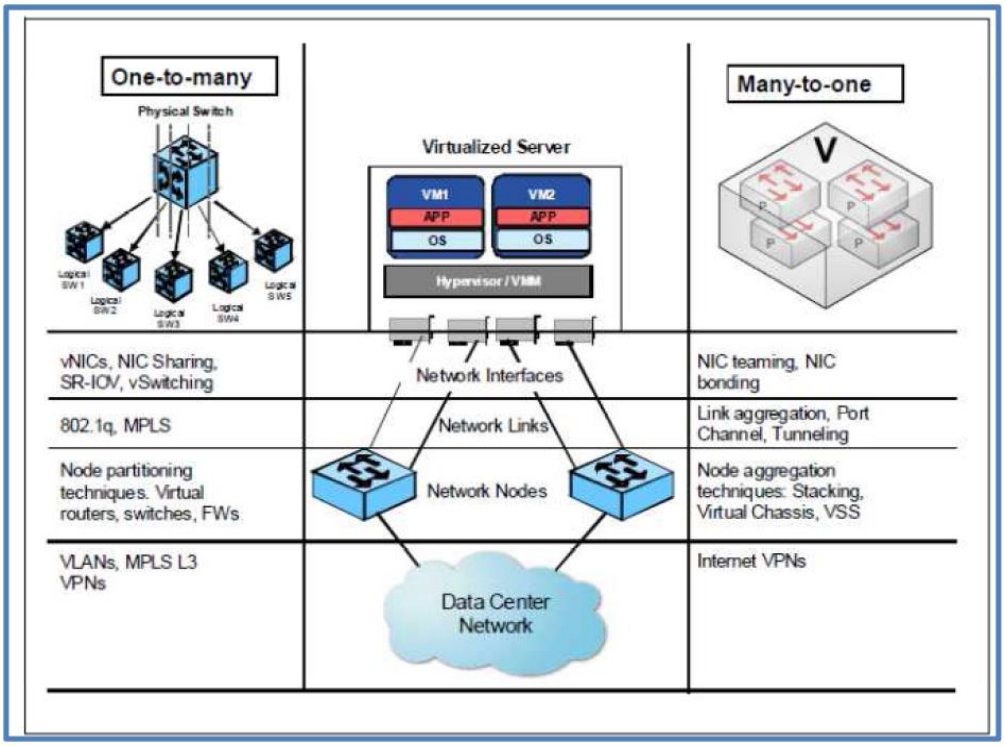


Figura 41. Redes Virtualizadas. (Fuente: Figueroa C.)

Mediante la virtualización de servidores, se puede lograr tener varios servidores dentro de un único servidor, donde cada servidor está formado con su propio conjunto de hardware virtualizado y sobre el cual corre un sistema operativo independiente, con diferentes aplicaciones (fig. 42).

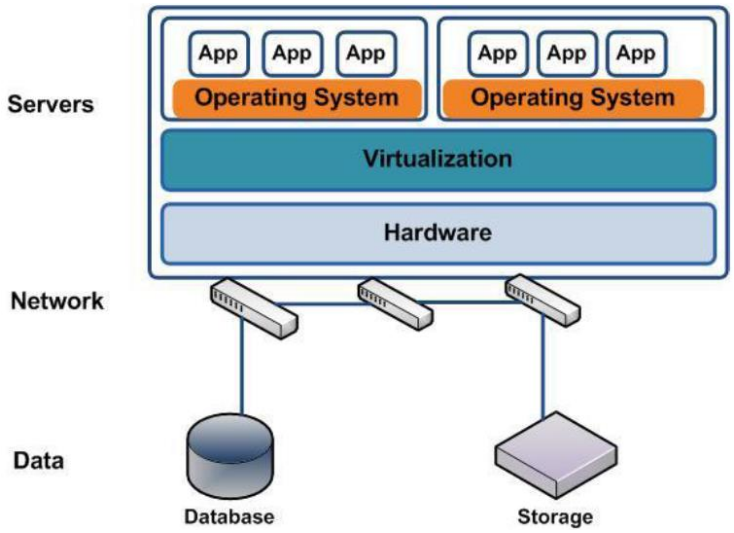


Figura 42. Servidores virtualizados. (Fuente: Gorelik E.)

Los entornos virtualizados gestionables por administradores de sistemas internos y plataformas de virtualización por defecto, no proporcionaban la capa de abstracción que hacia posible los servicios de *Cloud*. Para lograr lo necesario dentro de un entorno *Cloud*, una capa de abstracción y aprovisionamiento bajo demanda debe ser provista desde las capas superiores (fig. 43). Esta capa de servicio es un atributo importante de cualquier entorno *Cloud*, ya que oculta la complejidad de la infraestructura y proporciona una interfaz de gestión en la nube para los usuarios (Gorelik, 2013). Desde la introducción de la virtualización, se empezó a remediar el problema de deficiencia de rendimiento y las cosas han cambiado, ya que se ha mejorado la utilización de los recursos y la eficiencia energética, ayudando a reducir sustancialmente los gastos generales en el mantenimiento del hardware y proporcionando un alto desempeño para la recuperación de desastres, logrando así un servicio de alta disponibilidad.

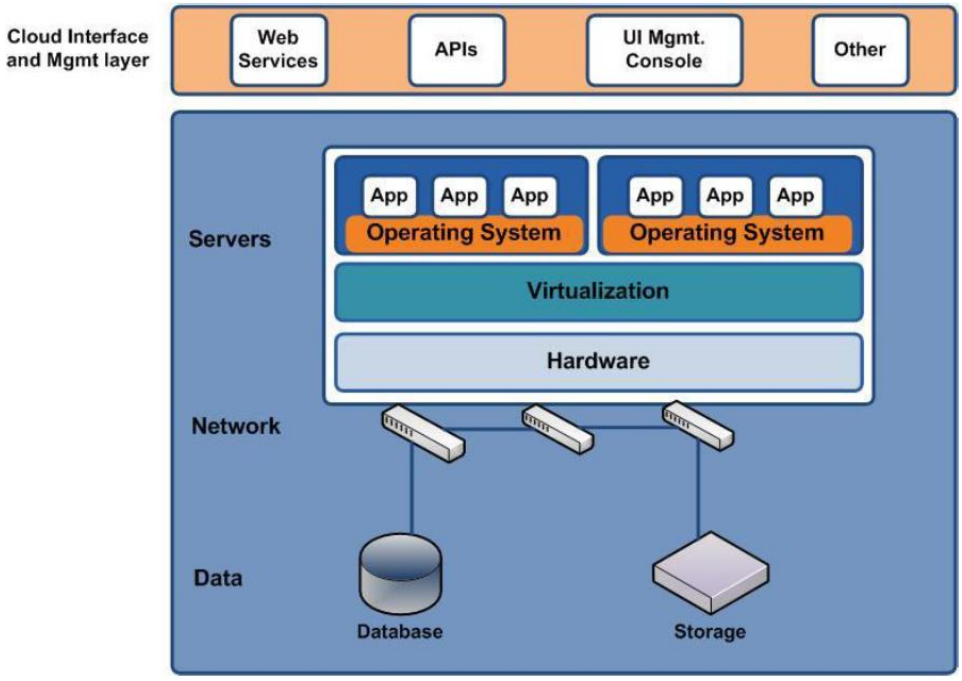


Figura 43. Infraestructura simplificada del Cloud. (Fuente: Gorelik E.)

2.5.2 Tipos de Virtualización

- **Emulación de Hardware**

Mediante esta técnica el equipo físico se abstrae ofreciendo varios servidores virtuales o múltiples sistemas operativos, es decir que en el sistema anfitrión se utiliza una máquina virtual que emula el hardware. Esta es el tipo de emulación más compleja, como se muestra en la figura 44 (Campos y Zamorategui, 2012).

La principal ventaja es la de ejecutar un sistema operativo sin modificar, esto permite que los múltiples sistemas operativos puedan coexistir sobre uno o más procesadores, incluso se pueden ejecutar varias máquinas virtuales simulando procesadores diferentes, recursos de memoria y recursos de almacenamiento. Mientras que el principal problema con la emulación de hardware es que puede resultar lenta ya que cada instrucción debe ser simulada por el hardware base (Figueroa, 2013).

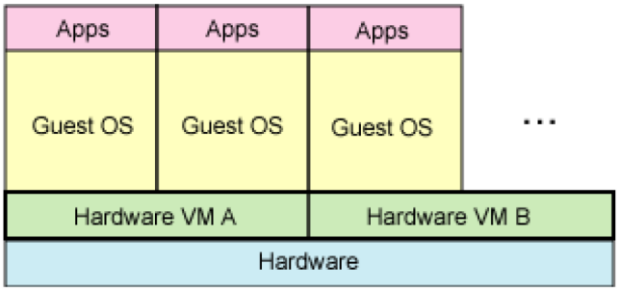


Figura 44. Emulación por hardware. (Fuente: Campos D. y Zamorategui E.)

- **Virtualización Completa.**

Es conocida también como Virtualización nativa, esta utiliza un gestor de máquina virtual que media entre el sistema operativo invitado y la capa de hardware subyacente (nativa). Algunas instrucciones protegidas deben capturarse y manejarse dentro del hypervisor. El hardware no es propiedad del sistema operativo invitado sino que es compartido a través del hypervisor (Hamilton, 2008).

La virtualización completa tiene la ventaja de ser más rápida que la emulación de hardware, debido a la mediación del *hypervisor*, aun así el hardware y sistema operativo nativos por ende tienen un mayor rendimiento (fig. 45).

A su vez permite que mientras un sistema operativo este corriendo no sufra modificaciones, pero el sistema operativo invitado debe soportar el hardware anfitrión.

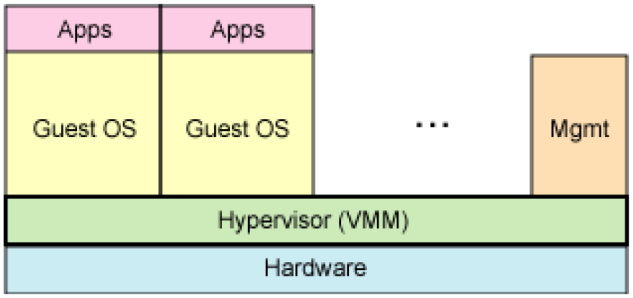


Figura 45. El *hypervisor* como mediador entre el hardware y la máquina virtual. (Fuente: Campos D. y Zamorategui E.)

2.5.3 Paravirtualización

Este método opera de manera similar a la virtualización completa, pero con la diferencia de no hacer una simulación de los recursos de hardware (Campos y Zamorategui, 2012), ya que utiliza un *hypervisor* para compartir el acceso al hardware anfitrión, pero a su vez integra código para las máquinas virtuales invitadas, mediante el uso de *Application Programming Interface* (API) que está al tanto de la virtualización en el propio sistema operativo (Figueroa, 2013).

La paravirtualización ofrece un rendimiento próximo al de un sistema no virtualizado, es posible soportar varios sistemas operativos diferentes de manera concurrente, lo que significa un mejor rendimiento ante otras soluciones de virtualización (fig. 46), ya que evita la necesidad de recompilar y capturar dado que los sistemas operativos virtualizados cooperan en el proceso de virtualización (Fernandez, 2009).

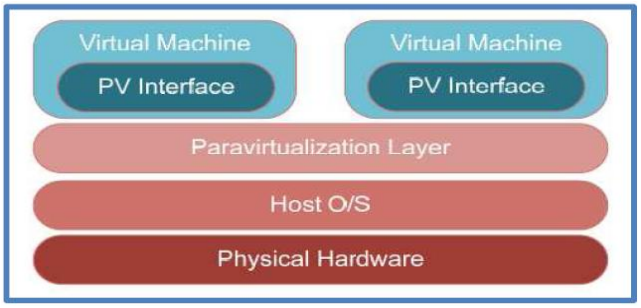


Figura 46. La Paravirtualización comparte el proceso con el SO invitado (Fuente: Figueroa C.)

2.5.4 Virtualización a Nivel del Sistema Operativo

Esta técnica virtualiza los servidores encima del propio sistema operativo. Sólo soporta un solo sistema operativo y simplemente aísla a los servidores de manera independiente

(Hamilton, 2008), por lo que el *kernel* se ejecutará en el sistema operativo anfitrión, pero su funcionalidad estará en cada una de las particiones virtualizadas (fig. 47). Siendo este un método muy diferente a anteriores, donde el servidor físico y una única instancia del sistema operativo anfitrión son virtualizadas en múltiples particiones aisladas conocidas como servidores privados, haciendo que cada partición duplique un servidor real (Figueroa, 2013).

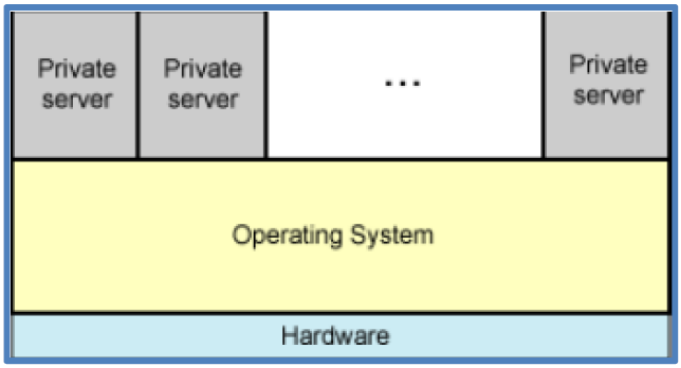


Figura 47. Virtualización de Sistema operativo (Fuente: Campos y Zamorategui E.)

2.5.5 Hypervisores

Un *hypervisor* es el software encargado de mediar el hardware físico con el hardware de las máquinas virtuales, También es llamado administrador de máquinas virtuales, ya que es un programa que permite que múltiples sistemas operativos compartan el hardware de un equipo físico. Existen hypervisores de 2 tipos, tipo 1 o *bare-metal* y tipo 2 o *hosted*.

2.5.5.1 Hypervisores Bare-metal

El *hypervisor* de tipo *bare-metal* no opera bajo un sistema operativo instalado, en vez de esto cuenta con un acceso directo sobre los recursos disponibles del hardware, ya que para este

tipo de tecnología de virtualización, el hardware que soporta es limitado debido a que normalmente está constituido por un conjunto limitado de *drivers* (Nerion, 2012). Estos *hypervisors* abstraen la capa de hardware facilitando la asignación de cuotas de recursos computacionales para la ejecución de múltiples máquinas virtuales.

Este tipo de *hypervisor* se adapta de mejor manera a un *Datacenter* de tipo profesional y de empresa, debido a que dispone de potentes características avanzadas, para entregar una alta disponibilidad y gran seguridad de servicios, con un buen rendimiento mediante una administración centralizada de la infraestructura de virtualización y recursos, ya que tiene facilidades operativas y administrativas sobre las máquinas virtuales bajo su cargo, incluyendo la posibilidad de clonación de máquinas virtuales y el cambio de configuraciones de hardware (procesador, memoria, almacenamiento y red) (fig. 48) en tiempo de ejecución (los cuales pueden o no reflejarse inmediatamente dependiendo del sistema operativo instalado en la máquina virtual) (Rosales, 2010), mientras que una desventaja es la necesidad de una infraestructura física robusta y dedicada a la provisión de máquinas virtuales.

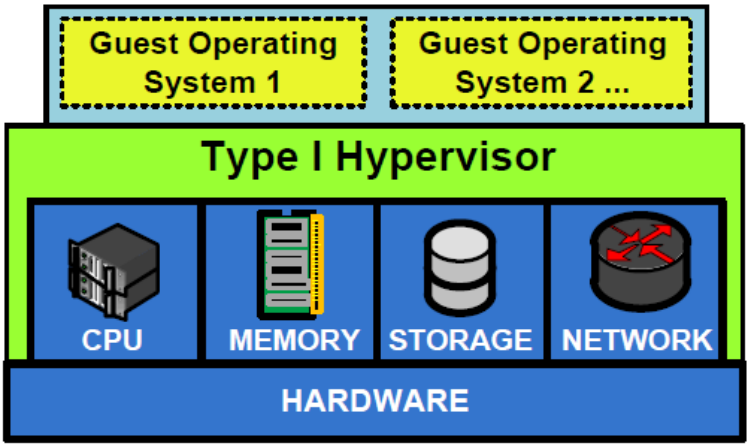


Figura 48. Diagrama de despliegue de un *hypervisor* Tipo 1. (Fuente: Rosales E.)

En el mercado existen los siguientes hypervisores:

- ***VMware ESX y ESXi***

VMware a través de estos hipervisores proporciona la base para la construcción de una infraestructura de TI fiable y dinámica al ser la plataforma líder y más madura de virtualización del mercado, a través ESX y ESXi soporta todos los niveles de la virtualización, desde el de uso de escritorio o una virtualización compleja de servidores con fiabilidad y rendimiento a las empresas dentro un *full-fledged Software-Defined Datacenter* (VMware, 2015). Ofrece recursos líderes, como características avanzadas de *hypervisors abstract processor*, memoria, almacenamiento, red y alta escalabilidad al manejar varias máquinas virtuales donde cada una puede ejecutar un sistema operativo y aplicaciones independientes sin modificar. Tiene los altos costes de licenciamiento (Nerion, 2012).

- ***Microsoft Hyper-V***

Es un producto independiente que cuanta con una solución de virtualización confiable y optimizada, además que permite a las organizaciones optimizar el uso de sus servidores y reducir costos de operación y administración, al incorporar herramientas y servicios que sus usuarios podrán utilizar para crear un entorno de servidores virtualizados (Microsoft. 2015); practicamente desde su aparición, ha llegado a ser un competidor directo de VMware y sus servicios. Cuenta con ciertas desventajas, ya que no dispone de ciertas características avanzadas disponibles en los tipos de productos de la misma gama VMware. También, dada su funcionalidad, se integra perfectamente con los productos Windows (Nerion, 2012). Es recomendado para ciertos usuarios que no necesitan funcionalidades avanzadas, ya que este producto puede ser perfecto para llevar a cabo un esquema de victualización.

- ***Citrix XenServer***

Citrix es una plataforma de gestión líder en virtualización de servidores e hipervisor, que permite reducir los costos de almacenamiento de una infraestructura virtualizada con la función de caché de lectura en memoria, donde mejora la experiencia del usuario final, además de ser gratuito (Citrix, 2016); en ciertas versiones no dispone de características

avanzadas, como con VMware y sus servicios. Mediante el licenciamiento de pago, se obtienen características lideres y avanzadas como vGPU – GPU, las cuales ofrecen gestión personalizada, automática y disponibilidad especializada (Nerion, 2012). XenServer cuenta con el mejor rendimiento en su clase para la aplicación y la virtualización de escritorio y el *hypervisor* tiene una plataforma basada en los proyectos *open source Xen*.

- ***Oracle VM***

Oracle VM fue diseñado para ofrecer eficiencia y hoy en día está optimizado para ofrecer rendimiento, los productos de virtualización de servidores que Oracle soporta, son las arquitecturas SPARC y x86, además de una variedad de cargas de trabajo, en Linux, Windows y Solaris (Oracle, 2016). Al igual que Citrix, también formó su *hypervisor* a partir del proyecto *Xen*. Ofrece un producto que se adapta perfectamente a la tecnología desarrollada por Oracle como tal, entregando soluciones que están basados en el hipervisor, la virtualización integrada en el hardware y sistemas operativos propios de Oracle, para entregar una solución completa y optimizada, pero no presenta funcionalidades avanzadas que se pueden encontrar en otros hypervisores *bare-metal* del mercado como VMware, Microsoft o Citrix (Nerion, 2012).

2.5.5.2 Hypervisores Hosted

Para que opere este *hypervisor* de tipo *hosted* es necesario que el software de virtualización sea instalado sobre un sistema operativo que haya sido previamente implementado, casi como si se instalara una aplicación cualquiera. En comparación con la tecnología de tipo *bare-metal*, esta presenta una mayor compatibilidad con el hardware, dado que es el propio sistema operativo el encargado de cargar y gestionar los *drivers* (Nerion, 2012), en razón a que opera sobre recursos computacionales no dedicados y de medianas prestaciones, ofreciendo las facilidades para ser instalados y ejecutados sobre un sistema operativo principal. Debido a que esta tecnología opera bajo un sistema operativo definido y no cuenta con un acceso

directo sobre el hardware, el consumo de recursos se incrementa, dado que tiene que ajustarse a la ejecución de múltiples máquinas virtuales, para evitar posibles problemas de desempeño, al degradar el rendimiento de la máquina virtual, por falta de recursos (Rosales, 2010).

Estos *hypervisors* utilizan los servicios del sistema operativo para acceder al hardware disponible y son capaces de ejecutar múltiples máquinas virtuales compartiendo los recursos físicos con el sistema operativo principal. Típicamente se la utiliza con propósitos de *testing* y desarrollo sobre estaciones de trabajo o para aquellos servicios que requieren ejecutar y funcionar con más de un sistema operativo (fig. 49).

Dentro del ámbito de los hypervisores de tipo *hosted* se encuentran varios: VMware y su diferentes modelos con *Workstation*, *Player*, *Fusion* y *Server*; Microsoft *Virtual PC*, Oracle VM *VirtualBox*, Red Hat Enterprise Virtualization (KVM) o Parallels Desktop.

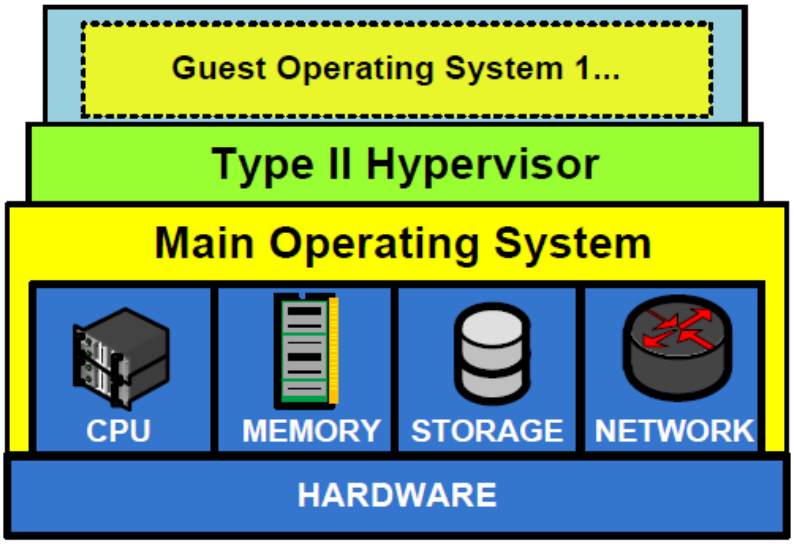


Figura 49. Diagrama de despliegue de un *hypervisor* Tipo 2. (Fuente: Rosales E.)

- ***VMware Player***

Este *hypervisor* es el más elemental de la gama ya que sólo puede ejecutar una única máquina virtual y no permite crear nuevas máquinas virtuales (Nerion, 2012).

- ***VMware Workstation Player***

Este *hipervisor* antes era conocido como *Pro Player* y es una aplicación de virtualización de escritorio optimizada y más robusta que ejecuta uno o más sistemas operativos en el mismo equipo, con una sencilla interfaz de usuario, soporte de sistema operativo sin igual y la portabilidad necesaria (VMware. 2015); en su funcionalidad entrega soporte para *snapshots* o guardado de estados. Suele ser empleado en entornos de *testing* o desarrollo (Nerion, 2012). Tiene un buen soporte de sistema operativo para los profesionales de TI para lograr poner sus usuarios en marcha y funcionando con un escritorio corporativo.

- ***VMware Fusion***

Es una versión del hipervisor Workstation pero para Mac (Nerion, 2012). Es muy recomendado para los usuarios nuevos y anteriores de Mac que quieren continuar ejecutando la aplicación de Windows imprescindibles de la manera más fácil, rápida y fiable de ejecutar.

- ***VMware Server***

Este *hypervisor* ofrece una aplicación gratuita y fácil para utilizar Windows y Linux, proporcionando una experiencia superior a la virtualización de cargas de trabajo tipo servidor para las empresas que requieren optimizar la utilización de sus activos tecnológicos y está diseñado para ser administrado a través de la red (VMware, 2015). Es una alternativa a la utilización de ESXi, ya que es similar a la versión Workstation, pero su plataforma de virtualización tiene ciertas limitaciones, debido a que sólo soporta un *snapshot* por máquina virtual (Nerion, 2012).

- ***Microsoft Virtual PC***

En el 2003 este *hipervisor* adquirió la tecnología *Virtual PC* de la empresa *Conectix* y es lo último en tecnología de virtualización de Microsoft. Esta tecnología se puede usar para

ejecutar más de un sistema operativo a la vez en un equipo, así como muchas aplicaciones de productividad en un entorno virtual de Windows (Microsoft, 2015). Todo esto permite que Windows *Virtual PC* sea gratuito y por ende corra bajo entornos de Windows 7 en adelante, con cierta disponibilidad para versiones antiguas (Nerion, 2012).

- ***Oracle VM VirtualBox***

VirtualBox es una solución profesional disponible como software de código abierto bajo los términos de la Licencia Pública General de GNU (GPL) versión 2, es un producto con una tecnología madura, características avanzadas de alto rendimiento y eficiente para virtualización de x86 y AMD64/Intel64 de uso empresarial o para usuarios individuales; debido a que comparte características con *vSphere* e *Hyper-V*, la hace una alternativa muy interesante a tener en cuenta frente a tecnologías que implican altos costos.

- ***Red Hat Enterprise Virtualization***

Red Hat Kernel-Based Virtual Machine (KVM) opera en base a que transforma el *kernel* de Linux en un *hypervisor*, con un rendimiento igual o superior a los equipos sin sistema operativo que ofrece características avanzadas de nivel empresarial y con un sistema de gestión sólido bajo entorno Windows para realizar la gestión de múltiples terminales, como con *Microsoft Exchange* y alta capacidad de E/S de cargas de trabajo de bases de datos Oracle (RedHat, 2015). Todas estas funcionalidades hacen que tenga características de las soluciones de tipo *hosted* y *bare-metal*, de tal manera que las máquinas virtuales logren acceso directo sobre el hardware físico, como lo hace *vSphere*. (Nerion, 2012)

- ***Parallels Desktop***

Al igual que *VMware Fusion* en su versión para Mac, este permite obtener todos los archivos, aplicaciones, navegador y mucho más desde una PC Windows a Mac. Sin embargo, tiene una edición más potente conocida como *Parallels* Server, que dispone de características

avanzadas como crear redes virtuales para escenarios de red complejas y de pruebas, incluyendo la simulación de diferentes entornos de inestabilidades de red, haciéndolo razonable en temas de rendimiento y precio (Nerion, 2012). Este producto también tiene una versión de escritorio para Windows o Linux e incluso permite actualizar su máquina virtual de Windows 7 o 8 para Windows 10.

2.6 TIPOS DE CONFIGURACIÓN DE CLOUD COMPUTING

El *Cloud Computing* puede ser utilizado en una gran variedad de formas, aquí se incluyen los siguientes tipos de configuración: *End User to Cloud, Enterprise to Cloud to End User, Enterprise to Cloud, Enterprise to Cloud to Enterprise, Private Cloud e Hybrid Cloud*. Esto hace posible que las empresas puedan variar en el empleo de varios fabricantes y comerciantes de soluciones *Cloud* o incluso utilizar el servicio de múltiples proveedores de servicios en la nube (*Cloud Providers* - CP)

2.6.1 End User to Cloud

En el caso de este tipo de configuración, la nube contiene los datos que el usuario final provee y aplicaciones que los usuarios finales acceden directamente. Un usuario puede acceder a este tipo de nube utilizando cualquier dispositivo con una conexión a Internet y la disponibilidad de un *open-client*, el cual es independiente de cualquier plataforma en particular (fig. 50). Comúnmente antes de lograr tener acceso a un aplicación que está corriendo en la nube, el usuario debe proveer al sistema un nombre de usurario y contraseña válidos. También se conoce que se puede tener acuerdos de calidad de servicio o SLAs que garantice el servicio provisto por el *cloud-vendor* a través de la nube. Ej. Dentro de este grupo

se puede nombre a los sitios web de redes sociales como: *Facebook, Twitter, Myspace*; así como sitio de alojamiento e-mail como: *Gmail, Hotmail o Yahoo*.

Figura 50. Configuración *End User to Cloud.* (Fuente: Skillsoft)

2.6.2 Enterprise to Cloud to End User

Este tipo de configuración de servicio *Cloud* se lo utiliza cuando una empresa utiliza la nube para proveer servicios y datos a usuarios finales. Donde cada usuario final tenga una identidad para la autenticación, con los derechos suficientes de permisos para que la empresa tenga también acceso al servicio en la nube, sin la necesidad de otra identificación. Se lo emplea cuando un usuario final contacta a una empresa solicitando datos específicos, en este caso, la empresa puede ingresar a la nube y obtener los datos, con la capacidad hasta de poder editarlos y finalmente poder entregarlos al usuario que los solicitó, pero sin la necesidad de conocer la localización física de los servidores de almacenamiento de los datos (fig. 51). Para asegurar la flexibilidad en el servicio, los usuarios finales se conectan a la nube a través de clientes abiertos y *API*s comunes para determinar la localización física del hardware responsable de entregar el servicio en la nube (Skillsoft, 2011). Para este tipo en particular se deben considerar ciertos acuerdos de servicio como el: ciclo de vida de la información, una seguridad apropiada, herramientas apropiadas de medición para control de costos y aprovisionamiento de recursos.

Figura 51. Configuración *Enterprise to Cloud to End User* (Fuente: Skillsoft)

2.6.3 Enterprise to Cloud

Para este tipo de escenario, una empresa utiliza y controla el servicio en la nube para procesamiento computacional interno, sin casi interacción directa del usuario final sobre el *Cloud*. Particularmente tiene los mismos requerimientos enunciados en el escenario de *Enterprise to cloud to end users*, pero también tiene otros como: *backups* y almacenamiento de información, funciones de administración y e-mail, *cloud databases* durante el procesamiento y el desarrollo de aplicaciones e imágenes de máquinas virtuales que podrían ser llevadas entre *cloud providers* (fig. 52).

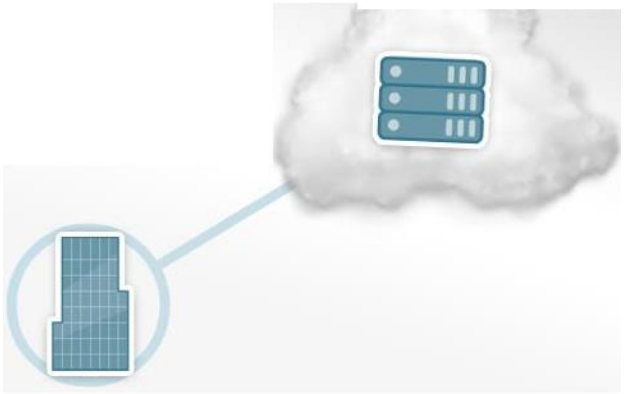


Figura 52. Configuración *Enterprise to Cloud.* (Fuente: Skillsoft)

2.6.4 Enterprise to Cloud to Enterprise

Este tipo de configuración involucra a dos empresas utilizando aplicaciones o datos en la misma nube, siendo en este caso el *Cloud* el medio que los une para trabajar juntos. Al igual que el resto de servicio de tipo corporativo, utiliza los mismos requerimientos de configuración y otros adicionales que incluyen la interoperabilidad y la administración de transacciones concurrentes, con el propósito principal de salvaguardar la integridad de la información o cambios realizados entre ambas empresas (fig. 53). Por ejemplo, el sistema funcionaria cuando un comerciante y su comprador desean automatizar el proceso de su relación comercial, haciendo que un sistema transaccional para sus órdenes y abastecimiento de productos este siempre actualizado y con información confiable en una base de información en la nube compartida (Skillsoft, 2011).

Figura 53. Configuración Enterprise to Cloud to Enterprise (Fuente: Skillsoft)

Un entorno de servicios en la nube puede proporcionar exactamente las mismas tecnologías que una infraestructura de IT tradicional, la principal diferencia, es que cada una de estas tecnologías se proporciona como un servicio. Este servicio puede ser accesible a través de una capa de interfaz de gestión de la nube, que proporciona acceso a través de REST / SOAP de servicios web, programación de API o un sitio web de la consola de gestión.

En base a la información analizada anteriormente sobre los tipos de *Cloud Compunting* y tomando como referencia los principales informes del NIST (*NIST Cloud Computing Standards Roadmap*) y Deloitte (*Cloud Computing: Forecasting change. Market Overview and Perspective*) se definen tres características fundamentales que marcan la clasificación de las soluciones C*loud*: Familias (*Cloud Computing Deployment Models*), Formas de Implementación (*Cloud Computing Services Layers*) y Agentes Intervinientes (Roles) (fig. 54).

Estas tres características, junto con sus diferentes tipos de soluciones asociadas, se pueden representar en un cubo de tres dimensiones, tal y como se muestra en la figura inferior:

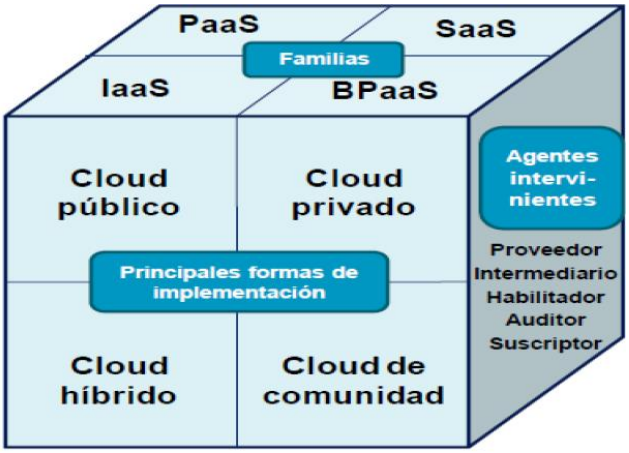


Figura 54. Cubo *Cloud Computing* (Fuente: Salazar R.)

Mediante la combinación de estas tres dimensiones se detalla los distintos tipos de *cloud computing* existentes en el mercado así como sus principales agentes:

2.7 CLOUD COMPUTING DEPLOYMENT MODELS

De acuerdo al Instituto Nacional de Estándares y Tecnología (NIST), los modelos de Implementación para *Cloud Computing*, tienen un método en particular que es el de entregar un servicio, con la características particulares que se adaptan dependiendo de la carga de trabajo que requieran. Los siguientes tipos principales de escenarios han permitido desarrollar un intercambio computacional de almacenamiento y de recursos entre desarrolladores y empresas que prestan servicios. Cabe destacar que estos escenarios son implementados en los *Datacenters*, los mismos que dependiendo de su disposición pueden tener un enfoque interno, externo o combinado:

2.7.1 *Public Cloud*

Las nubes públicas son nubes donde el proveedor entrega un servicio en la nube a cualquier cliente que desee acceder a ella, incluyendo usuarios y organizaciones académicas, científicas o comerciales; el acceso a este tipo de nubes es a través del uso de Internet o redes privadas virtuales (VPNs). Este escenario de tipo público se da cuando las empresas requieren mover datos o aplicaciones desde un entorno interior hacia el exterior. En muchos casos, los proveedores de la nube de hoy ofrecen poco en compensación por las pérdidas que se generen dentro de los acuerdos de nivel de servicio (SLA), muy a menudo, la única compensación ofrecida es el reembolso de los honorarios pagados por el consumidor (Halpert, 2011).

Para muchas organizaciones, el impacto en el negocio por el tiempo de inactividad de la aplicación puede tener repercusiones fueres que van por encima del costo del servicio en la nube. Considere el ejemplo extremo de una bolsa de valores, donde el tiempo de inactividad se mide en millones de dólares por hora frente a un servicio en la nube que se mide en centavos de dólar por hora. Esto ha provocado que muchas organizaciones eviten utilizar los servicios de nubes públicas para realizar tareas de misión crítica que representen

cargas de trabajo y mejor emplearlo para actividades no críticas. El escenario que se crea con la interconexión de varias nubes públicas es conocido como *External Cloud*.

Las nubes externas involucran recursos y servicios IT que son asociados fuera del establecimiento, a modelo de *hosting*. Estos recursos y servicios son vendidos con las cualidades de *Cloud Computing*, el auto-servicio, *pay-as-you-go*, administración de recursos de forma dinámica y escalabilidad infinita. Cabe destacar que estos recursos no son totalmente controlados por el cliente (CISCO, 2014). Las nubes públicas más populares incluyen *Amazon Web Services, Google App Engine y Microsoft Azure* (fig. 55).

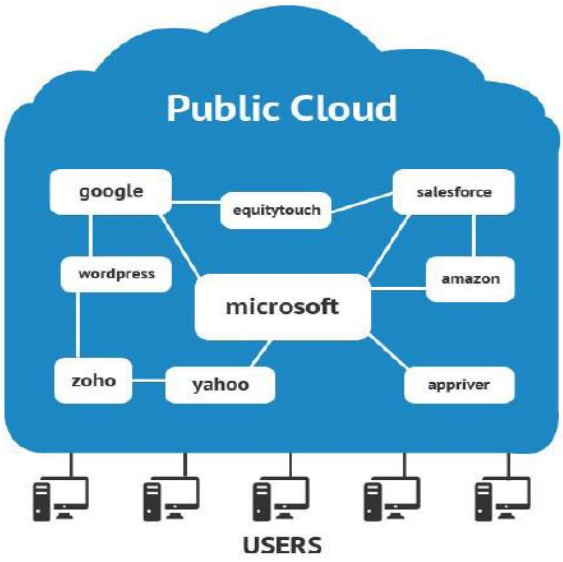


Figura 55. Diseño de una *Public cloud* (Fuente: Cloudoye.com)

Otras preocupaciones se dan cuando se tiene relación con temas de seguridad, tales como los requisitos de alojamiento de la información, han impedido la adopción de la computación en nube pública a través de algunos países, ya que se debe tener en cuenta, que cuando se aprovecha el uso de las nubes públicas, los consumidores deben garantizar que su uso no violente los requisitos de regulatorios, industriales o legislativos de un área (fig. 56). La *cloud* pública proporciona varias ventajas como elasticidad, a través de la cual los proveedores de servicios pueden equilibrar las demandas de consumo de los clientes

mediante técnicas de virtualización ocupadas para obtener un mejor aprovechamiento de los recursos computacionales, lo cual a su vez se traduce en tarifas más baratas en el uso de recursos de cómputo para el usuario, pues solo pagará lo que consume (Melendez, 2012).

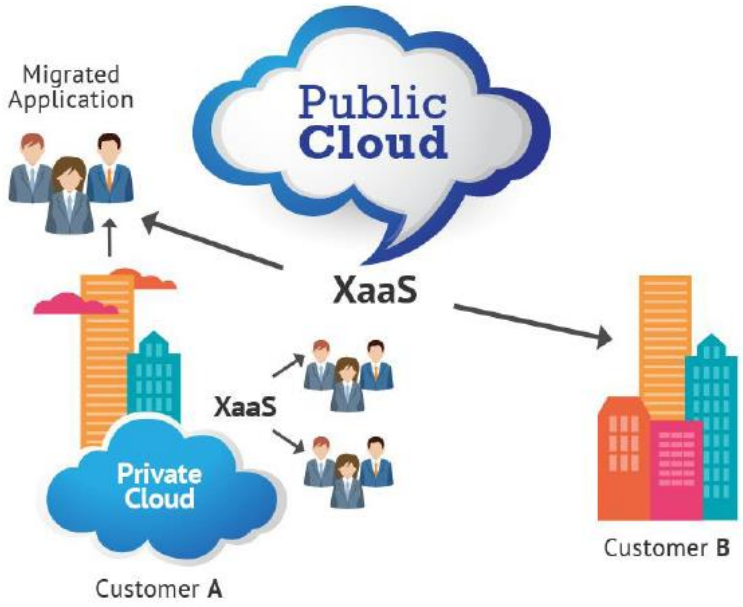


Figura 56. Como funciona una Public cloud (Fuente: Cloudoye.com)

2.7.2 *Private Cloud*

En una nube privada o nube interna, los servicios de *cloud computing* son proporcionados por una organización interna para que sea utilizado por otras organizaciones internas. El proveedor, en este caso es más a menudo el departamento interno de Tecnología de la Información (TI) o el departamento de Sistemas de la Información (SI). Sin embargo, esta misma puede ser administrada por la misma organización o por un tercero especializado (fig. 57). Los consumidores varían, pero por lo general estos consumidores son consumidores de otros servicios de TI. Dado que tanto el consumidor y el proveedor son organizaciones internas, las nubes privadas permiten a los consumidores un mayor control sobre la calidad del servicio prestado por la nube (Halpert, 2011).

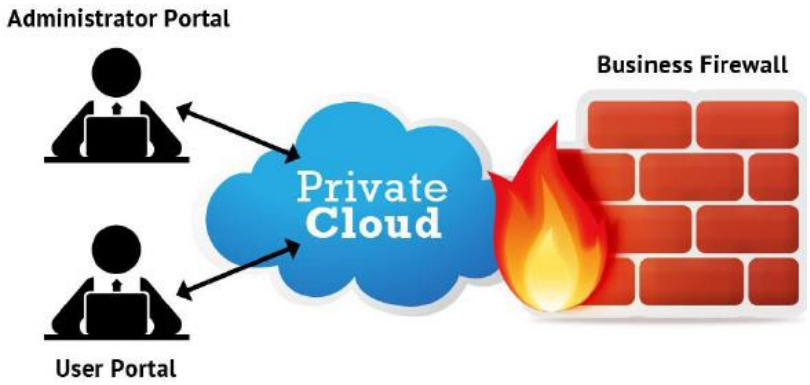


Figura 57. Diseño de una Private cloud (Fuente: Cloudoye.com)

Estas nubes privadas se ejecutarán en *Datacenters* seguros fuera de la empresa, vinculando a todas las oficinas de la compañía a través de túneles cifrados por la Internet pública. Una vez comunicada de forma segura la nube privada con todas las sucursales de la empresa, esta se dispondrá a colocar sus datos y aplicaciones en la nube, siempre de manera segura y privada (Lasica, 2009). Por ejemplo, un cliente interno puede más fácilmente tener la prioridad relativa de una carga de trabajo en particular. En otro caso, el consumidor interno podría requerir características específicas respecto a lo importante que es la carga de trabajo, y por lo tanto, los requisitos de disponibilidad de la nube.

Las nubes privadas generalmente proporcionan el mayor control, privacidad, gran flexibilidad, seguridad, menor latencia y habilidad para estandarizar los procesos tecnológicos, ya que tanto proveedor y el consumidor son parte de la misma organización. Este control tiene un precio, como la organización en última instancia lleva el costo total de la infraestructura *cloud*. De acuerdo con un reporte generado por *Sun Microsystem* en el 2009 titulado "*take your business to a higher level*", menciona que la *cloud* privada es una buena opción para aquellas empresas cuya preocupación sea la protección de sus datos y niveles de servicio (Melendez, 2012).

Una alternativa recientemente introducida por los proveedores de este servicio es crear una nube privada virtual (fig. 58). Una nube privada virtual está constituida por una nube privada dentro de la infraestructura física de una nube pública. Debido a la asignación de recursos específicos dentro de la nube, el cliente asegura todos sus datos almacenados en

la nube privada y solo realiza las tareas en los servidores dedicados o nube virtual (es decir estos servidores no se comparten con ningún otro cliente del proveedor de la nube) (Ullauri, 2013). Las organizaciones utilizan software que permite la funcionalidad de *Cloud*, como *VMWare, vCloud Director, vs OpenStack.*

Figura 58. Beneficios claves del Private cloud (Fuente: Cloudoye.com)

2.7.3 *Community o Inter-Clouds*

La operatividad de las *Community Clouds* tiene su pertenencia a una o más organizaciones, cuyas funciones y servicios son comunes con el propósito de lograr una misión en particular o la colaboración entre grupos de interés. Dentro de este modelo podría estar un consorcio industrial, un grupo de concientización o algún otro tipo de agrupación. En algunos casos el *community cloud* se compone de una responsabilidad compartida, ya sea por temas de financiamiento o desde una perspectiva para el uso del recurso computacional. En otros

casos, un miembro de la comunidad ofrece todos los fondos y recursos con otros miembros contribuyentes, ya que en términos de elasticidad existe una ventaja evidente, en base a que el conjunto de recursos disponibles será algo mayor que en el privado (fig. 59). El factor determinante del *community cloud* es que los diferentes componentes están todos reunidos por una causa común (Halpert, 2011). Como las *community cloud* son provistas y administradas por consorcios unificados por un bien común, los miembros tienen cierta influencia sobre la calidad del servicio, sometiendo la última palabra de servicio a la decisión de la mayoría.

Figura 59. Modelo Community Cloud (Fuente: Skillsoft)

Algunos ejemplos de este modelo de implementación de tipo *Community* pueden ser los que se prestan a servicios de salud pública, conocidos en el extranjero como *healthcare community clouds* y que son empleados para mejorar el acceso hacia aplicaciones que contengan información crítica de carácter socio-sanitario, de igual manera también están los *clouds* de comunidad gubernamentales o *government community cloud*s que se emplean para facilitar el acceso a recursos de interoperabilidad y gestión de organismos públicos y entre administraciones de carácter públicas (Salazar, 2013).

2.7.4 *Hybrid Cloud*

Las nubes híbridas no son un modelo de despliegue separado, sino una mezcla de recursos de computación, provistos por dos o más formas de *Cloud* como los públicos, privados y comunitarios, cada uno con sus características específicas que los distinguen entre ellos y con estrategias distintas a la hora de gestionar las TIC, por lo cual continúan siendo entidades únicas interconectadas mediante tecnología estandarizada o propietaria (*Mell, Grance*, 2009), que permite la portabilidad de datos y aplicaciones, por ejemplo, una aplicación SaaS privada que se basa en un IaaS público o el rebalanceo de cargas entre nubes (Salazar, 2013).

Las nubes híbridas suelen centrarse en el aprovechamiento de las economías de escala presentes en las ofertas de la nube pública, sino que también se enfoca en manejar cargas de trabajo que tienen requerimientos más estrictos de calidad de servicio (QoS). Por esta razón, en muchos casos las nubes híbridas son nubes internas, que recurren a la capacidad de las nubes públicas para solventar picos de demanda o reducir costos, el control del escenario privado obviamente lo tendrá la empresa por confidencialidad (Halpert, 2011).

La nube híbrida contiene dos elementos: el *Cloud File Server* y el *Local Cloud*. El *Local Cloud* está conectado al almacenamiento en las premisas de la sede (disco duro del ordenador, el almacenamiento conectado a la red o el dispositivo virtual que se ejecuta en el almacenamiento unificado) para integrar los archivos y servidor de archivos de la nube permitiendo a los usuarios acceder a los archivos en cualquier momento y en cualquier lugar a gran velocidad (Cloudoye, 2015). Capacidades de acceso móvil y un mayor intercambio de archivos son los principales atributos de este modelo de computación en la nube y que puede satisfacer las necesidades de las organizaciones, independientemente de su tamaño y de la naturaleza del negocio. Al permiten a las empresas externalizar las nubes públicas y privadas, gran parte de la infraestructura de los proveedores de la nube perderían el control sobre los recursos y la gestión de la distribución de código y datos que se utilizan (fig. 60). En muchos casos, esto no es deseado por las organizaciones que brindan estos servicios de *cloud computing* (Ullauri, 2013).

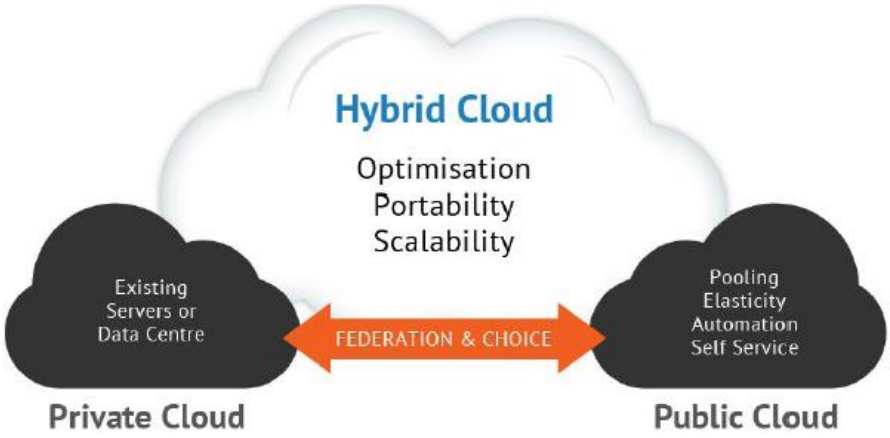


Figura 60. Beneficios claves del *Private cloud* (Fuente: Cloudoye)

Según el artículo "La nube híbrida domina las estrategias de negocio" (Computing, 2016), se realizó un estudio independiente sobre la adopción de la nube híbrida o "nube desplegada en múltiples niveles" en colaboración con *EMC, Atos, Canopy*, donde se considera que al parecen la nube se ha convertido en la plataforma de referencia para muchas aplicaciones, con más de 150 empresas de varios países, muestran un aumento en los servicios de nube híbrida, debido a la variedad en el conjunto de servicios combinados entre lo público y privado:

- Más del 58% de los encuestados afirma que está dentro de un entorno de tipo *Hybrid Cloud*.

- El 51% de los encuestados opina que *Hybrid Cloud* es una herramienta con suficiente potencial para permitir la consolidación de una gestión integrada de todos los servicios que se ofrecen en la nube.

- El 55% de los encuestados apunta que los factores más importantes a la hora de desarrollar una solución de nube híbrida son la monitorización y la optimización (gestión de servicios), esto favorece que la organización y la negociación de servicios

pasen a formar parte del núcleo principal de las estrategias de TI, debido a los costes de la nube, factores a los que el 49% sumó el rendimiento.

- El 46% de los participantes opina que rehacer las aplicaciones a escala es el escenario de implantación de *Hybrid Cloud* más probable.

2.8 CLOUD COMPUTING SERVICES LAYERS

Algunos expertos del *Future Trends Forum* opinan que no debe entenderse como una nueva tecnología, sino como un "Modelo operacional que viene determinado por el modo en el que un negocio o un individuo obtienen acceso a funciones de información" (Melendez, 2012). Los modelos de servicios de *Cloud* han proporcionado un medio para poner a disposición un servicio en la nube para clientes, ya que conceptualmente se basa en capas de servicios y cada capa provee un nivel de funcionalidad distinto (fig. 61).

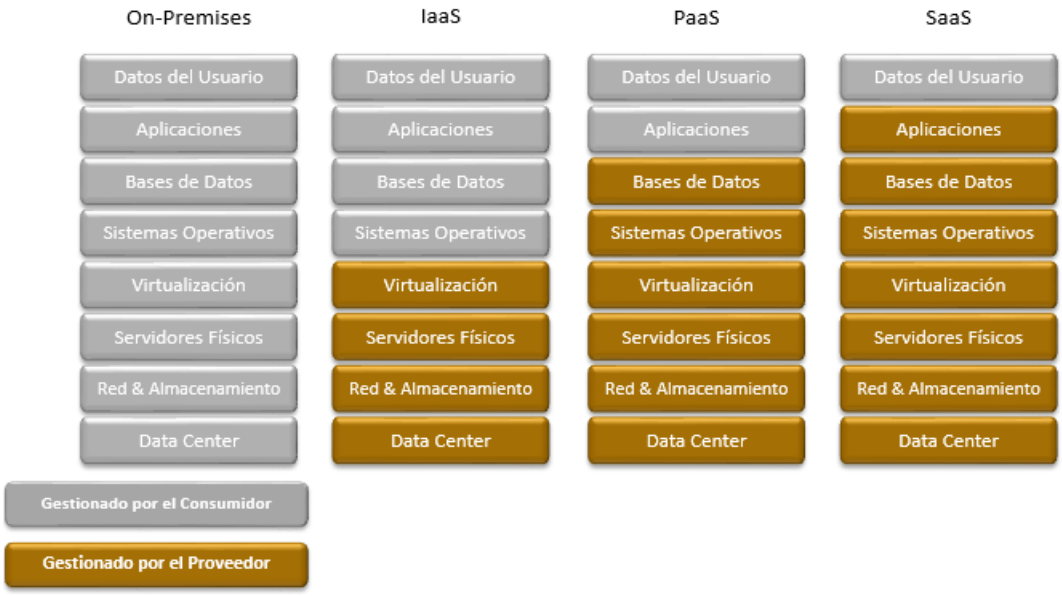


Figura 61. Modelos de servicio de *Cloud Computing* (Fuente: Cisco)

La mayoría de los modelos fundamentales de servicio incluyen una combinación de IaaS (infraestructura como servicio), PaaS (plataforma como servicio) y SaaS (software como servicio) (Gorelik, 2013). El tipo de servicio que se presta tiene muchas implicaciones en el proveedor, incluyendo la forma en que direccionan asuntos concernientes a la seguridad, la resistencia, el cumplimiento y el *multi-tenancy* (fig. 62).

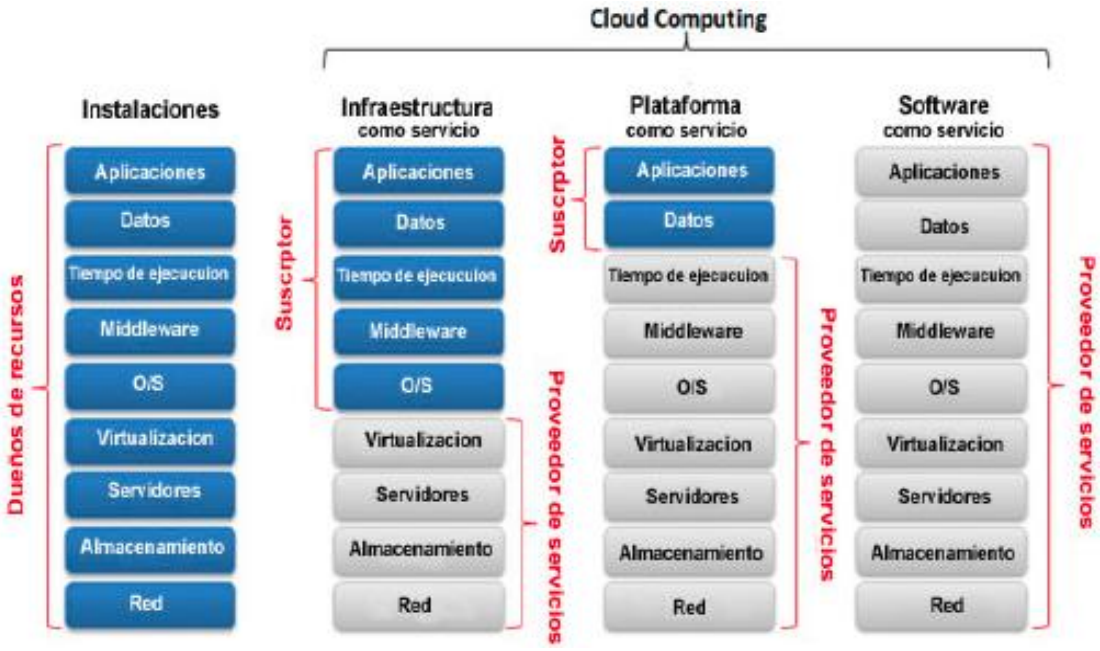


Figura 62. Responsabilidades de los modelos de servicios de *cloud computing* (Fuente: http://blogs.cisco.com/security/penetration-testing-in-the-cloud/)

Estos modelos de servicios pueden tener sinergia entre sí y ser interdependientes, por ejemplo, el servicio de PaaS depende de IaaS (fig. 63), porque las plataformas de aplicaciones requieren infraestructura física (Halpert, 2011).

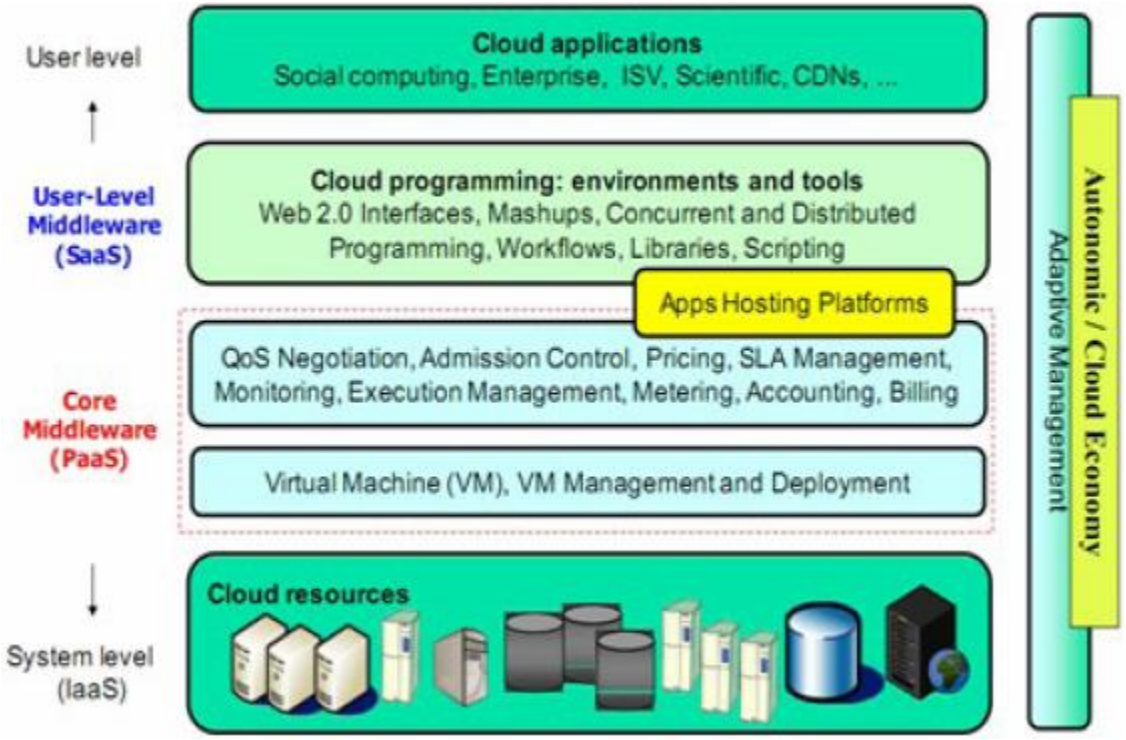


Figura 63. Arquitectura Layered de Cloud Computing (Fuente: Wang Y.)

2.8.1 *Infrastructure as a Service* (IaaS)

Los proveedores que utilizan esta categoría de servicio de computación en la nube conocido como *Infrastructure as a Service* (IaaS) permiten a sus clientes el acceso a diferentes tipos de infraestructura y sus componentes que pueden incluir máquinas virtuales, almacenamiento, redes, *firewalls*, *routers*, *load balancers* y otros más. El proveedor normalmente proporciona este servicio dividiendo una gran cantidad de recursos de infraestructura físicos en recursos virtuales más pequeños para que sean accesados por el consumidor. A veces el servicio prestado es una máquina virtual completa con un sistema operativo, pero en otros casos, el servicio prestado es simplemente para el almacenamiento, o también puede tratarse de una máquina virtual de tipo *bare* sin sistema operativo. En los casos en que se incluye el sistema operativo u otro software, el costo de la licencia requerida debe estar implícita en el costo que se cobra por el servicio, o debe incluirse como un

suplemento. Al emplear IaaS los clientes tienen acceso directo al software de nivel más bajo del *stack*, que es el sistema operativo en máquinas virtuales o para el tablero de gestión del firewall o el balanceador de carga.

Los proveedores de IaaS son con frecuencia proveedores de servicios de otros *Cloud Providers* CP (o conocidos por su rol de *Integrators*), por esta razón se acelera el crecimiento, ofreciendo nuevas fuentes de ingresos (fig. 64). Muchos proveedores actuales de *Platform as a Service* apoyan los servicios que prestan los proveedores de IaaS para solventar la demanda de capacidad adicional. Uno de los proveedores más populares IaaS es Amazon, que ofrece su EC2 IaaS.

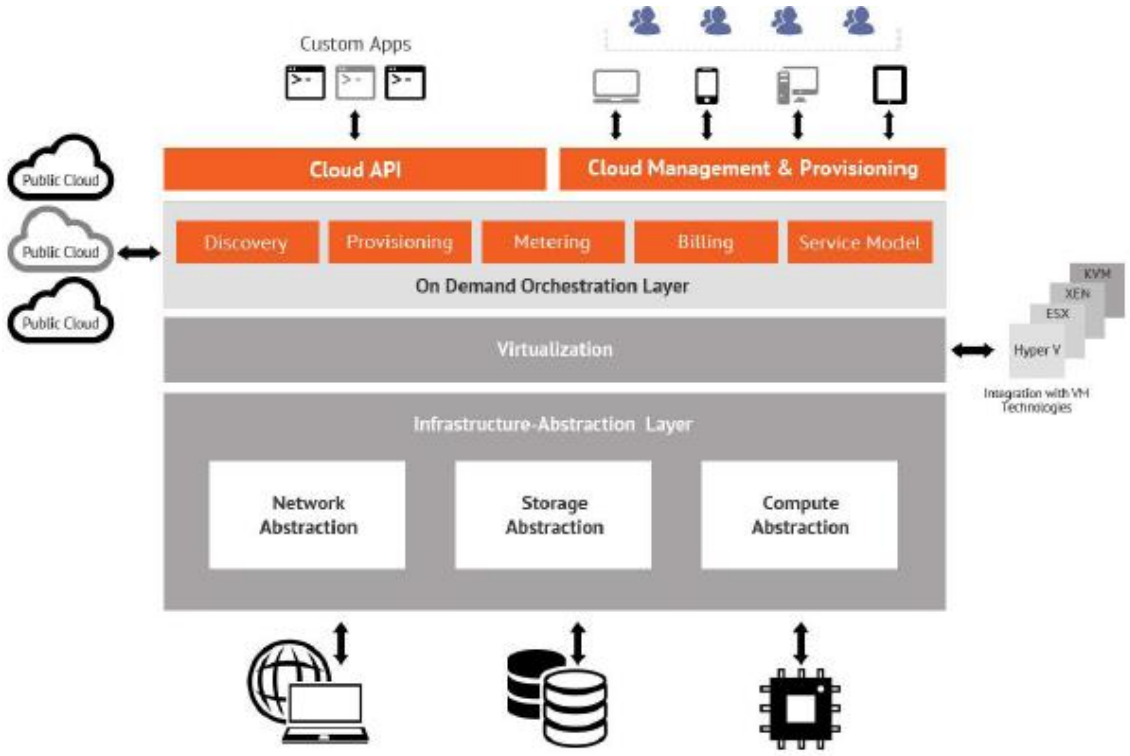


Figura 64. *Infrastructure-as-a-Service* (IaaS) acelera el crecimiento y la entrega de nuevos recursos para *streams* (Fuente: Cloudoye)

Dentro de IaaS se puede mencionar algunos de los servicios de infraestructura más importantes:

- ***Cloud Storage***

Es el tipo de almacenamiento localizado en *Datacenters* públicos o privados, totalmente separado del almacenamiento primario o local. Su implementación se realiza a través los servicios orientados a la arquitectura (SOA). Cuenta con dos formas para accesar al *storage*, la primera es accesando directamente por medio de archivos o bloques, y la segunda es accesando indirectamente por medio de aplicaciones que se encuentran alojadas conjuntamente con el *storage* (Kamaraju & Nicolas, 2009).

A continuación se indicará las dos tecnologías SAN y NAS que proporcionan métodos de almacenamiento:

- ***Storage Area Network* (SAN)**

Es una red de almacenamiento integral, con una arquitectura completa que combina elementos como una red de alta velocidad mediante un canal de fibra óptica, un equipo de interconexión dedicado y elementos de almacenamiento (Revelo M, 2013).

- ***Network Attached Storage* (NAS)**

NAS es el tipo de tecnología de almacenamiento dedicada a compartir la capacidad de almacenamiento de un computador con ordenadores de la misma red, empleando un sistema operativo optimizado para dar acceso con protocolos específicos como (CIFS, NFS, FTP o TFTP) (Mendoza, 2007).

Como ejemplo general, se pondrá a consideración a los servicios de infraestructura *Cloud* que ofrece *Amazon Web Services* (AWS) al ser uno de los más importantes CCSP (fig. 65) (Gorelik, 2013):

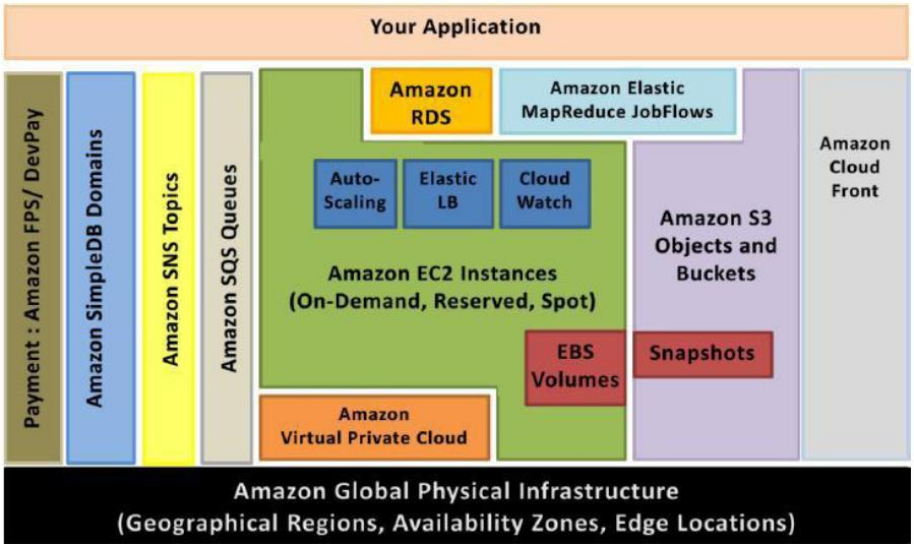


Figura 65. Servicios *Cloud de Amazon Web Services* (Fuente: Gorelik E.)

- ***Amazon Elastic Compute Cloud*** **(EC2)**

Es un servicio web clave que proporciona una facilidad para crear y gestionar instancias de máquinas virtuales con sistemas operativos que se ejecutan dentro de ellos. Hay tres maneras de pagar por las instancias de máquinas virtuales EC2, y las empresas pueden elegir la que mejor se adapte a sus necesidades. Una instancia bajo demanda (*on-demand*) ofrece una máquina virtual (VM) siempre que lo necesite y termina automáticamente. Un ejemplo de servicio por reservada (*reserved*) permite al usuario comprar una VM y pagar por adelantado por un cierto período de tiempo. Un ejemplo de servicio punto (*spot*) puede ser adquirido a través de licitación y sólo se puede utilizar siempre y cuando el precio de la licitación sea superior a los demás (Jinesh, 2010). Otra característica conveniente de los servicios en la nube que ofrece Amazon, es que permite el alojamiento a través de múltiples ubicaciones

geográficas, ayudando a reducir la latencia de la red de la localidad estratégica del cliente que requiere el servicio.

- ***Amazon Relational Database Service (RDS)***

Proporciona servicios de bases de datos MySQL y Oracle en la nube (Gorelik, 2013).

- ***Amazon S3***

Es un servicio de almacenamiento en la nube rápida y redundante, que proporciona el acceso del público a los archivos a través de HTTP.

- ***Amazon SimpleDB***

Es un tipo de servicio veloz para bases de datos, por ejemplo el no estructurado NoSQL.

- ***Amazon simple Servicio Queuing (SQS)***

Proporciona un mecanismo de cola fiable, sobre el cual los desarrolladores de aplicaciones pueden encolar diferentes tareas de procesamiento en segundo plano (Gorelik, 2013).

Todos estos ejemplos de avances tecnológicos, facilitaron el desarrollo del *Cloud Computing* incluyendo un incremento en la disponibilidad de hardware básico, a través de las más altas velocidades de red, con aplicación constante de virtualización, teniendo en cuenta los diferentes enfoques en la arquitecturas de su aplicación (fig. 66), innovando constantemente en las arquitecturas del almacenamiento de los datos y con una impresionante calidad para el acceso continuo a la red (Gorelik, 2013).

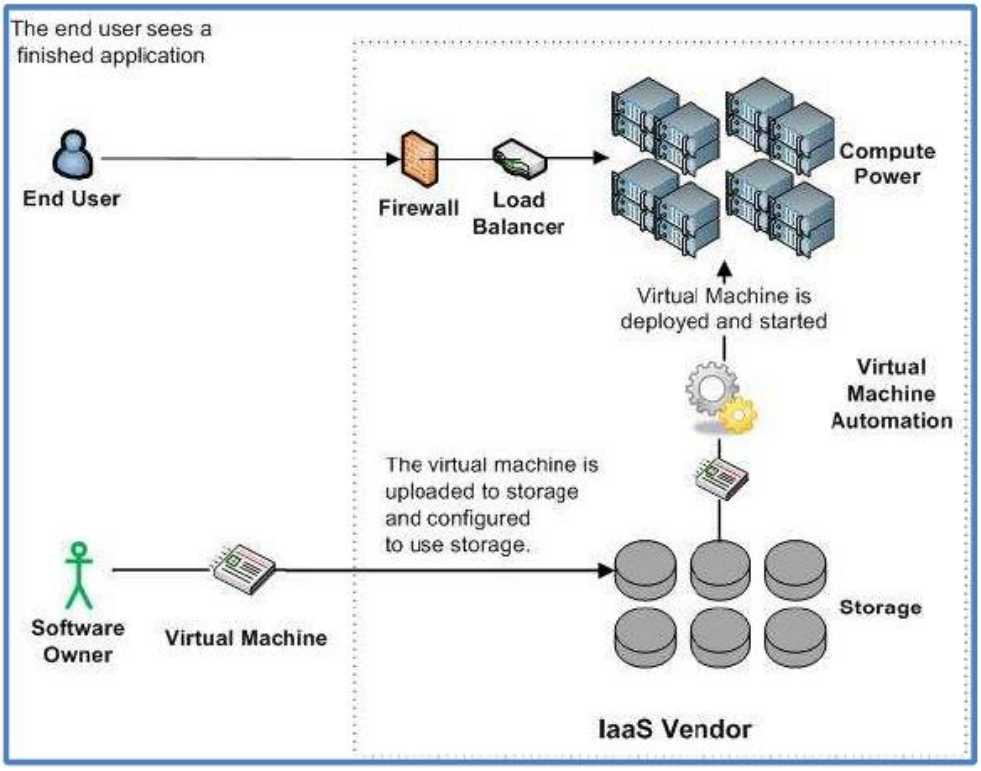


Figura 66. Esquema funcional de *Infrastructure-as-a-Service* (IaaS) (Fuente: Figueroa C.)

2.8.2 *Platform as a Service* (PaaS)

"El *cloud computing* ha evolucionado hasta incluir plataformas para crear y ejecutar aplicaciones personalizadas basadas en web, un concepto que es conocido como Plataforma como Servicio" (*Rittinghouse*, 2010, pág. 48).

En este modelo de *Platform as a Service* (PaaS) se ofrece una plataforma de aplicaciones pre-construidas para el cliente. Los clientes no necesitan perder tiempo en la construcción de una infraestructura subyacente para sus aplicaciones, ya que los proveedores de esta categoría extienden el *stack* del software proporcionado por IaaS para incluir el *middleware*. El *Middleware* genéricamente se refiere al software como una base de datos DB2, un entorno de ejecución tal como el *Java Runtime Environment* (JRE) o un servidor de

aplicaciones *Websphere*. Este *middleware* es un requisito previo para la ejecución de aplicaciones más sofisticadas y poder proporcionar un entorno operativo más enriquecido para las aplicaciones que se utilizarán en esta capa (Halpert, 2011). Un entorno de tiempo de ejecución que no está constituido de manera sólida o una base de datos mal configurada puede hacer que un usuario afecte negativamente a la calidad de servicio de otros usuarios.

PaaS escala y aprovisiona automáticamente componentes de la infraestructura requeridos en función de los requisitos de aplicación (fig. 67). Por lo general, las soluciones PaaS proporcionan un API y plantillas que incluyen un conjunto de funciones para construir, ejecutar, evaluar y administrar su software de programación en la plataforma y el desarrollo de soluciones, proveyendo acceso al sistema operativo y servicios asociados, permitiendo desarrollar aplicaciones para la nube con herramientas soportadas por el proveedor, con el respectivo mantenimiento de hardware y software que soporta la funcionalidad de PaaS (Gorelik, 2013).

Figura 67. Solution stack for PaaS (middleware, database, applications) (Fuente: Cloudoye)

Los proveedores de PaaS tienen dos métodos con los que facilitan la capacidad adicional necesaria para un gran sistema *multi-tenant*. También constan de una arquitectura virtualizada de *Grid Computing* la cual es la base de dicho software de infraestructura. En algunos casos, proporcionan máquinas virtuales de estilo IaaS para el consumidor.

En otros casos, proporcionan una interfaz a través de la cual las aplicaciones pueden ser cargadas en el caso de un entorno de ejecución o al cargar los datos en el caso de una base de datos. El cliente puede administrar la implementación de aplicaciones, sin el coste y la complejidad derivados de la compra, ya que facilita el despliegue de las aplicaciones, *frameworks*, gestión del hardware y de las capas de software asociadas. De esta manera se aclara que los usuarios de PaaS son típicamente los desarrolladores de software que son aplicables a la plataforma y las ofrecen a los usuarios finales (Ullauri, 2013).

Cada método tiene sus ventajas y desafíos. Con un enfoque de estilo IaaS, el proveedor suele tener más control y separación más fuerte entre los *tenants*. Este enfoque es menos eficiente, sin embargo, con la sobrecarga común, tales como el sistema operativo y la propia máquina virtual se duplican a través de los *multiple tenants*.

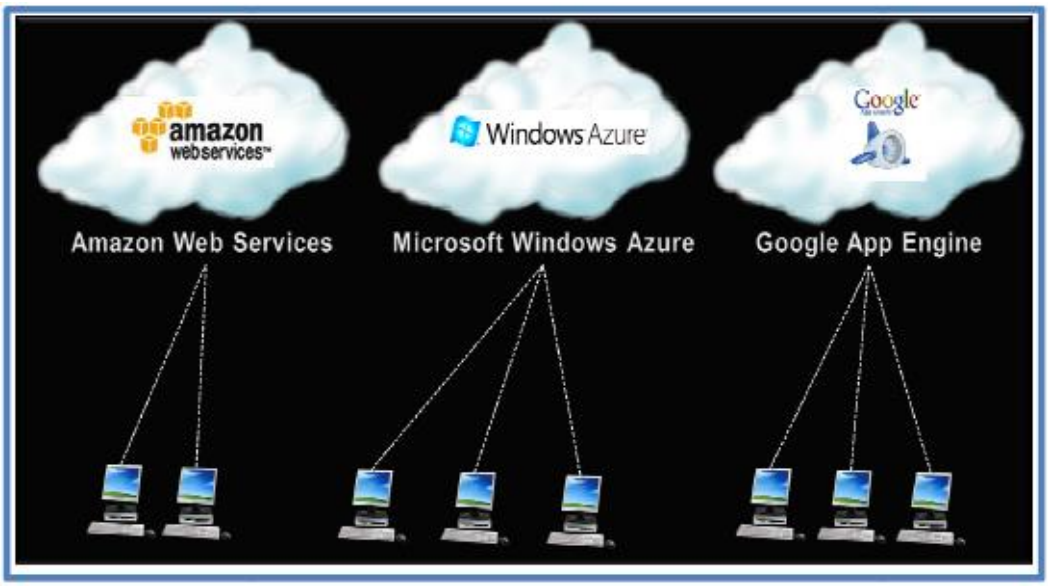


Figura 68. Proveedores de servicios *PaaS* (Fuente: Figueroa C.)

En el segundo caso, la infraestructura subyacente es direccionada de manera más eficiente, con una sola imagen del sistema y la sobrecarga del *middleware* se aplacará entre varios clientes. Por el contrario, el principal reto con este enfoque radica en el grado de separación que puede ser proporcionado entre los *tenants* (Halpert, 2011).

Como algunos de los ejemplos más conocidos de empresas que ofrecen servicios de PaaS están Microsoft con su plataforma de *Windows Azure*, al igual que *Google App Engine* con su servicio muy popular de PaaS y otro de los grandes que es *Amazon Web Services* que también ofrece algunas soluciones PaaS además de las que brinda para IaaS (fig. 68).

2.8.3 *Software as a Service* (SaaS)

Esta categoría *de Software as a Service* (SaaS) es conocida también como *Application as a Service*, y es aquella que provee una solución lista de software en línea o interfaz basada en la web para sus clientes. El cliente, en la mayoría de los casos, está completamente abstraído de la aplicación que se ejecuta detrás de escena. La separación de los *tenants* se hace a menudo en la capa de aplicación, dejando a la vista una capa común de aplicación, plataforma y de infraestructura (fig. 69).

Los proveedores de SaaS tienen el control completo de software que aloja la aplicación y suelen aumentar la capacidad de sus sistemas a través del escalamiento de los métodos vertical u horizontal en función de las características de la aplicación. Aplicaciones SaaS que escalan verticalmente suelen ser trasladadas a plataformas más grandes a medida que crecen sus necesidades de capacidad. Aplicaciones SaaS que escalan horizontalmente se ejecutan en grandes grupos de servidores. Como se requiere una capacidad adicional, el proveedor añade máquinas adicionales al clúster (Halpert, 2011).

Figura 69. Proveedores de servicios PaaS. (Fuente: Cloudoye)

Como hay una cantidad significativa de recursos compartidos utilizados entre los *tenants* en un entorno SaaS, la capacidad de que un *tenants* afecte la calidad del servicio de los otros *tenants* es siempre una preocupación. La capacidad de un proveedor de SaaS para acomodar o aislar a un *tenants* del otro es la clave para mantener la calidad del servicio. La principal diferencia entre SaaS y PaaS es que PaaS normalmente representa una plataforma para el desarrollo de aplicaciones, mientras que SaaS provee de aplicaciones en línea que ya han sido desarrolladas o también licencias de su aplicación, otorgando un menor grado de administración, control y propiedad al usuario final con respecto al software que va a ser desplegado, incluyendo las configuraciones posibles sobre el mismo y que estén listas para el uso (Gorelik, 2013).

Ejemplos de aplicación SaaS incluyen correo en línea, sistemas de gestión de proyectos, servicios de ofimática en *cloud* como Office 365, Oracle CRM, ERP, las sociedades de gestión y plataformas de medios o redes sociales. Es común que SaaS sea implementado para proveer funcionalidad de software a los clientes o usuarios a un bajo costo, al mismo tiempo, los clientes obtienen los mismos beneficios que se generan al utilizar una aplicación de licencia comercial, pero sin la necesidad de instalarlo (Melendez, 2012). Los ejemplos populares de SaaS incluyen Google Apps y Salesforce.com (fig. 70).

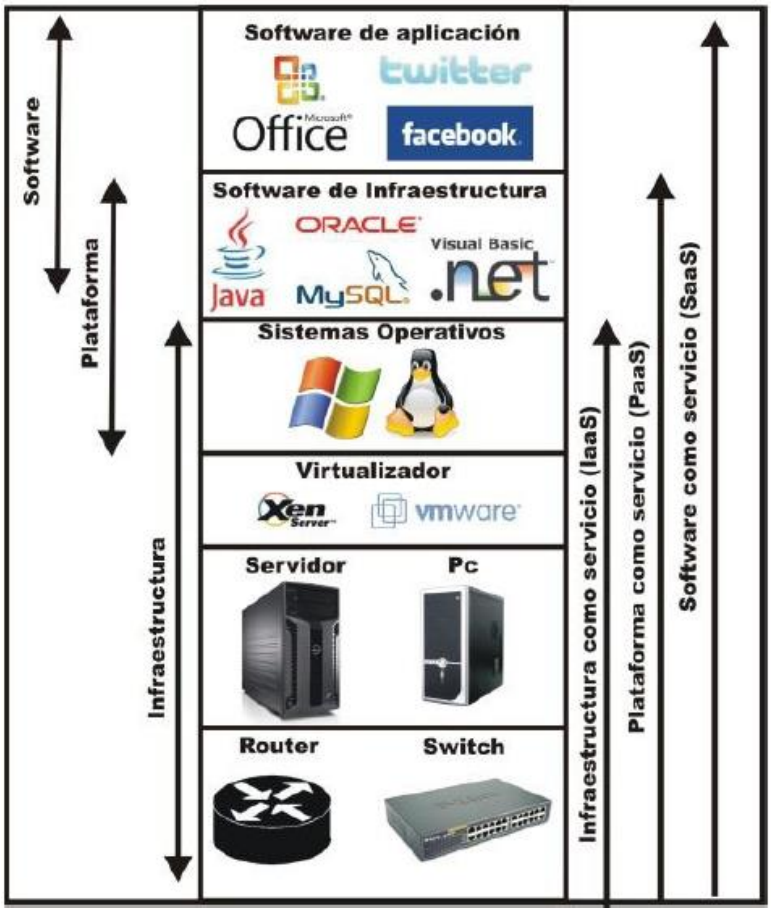


Figura 70. Servicios de Cloud Computing por capas. Fuente: http://www.katescomment.com/iaas-paas-saas-definition/

2.8.4 *Business Process as a Service* (BPaaS)

Esta categoría de *cloud computing* consiste en el aprovisionamiento de recusos, como lo hacen los servicios durante los procesos de negocio *end-to-end* altamente estandarizados a través de su entrega dinámica, mediante una modalidad de pago por uso o bajo demanda. Hay una razón práctica para seleccionar un modelo de *business process service*. En primer lugar, una organización puede seleccionar un proceso que coincide con la política empresarial, por lo que se puede entonces utilizar en muchos entornos de aplicaciones diferentes y esto asegura que exista un bien definido y consistente en toda la organización.

Por ejemplo, una empresa puede tener un proceso complejo para el procesamiento de la nómina o la gestión de envío. Este servicio puede estar vinculado a otros servicios en la nube, como SaaS, así como a las aplicaciones del centro de datos (Salazar, 2013).

- Un servicio BPaaS debe tener APIs bien definidas para que se pueda conectar fácilmente a los servicios relacionados, ya que los recursos utilizados mediante esta solución facilitan la ejecución de procesos de negocio y puedan ser compartidos entre los diferentes clientes del proveedor. En muchos casos este hecho proporciona un aporte de valor al esquema del negocio; pero sin embargo, la solución BPaaS es un modelo de negocio en el que los proveedores todavía tienen su negocio dentro de nichos concretos.

2.8.5 *IT as a Service* (ITaaS)

La categoría de IT como servicio representa un modelo de servicios que la empresa contrata a un proveedor para obtener conectividad de red, *backup* de red, recuperación de desastres, VPN, conferencias web, etc. Muchos autores prefieren alojar este nivel entre los otros tres y no crear uno nuevo debido a que el servicio que se presta es una combinación de los tres anteriores (Revelo, 2013).

Sus principales características permitir una inversión mínima en procesos de TI, gastos previsiblemente regulares, ventajas fiscales, monitorización continua de los servicios, apoyo técnico de expertos, escalabilidad, actualizaciones regulares de software y parches y garantía de hardware. Donde la base para implementar estos servicios son la virtualización y el *multitenant*. La virtualización permite que los recursos de la red estén disponibles siempre independientes de su ubicación física o la conexión física a la red (fig. 71). El *multitenant* se refiere al principio de arquitectura que permite el intercambio de recursos entre un gran número de usuarios.

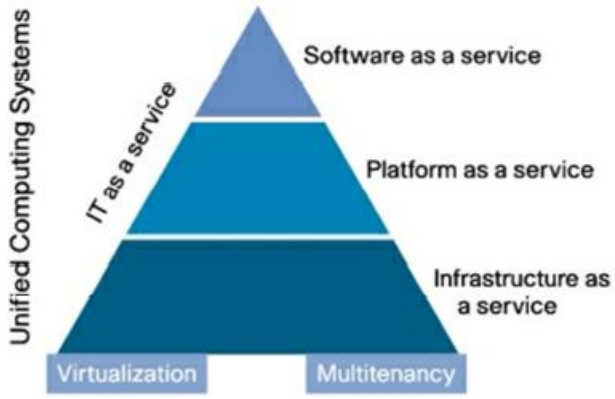
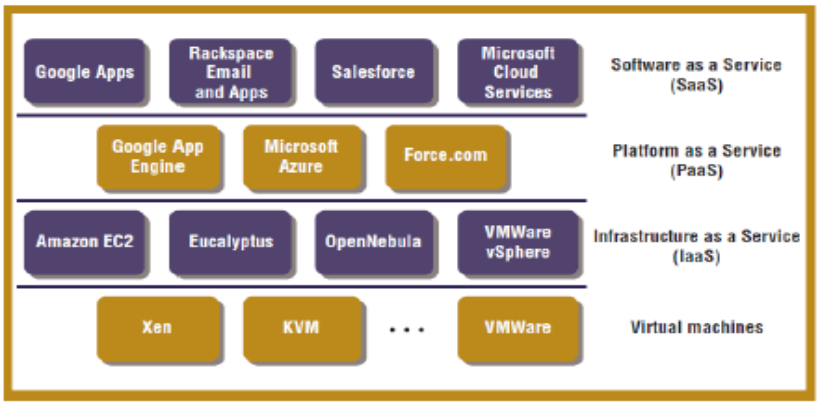


Figura 71. Servicios de Cloud Computing unificado. (Fuente: Revelo M.)

2.8.5 *Anything as a Service* (XaaS)

La categoría de *Anything as a Service* (XaaS) es la entrega de Tecnología de la Información TI como un servicio a través del modelo de *Hybrid Cloud Computing* y hace referencia a una o la combinación de las categorías de *Software as a Service (SaaS), Infrastructure as a Service (IaaS), Platform as a Service (PaaS), Communications as a Service (CaaS) or Monitoring as a Service (Maas)*. XaaS está emergiendo rápidamente como un término que está siendo reconocido fácilmente como los servicios que fueron separados previamente en los modelos de *private o public Clouds* y que de momento se están haciendo transparentes e integrados para el usuario (fig. 72).



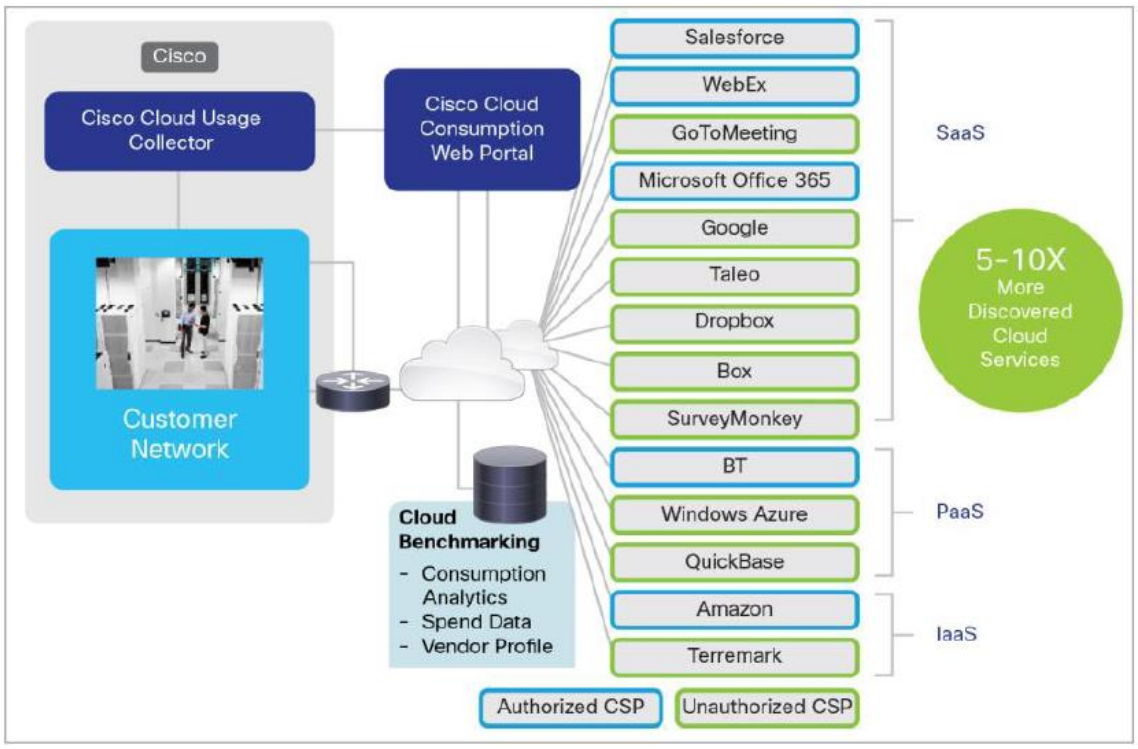


Figura 72. Servicios de Cloud Computing con proveedores (Fuente: Cisco)

2.9 ROLES DENTRO DEL SERVICIO DE CLOUD COMPUTING

Se definen tres roles claves dentro del paradigma de *cloud computing*. Estas funciones tienen diferentes responsabilidades y expectativas la una respecto a la otra, pero se debe entender que cualquiera de las partes puede tener múltiples roles en función del contexto del servicio, como se muestra en la figura 73.

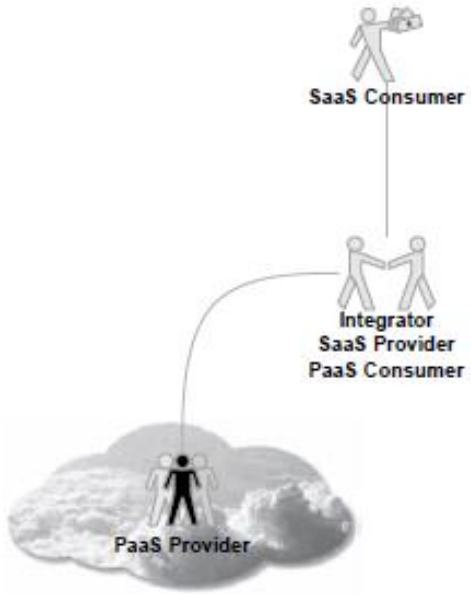


Figura 73. Roles claves dentro de Cloud Computing (Fuente: Halpert B.)

- **Consumidor**

Para el caso del consumidor, se sobre entiende que es el que consume cualquier servicio que se proporcione. Como se muestra en el ejemplo, el proveedor de SaaS expone un SOS al consumidor SaaS. El consumidor se permite el acceso a este servicio por una tarifa de algún tipo, aunque en muchos casos esta tasa se ve aumentada o reemplazado a través de los ingresos por publicidad (Figueroa, 2013). El consumidor no tiene ninguna responsabilidad, ni el acceso más allá de los SaaS que se les prestan.

- **Proveedor**

Los proveedores en este caso son tanto el proveedor de PaaS y el proveedor de SaaS. El proveedor de PaaS proporciona un servicio de PaaS al proveedor de SaaS. El proveedor de SaaS a su vez proporciona un servicio de SaaS para el consumidor (Figueroa, 2013). En última instancia, el proveedor es cualquier persona que preste un servicio a uno o más consumidores.

- **Integrador**

El rol de un integrador se refiere a veces al de un corredor (*broker*), que esencialmente reúne los servicios de muchos proveedores bajo el modelo de un nuevo servicio. En algunos casos, esto puede implicar la integración de varios proveedores de un mismo servicio, por ejemplo, cuando hablamos de la integración de varios proveedores de IaaS para proporcionar un servicio de IaaS más robusto o con todas las funcionalidades de un proveedor de clase. De acuerdo al ejemplo del gráfico 74, el integrador puede consumir el servicio de otro proveedor (en este caso PaaS) con el fin de ejecutar un servicio propio. El servicio del integrador se expone en última instancia como un servicio de SaaS para el consumidor que solicita el SaaS. Dependiendo de la perspectiva, veremos que cada parte puede tener varios roles (Figueroa, 2013). El proveedor de SaaS es en última instancia un consumidor del proveedor de PaaS y un integrador del servicio PaaS con sus funcionalidades de SaaS.

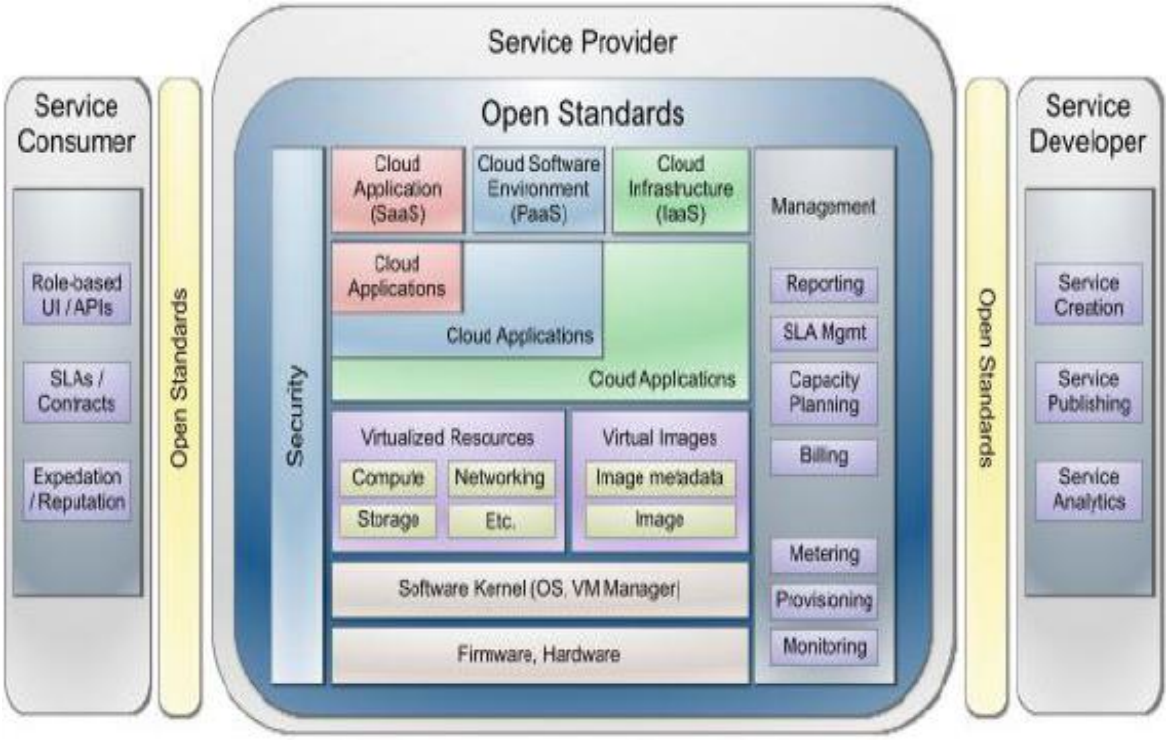


Figura 74. Taxonomía de *Cloud Computing - Use Case Discussion Group*. (Fuente: http://outsourceando.blogspot.com/2011/05/jornada-clud-computing-en-esic-madrid.html)

2.9.1 Soluciones Basadas en Cloud Computing

Las soluciones de *cloud* están disponibles en todos los niveles de TI de una empresa, actualmente existe una gran cantidad de proveedores de servicios de *cloud computing*, quienes cubren servicios de software, plataforma e infraestructura (fig. 75). *OpenCrowd* (2015) ha creado una taxonomía del *cloud computing* para proveer información de los servicios en el *cloud* y crear un dialogo entre los proveedores de servicios y los consumidores.

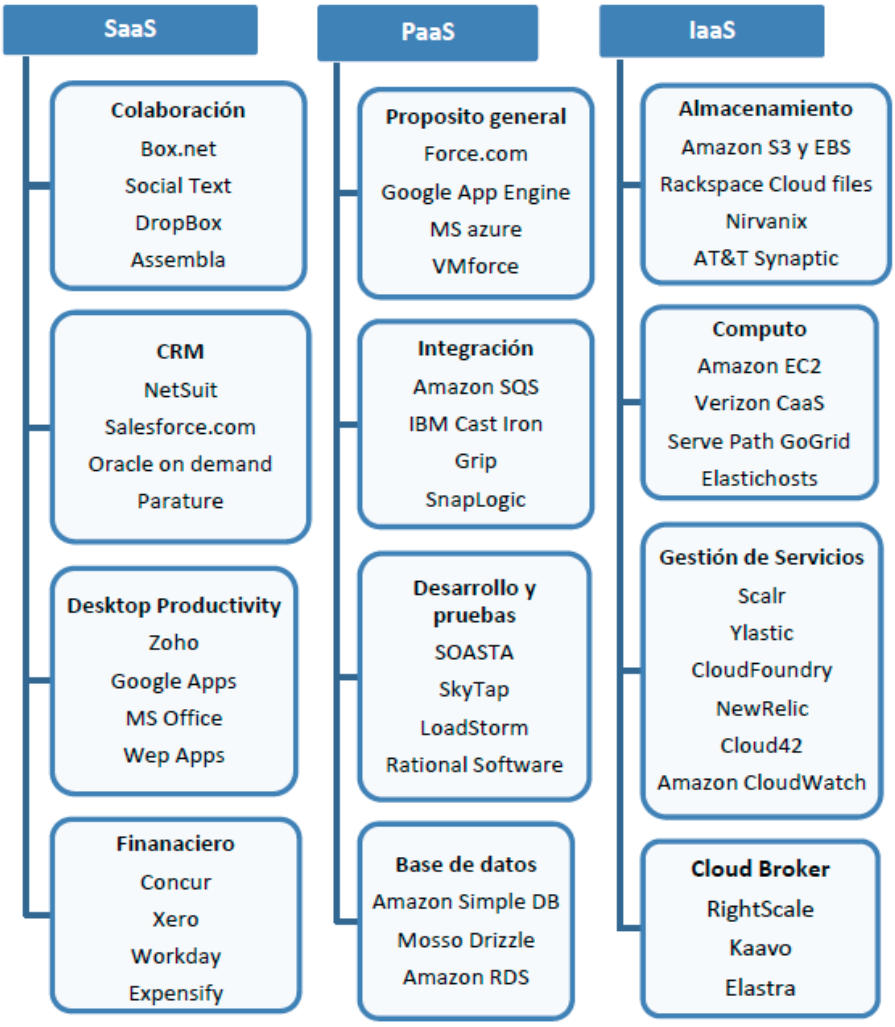


Figura 75. Soluciones basadas en *cloud computing* (Fuente: http://cloudtaxonomy.opencrowd.com/)

2.10 PROTOCOLOS EMPLEADOS EN CLOUD COMPUTING

- ***Representation State Transfer*** **(REST)**

Es protocolo con técnica de arquitectura de software para sistemas hipermedia distribuidos como la *World Wide Web*, que brinda métodos simples de acceso web, define las operaciones en recursos y en formatos de datos. Se basa en los principios o reglas de arquitectura de red, que utiliza HTTP para la transferencia de datos (Kamaraju & Nicolas, 2009).

- ***Simple Object Access Protocol*** **(SOAP)**

Es una especificación de protocolo de intercambio de información estructurada en la aplicación de Servicios Web en redes informáticas, que brinda métodos simples de acceso web, usa XML para la comunicación en Internet, específico para el intercambio estructural de información. Ocupa un lenguaje multiplataforma, utiliza HTTP o RPC para la transferencia de datos (Kamaraju & Nicolas, 2009).

- ***Web Services Description Languaje*** **(WSDL)**

Es un lenguaje basado en el formato XML que provee un modelo para describir servicios de red, como un conjunto de puntos finales que contengan información de tipo *document-oriented* o *procedure-oriented*. Los servicios están definidos en términos del nombre, protocolo, parámetros y tipos de datos como SOAP 1.1, HTTP GET/POST y MIME.

- ***Universal Description, Discovery and Integration*** **(UDDI)**

Es un estándar básico de los servicios web, diseñado para ser interrogado por mensajes SOAP y proveer acceso a documentos de WSDL en los que se describe los requisitos del protocolo (Mendoza, 2007).

- ***Common Internet File System*** **(CIFS)**

Protocolo muy estable, inventado originalmente por IBM, igual que SAMBA este protocolo es basado en SMB (*Server Message Block*), define las operaciones de transferencias (Leung et al, 2008).

Además *Cloud Computing* utiliza protocolos ya conocidos como HTTP, FTP, RPC26, TCP, IP, DNS, SNMP entre otros. Cada protocolo ayuda a la implementación de servicios robustos de *Cloud*.

2.11 SEGURIDAD EN CLOUD COMPUTING

El desafío más importante que debe superar *Cloud Computing* es la seguridad y ahora con mayor razón que está ganando terreno en los negocios, con esta premisa es necesario conocer acerca de algunos de los riesgos clave de seguridad que presenta. Es conocido que el activo más importante de una empresa es la información, por lo que es primordial salvaguardar la integridad de datos, la recuperación, la confidencialidad y una evaluación de las preocupaciones legítimas en áreas, tales como *e-discovery*, monitoreo y apreciativa (Cloudoye, 2015).

Por otra parte, los usuarios en línea que manejan asuntos del negocio dentro del *Cloud*, crean aplicaciones y mueven datos con información de misión crítica sin entender todas las implicaciones de seguridad. Se describirá algunos de los riesgos de seguridad percibidos en el *cloud computing* y algunas de las principales soluciones para mitigar los riesgos, para así poder desarrollar una infraestructura de TI segura y holística.

Como se ha mencionado, una nube se constituye básicamente de aplicaciones y datos de misión crítica del negocio y si cualquier invasor penetra lo suficiente, entonces no será posible conocer con seguridad cómo serán expuestas muchas piezas sensibles de la información. Además de esto, si no se está claro sobre los parámetros básicos de seguridad

en una base de datos de algún servicios en la nube de tipo *multi-tenant*, con un obstáculo de seguridad sobre uno de los usuarios del *multi-tenant*, se podría permitir inconscientemente un acceso no autorizado a cualquier invasor y no sólo a los datos de ese usuario, sino que también se podría comprometer los datos de cualquier otro usuario. La integración hacia claves de cifrado podría proteger los datos en reposo, pero este método también es susceptible a pérdida de información en caso de que se pierda inesperadamente la clave de cifrado, razón por la cual es de vital importación seguir las siguientes consideraciones (Revelo, 2013):

- El cifrado de datos es necesario ante posibles penetraciones de intrusos en el sistema.
- Cifrar todo tipo de datos, considetando que los datos pasarán por redes públicas.
- Requerir de un fuerte tipo de autenticación entre aplicaciones.
- Poner atención al manejo de la criptografía y estar actualizados en los algoritmos.
- Manejar de manera segura los permisos y accesos concedidos a los usuarios.
- Es necesario que los proveedores sigan las mejores prácticas en encriptación, tanto en seguridad física como lógica.
- Los usuarios deben cifrar sus propios ficheros, no se debe permitir que sea el proveedor quien lo haga.
- Si es el proveedor de servicios quien vaya a proporcionar el cifrado de datos, se debe solicitar las pruebas del correspondiente esquema de cifrado y debidamente probado por un especialista, ya que se debe considerar que los accidentes de cifrado pueden hacer que los datos queden totalmente inutilizables.

Otros expertos afirman que, la mayoría de los problemas de seguridad en la nube son exactamente los mismos inconvenientes de seguridad asociados a la gestión de equipos que son de propiedad de la organización, con un gran inconveniente, como el que se describe en los distintos modelos de servicio que se describen a continuación:

Cuando se trata de SaaS, el centro de los problemas de seguridad se basa en el control de acceso y problemas de contraseña (uso de credenciales débiles), problemas con los protocolos de comunicaciones y defectos en las aplicaciones basadas en la Web (Cox, 2009). PaaS tiene problemas con la configuración de las aplicaciones incluidas en el paquete de servicios, la

implementación y el despliegue del protocolo SSL y problemas de permisos con los datos almacenados en la nube. Mientras que al utilizar los servicios de IaaS, el problema número uno de seguridad es la seguridad del sistema operativo subyacente y los servicios (Cox, 2009).

Así que, ¿cuál es el inconveniente para el control de estos riesgos en las aplicaciones de *cloud computing*? Una organización que requiere estos servicios por parte de un proveedor no puede controlar nada de esto directamente. Al ser un servicio arrendado, la organización puede como no puede tener inferencia en la implementación de la seguridad de los servicios arrendados. Sobre lo que una organización tiene el control y es la herramienta de seguridad número uno cuando se está trabajando con servicios de *Cloud Compunting*, es el acuerdo de nivel de servicios (SLA). Como se ha discutido, el contenido del acuerdo de servicio será el método principal de auditoría de la organización, ya que garantizará la seguridad de los servicios por el lado de la nube arrendada y el resto de sus componentes de servicio. Por esta razón, tener un control total y entendimiento de los acuerdos de servicio que se firma con el proveedor es la piedra angular de la gobernabilidad (Halpert, 2011).

- ***SLA Agreements***

Un *Service-Level Agreement* (SLA) es un contrato que describe el nivel de los servicios ofrecidos por un *Cloud provider*. En el caso de los servicios en la nube, SLA podría medirse en términos de tiempo medio entre fallos, el tiempo para reparar la falla y otras métricas operacionales tales como el tiempo de respuesta de la red y el rendimiento del sistema (Gorelik, 2013). Las empresas deben llevar a cabo la debida diligencia para examinar cuidadosamente los acuerdos de SLA entre sus proveedores de servicios. No todos los *cloud providers* quieren (o puede) ofrecer el nivel de continuidad de negocio que requieren las organizaciones. Incluso unos de los más grandes y representativos *cloud providers* como Amazon solo proporcionan un 99.95% de disponibilidad anual como garantía para sus servidores, mientras que algunas organizaciones requieren tiempo de actividad anual de un 99,99%.

- ***Data Breaches***

Las brechas de seguridad de los datos en línea llevan a la pérdida de información personal y de tarjetas de crédito, siendo esta una de las formas más comunes de robo y que suelen tener lugar en el momento de realizar una simple transacción en línea o al momento del almacenamiento de datos (fig. 76). La computación en nube ha traído muchas buenas soluciones y oportunidades de negocio, sin embargo también ha introducido nuevas posibilidades y formas de ataque, debido a que aún se tiene una gran interrogante con el manejo de la seguridad del *hypervisor* y las operaciones de virtualización sobre máquinas virtuales (VM) (Cloudoye, 2015).

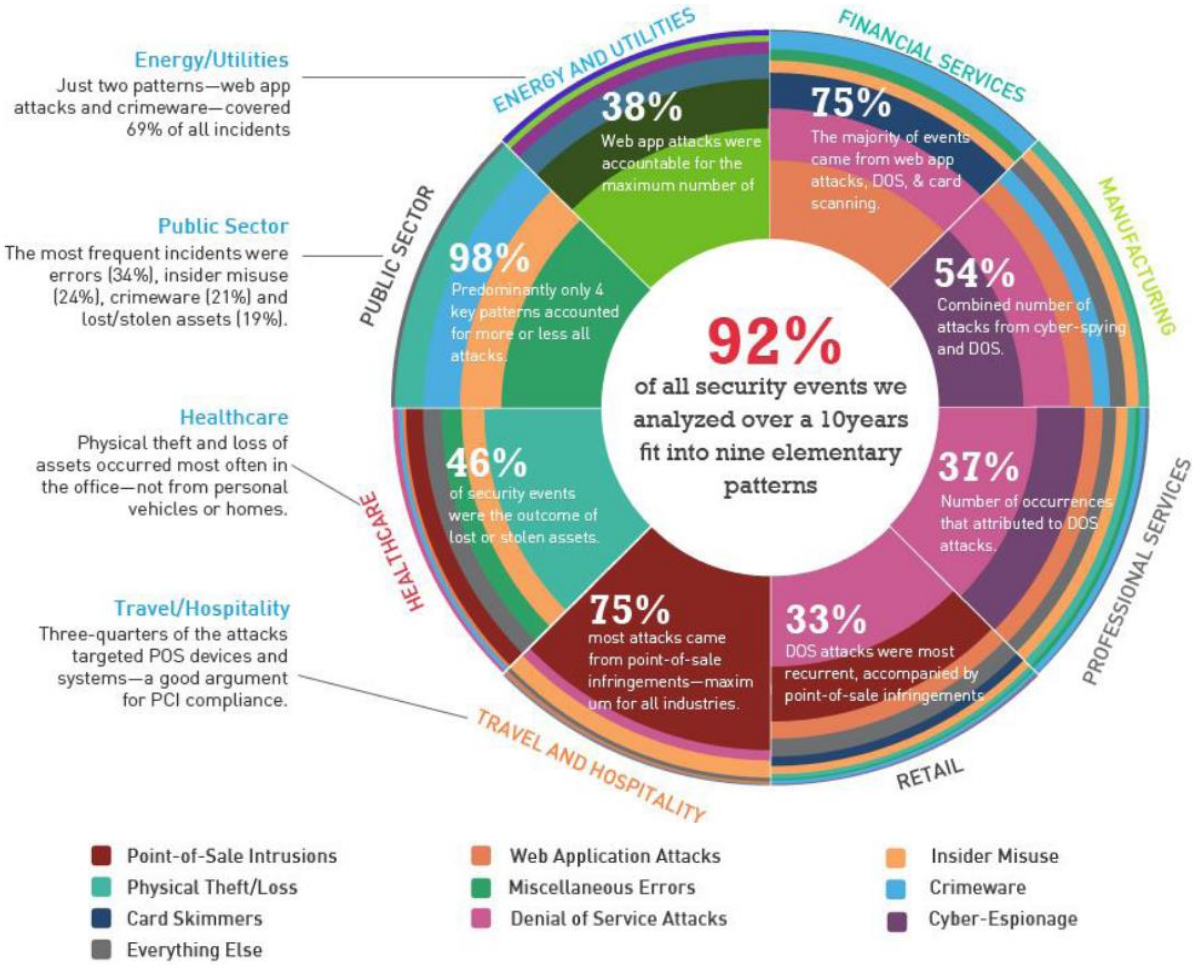


Figura 76. Data Breach Report 2014 - Classification of the key patterns that covered the mainstream of security incidents (Fuente: Cloudoye)

- ***Data Segregation***

Al ser el *Cloud* típicamente un entorno compartido, se debe tener en cuenta que los datos de los suscriptores deben estar separados y no deben ser afectados por las acciones en los datos de otros clientes.

- ***Data Location***

En ciertos casos los clientes necesitan saber la localización exacta de los datos por motivos de cumplimiento normativo, es por esto que al iniciar con un servicio en la nube, siempre será buena práctica preguntar a los proveedores si se van a comprometer a almacenar y procesar los datos en jurisdicciones específicas y si se va a hacer un compromiso contractual para obedecer los requisitos de privacidad locales en nombre de los suscriptores (Eludiora et al, 2011).

- ***Data Loss***

La pérdida de datos puede ocurrir inesperadamente a través de la corrupción de datos, borrado (por error humano) y el robo o fallo de hardware (fig. 77). Además, cuando el propietario olvida sus datos de cifrados o extravía la llave que permite la desencriptación (Cloudoye, 2015). Irrefutablemente, la idea de perder completamente los datos de forma permanente e irreversible es aterradora tanto para los usuarios como para las empresas responsables del negocio. Sin embargo, existen numerosas técnicas para frustrar este tipo de pérdida de datos.

- ***Recovery***

Con un servicio en la nube, se debe preguntar a los proveedores si se han implementado y comprobado los procedimientos de recuperación de desastres (*Disaster Recovery Procedures* DRP), los cuales le proporcionaran "la capacidad de hacer una restauración completa" y sin dejar a un lado lo más importante, que es el tiempo de respuesta o cuánto tiempo tomará la

ejecución del DRP (So, 2009). Esto puede lograrse mediante la replicación de datos a través de diferentes lugares y el plan debe ser abordado de acuerdo al nivel de servicio. A pesar de estos detalles, los *cloud services* pueden ser más fiables que el propio *datacenter* del cliente.

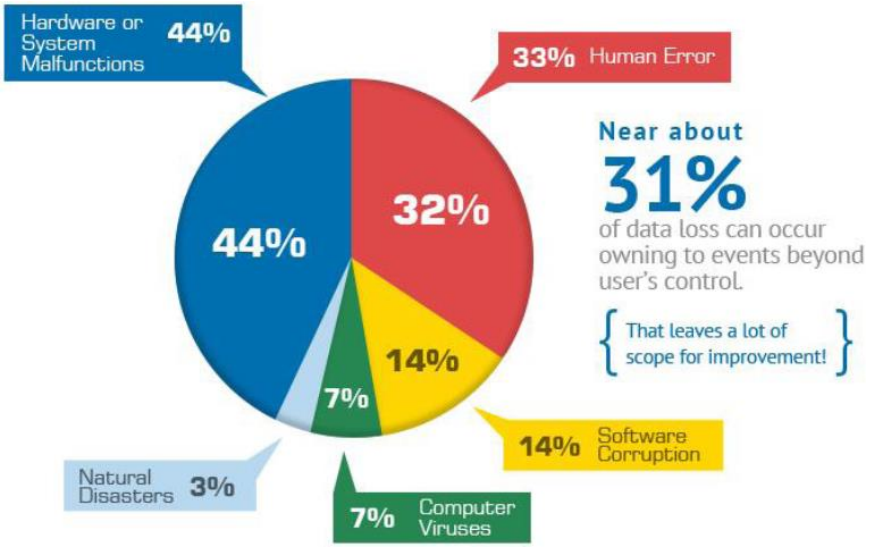


Figura 77. Las principales causas de pérdida de datos (Fuente: Cloudoye.com)

- ***Auditing***

Es el proceso de recopilación de información sobre las solicitudes de acceso del usuario a un recurso en particular o los registros al realizar alguna acción sobre el servicio en la nube. Los *logs* de registro deben ser capaces de grabar todas las acciones realizadas sobre ese recurso específico del sistema. En caso de que haya algún problema, el archivo de registro puede ser comprobado para rastrear a un usuario, permitiendo averiguar si se ha modificado los datos.

- ***Insecure Interface and APIs***

La era *Cloud* ha traído la paradoja de tratar de hacer que los servicios sean accesibles a millones, mientras que regularizan cualquier daño que estos usuarios puedan hacerle al servicio. Básicamente, las claves del API son utilizados por los servicios web y *cloud* para

reconocer una aplicación de terceros que utilice los servicios. Si por ejemplo el proveedor de los servicios *cloud* no está atento de todos los riesgos de seguridad que pueden existir, lo más probable es que el atacante pueda irrumpir en su sistema y causar daño a los datos críticos de la empresa (Cloudoye, 2015).

- ***Integrity***

Es la capacidad de proteger los datos de que no sean alterados o destruidos por personas o procesos no autorizados durante el curso de la transmisión. Esto es importante en el entorno de *cloud computing,* ya que los dispositivos móviles utilizan como medio del aire y por esta razón, los datos deben estar bien protegidos.

- ***Account or Service Traffic Hijacking***

Este tipo de riesgo de seguridad ha sido una de las principales preocupaciones relacionadas con la vulnerabilidad en la nube. El *Phishing*, la corrupción de la integridad del software en las aplicaciones, los brotes de ataques de DoS y DDoS que desbordan los *buffers*, la pérdida de contraseñas y credenciales, pueden provocar la pérdida o el control sobre la cuenta de usuario clave (Cloudoye, 2015). Un intruso con acceso a través de una cuenta de usuario puede husmear en las transacciones, manipular datos o redirigir a los clientes al sitio web de un competidor o más grave al ser redirigido hacia algún sitio donde se cometa algún tipo de fraude (fig. 78).

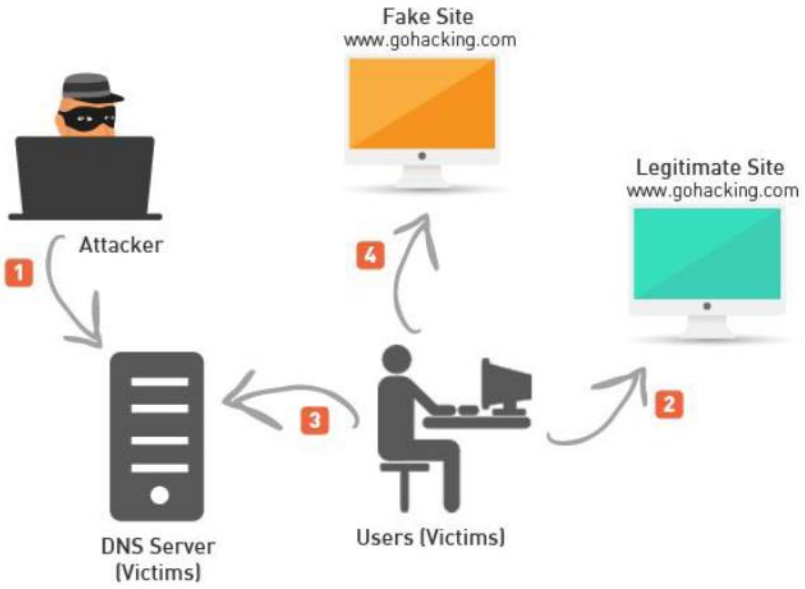


Figura 78. DNS Hijacking *(Fuente: Cloudoye)*

La mejor manera de mitigar los riesgos de seguridad para los servicios en la nube es desarrollar una infraestructura altamente segura que cubra todos los sistemas informáticos, sean estos físicos y virtuales donde sea que estos se localicen y sin importar dónde almacenen los datos.

CAPÍTULO 3: ANÁLISIS DE LA INTEGRACIÓN

CDN & Cloud Computing

Diseño Nodo CCDN

(Cloud-Oriented Content Delivery Network)

CLOUD-ORIENTED CONTENT DELIVERY NETWORK

3 ANÁLISIS DE LAS ALTERNATIVAS DE INTEGRACIÓN

Como se revisó en el primero y segundo capítulo, durante la última década, los usuarios finales han incrementado agresivamente el consumo de Internet para acceder a sitios web de todo tipo, no sólo para obtener información con contenido típico a través de una simple navegación web sin mayor carga de contenido, sino que actualmente dichos usuarios ingresan a sitios con contenido web altamente enriquecido y donde gran parte de ese incremento se da por parte del contenido de vídeo de alta definición (HD), multimedia, *streaming*, etc. Toda esta alta demanda de uso de contenido y las exigencias propias del video, han hecho que los usuarios finales exijan una mayor calidad de servicio, un mejor rendimiento red a través de grandes y consistentes anchos de banda, hasta obtener una baja latencia para todo el contenido que es entregado (Xin et al, 2011). La solución más viable y con visión de futuro, que se propuso por parte de los principales proveedores de servicios referentes del mercado, fue la de satisfacer la calidad de la experiencia (QoE) del usuario, a través de la implementación de un eficiente mecanismo de distribución de contenidos como lo es el *Content Delivery Network* (CDN) (Pathan et al, 2008), el cual ofrece mayor crecimiento por ser escalable y confiable al entregar una eficiente mejoría en el tiempo de carga de las páginas web, una consistente fiabilidad del contenido sobre los dispositivos móviles, mejor rendimiento en la descargas y *broadcast* de contenido.

CDN fortaleció el concepto de ser un sistema colaborativo de elementos de red que soporten el manejo distribuido del tráfico sobre la Internet y donde todos los contenidos que maneje, sean servidos y entregados a través de una infraestructura de *mirrored web servers* (*edge* y *replica servers*) con arquitectura escalable entre los diferentes *Points of Presence* (PoPs) del sistema y que están desplegados en el borde de la red de los proveedores de servicios de Internet (ISPs) a las que están conectados los usuarios finales. De esta manera, el contenido originado desde la fuente (*origin server*), es servido una sola vez hacia la CDN,

luego de lo cual, dicho contenido será entregado a los usuarios finales desde los *edge servers*, en lugar de que cada solicitud sea servida individualmente desde el *origin server* del proveedor de contenidos (CP) (Peng, 2008). Específicamente, la estructura de CDN proveerá el soporte necesario para un procesamiento controlado de los datos, que es una función bastante genérica del concepto de distribución de contenidos, pero a su vez, durante los últimos años este servicio se ha vuelto bastante crítico, exigente y necesario para el ámbito del negocio de casi todas las empresas que quieren innovar en la distribución de su información y que será el punto focal de esta Tesis.

Lo cierto es que, si bien inicialmente las diferencias entre CDN (básicamente, un conjunto de servidores distribuidos) y *Cloud Computing* (básicamente, un conjunto de servidores centralizados) era obvia, las ineficiencias de la prestación de servicios sobre la Internet (falta de QoS y QoE) han obligado a una paulatina integración de ambas tecnologías (Molina, 2013). Este capítulo se centrará principalmente en la integración de CDN y *Cloud Computing,* mediante la exposición de una serie de ventajas y desventajas que se dan durante la puesta en marcha de las diferentes funciones de procesamiento multimedia dentro del entorno *Cloud (NCE),* para complementariamente entender los despliegues y alcances de *Cloud/CDN*. También se mostrará cómo la popularidad de los diferentes modelos de servicios *Cloud* y su despliegue a diferentes escalas en varios medios tecnológicos, ha desdibujado aún más la distinción de conceptos entre las diferentes infraestructuras de *service delivery*.

3.1 EVOLUCIÓN TECNOLÓGICA DE LAS CDNs

Las CDNs han ido evolucionando desde diversos puntos de vista y pueden ser capaces de adaptarse teniendo en cuenta el ámbito del tipo y gama de contenidos ofrecidos a cualquier usuario a través de un terminal con conectividad a la red de datos (Internet), incluyendo contenido estático de (1ª generación) como objetos diseñados para el almacenamiento Web, ej.: HTML, documentos, texto, gráficos, *uniform resource locators* (URL) y scripts, objetos

descargables o contenido de (2ª generación) como servicios de *caching* dinámico multimedia como el directorio de base de datos (DHTML, distribución de *software*, *gaming*, animaciones, películas, imágenes médicas, scripts), aplicaciones (ej: *e-commerce* y portales) y contenido con medios de *streaming* que puede ser video bajo demanda (*on-demand*) o medios de comunicación en vivo (*live-streaming*) (Leivadeas, 2012). Por otro lado, las redes sociales y muchos otros grupos dentro del *social media marketing*, han implementado el soporte de los *Content Service Providers* (CSPs) para la disminución de costos, riesgos en la transacción de datos y carga agresiva sobre la red. Aunque CDN entrega contenido a los usuarios finales con alta disponibilidad, la mayoría de proveedores de CDN no cumplen con ciertos requerimientos de QoS que se han propuesto recientemente (debido a sus débiles políticas de SLAs durante la implementación de un servicio extremo a extremo), de igual manera podrían presentar inconvenientes de servicio (debido al aumentando exponencial en la demanda de funciones multimedia del lado del *delivery/server*), como parte del enorme incremento en el manejo de contenido (*BigData)* (Ericsson, 2012).

Un aspecto evolutivo de una CDN se da desde la perspectiva de su infraestructura, donde existe la prestación y aprovisionamiento elástico de servicios críticos que se están llevando a cabo cada vez más a través de *Cloud Computing* (aquí la disponibilidad debe ser prácticamente del 100%), ya que en la actualidad un operador de tipo *Cloud* puede emplear perfectamente una CDN como infraestructura, para ofrecer cualquiera de los servicios de valor agregado mencionados anteriormente. Por esta razón las arquitecturas CDN a menudo aprovechan la infraestructura y servicios de *Cloud* dada su necesidad de almacenamiento y capacidad de cómputo para proporcionar escalabilidad, flexibilidad, disponibilidad y rendimiento. Es por esto que *cloud computing* y toda su colección de servicios de aplicaciones, hardware y sistema de software, se ha convertido en la tecnología preferida para prestar servicios a los usuarios finales a través de la Internet (Armbrust et al 2009).

También se debe considerar que es el *Datacenter* y su infraestructura el medio a través del cual se despliega el *hardware* y *software* necesarios, constituyendo la esencia de los servicios de *Cloud*. Por múltiples razones, la sinergia entre *cloud computing* y CDN resulta especialmente interesante en el procesamiento multimedia, ya que incluye un amplio espectro de funciones para aplicaciones de *video streaming*. El *cloud computing* permite disponer de

un espacio de almacenamiento prácticamente ilimitado para almacenar una gran cantidad de contenido de vídeo, así como una elevada capacidad de procesamiento para la transcodificación de dicho contenido, la entrega de medios interactivos, la mezcla de diferentes flujos o *streams* de información, la modificación dinámica de la resolución del video, la personalización de servicios, así como el incremento y decremento del número de clientes sobre el sistema; en base a las necesidades que tenga el equipamiento del usuario.

Aprovechando todas las ventajas de los servicios de *cloud computing*, una tendencia reciente ha surgido en los proveedores de CDN y es la de desplegar elementos de la infraestructura *Cloud* (ej. la arquitectura Netflix, al igual que algunos *cloud service providers* (CP) ofrecen funciones de CDN (Adhikari et al, 2012). Sin embargo, todos estos desarrollos recientes han fracasado a la hora de aprovechar todos los beneficios del *Cloud Computing*, es decir que, los servicios derivados de la CDN y que se despliegan en lo más alto de la infraestructura de *cloud computing* siguen siendo limitados, es por esto que una evolución del *cloud computing* puede combinar los beneficios y aprovechar todas las técnicas. Por ejemplo, hace un par de años, Netflix tuvo que transcodificar más de 17.000 películas y shows de televisión para que su contenido pueda ser soportado por la consola PlayStation 3 (Netflix, 2014), enviando más de 80 TB de datos a Amazon para que se apliquen los cambios necesarios dentro del servicio de *Cloud Computing*; teniendo en cuenta que el material estaba en 19 formatos diferentes y Netflix necesitaba 6 tasas de video distintas además de una tasa de audio, Amazon empleó 1200 instancias de su producto EC2 para realizar el trabajo en unos pocos días. Una vez que el contenido fue tratado dentro de la infraestructura de *cloud computing*, fue entonces cuando el almacenamiento y la capacidad de procesamiento estuvieron disponibles para el vídeo, luego de lo cual el siguiente paso dependió del mecanismo de distribución, sobre el cual Netflix tuvo que apoyarse y donde típicamente emplea una CDN que le ha permitido disminuir las latencias, así como los flujos necesarios, ya que los *proxy cachés* son capaces de agregar múltiples flujos desde el *datacenter*, minimizando de esta forma el consumo del ancho de banda.

De acuerdo con la herramienta *Visual Networking Index* de Cisco, en el 2013 las distintas formas de vídeo (CISCO, 2013), como por ejemplo: la transmisión de un TV inteligente, el *video on demand* (VoD), el video por Internet (*streaming*) y el tráfico *peer-to-*

peer (P2P), han empezado a superar el 90% del tráfico mundial consumido. Por lo tanto, el enfoque de CDN no debe limitarse solamente a entregar el contenido, sino más bien debe extenderse y adaptarse sobre todas sus formas de procesamiento, entregando productos más allá de las meras tecnologías de *caching,* a través de la interoperabilidad, balanceo de carga y la globalización de servicios; comprendiendo que los negocios requieren de infraestructuras y modelos de comercio más versátiles, heterogéneos, con capacidades de satisfacer los requisitos actuales y futuros de los consumidores (Molina, 2013).

3.2 BENEFICIOS DE LA IMPLEMENTACIÓN DE SERVICIOS CCDN

Las CDNs tradicionales siempre redirigen clientes a los servidores de contenido que están geográficamente más cerca, pero carecen de técnicas estándar de balanceo de carga y sufren de sobrecarga en el *edge server* cuando la popularidad del contenido local aumenta. Es por esto que el uso de las *Cloud-Oriented Content Delivery Networks* (CCDNs) constituirá una alternativa prometedora ante las redes de distribución de contenido tradicionales, ya que aprovecha las ventajas y los principios del servicio en la nube, tales como el modelo de negocio *pay-as-you-go* y la dispersión geográfica de los recursos, CCDN puede proporcionar una solución viable y rentable al proporcionar redes y servicios de distribución de contenidos. En este trabajo se propone y evalúa un marco jerárquico hacia una solución eficiente y escalable de la distribución de contenidos a través de un entorno *Cloud* (NCE) sobre una red multiproveedor, donde los recursos de comunicación *inter* e *intra Cloud* son considerados de forma simultánea junto con los recursos del *Cloud Computing* tradicionales (Papagianni et al, 2013). CCDN es una de las soluciones para resolver los problemas de latencia y congestión de la red, ya que integra el *content delivery network* (CDN) con los beneficios del *cloud computing* y la infraestructura del *datacenter*. En el contexto de relacionar dichas tecnologías mediante una analogía, el *datacenter* sería la planta de fabricación y los servidores CDN serían los almacenes de los distribuidores. Del mismo modo, la plataforma *Cloud* puede utilizar estrategias de CDN como el *global load balancing* para reducir el *cloud*

export congestion y el retardo de acceso para los usuarios, haciendo del *CDN/Cloud* un esquema viable y significativo para cualquier red.

Cloud-Oriented CDN (CCDN) utiliza los servicios en la nube para incrementar los requerimientos de capacidad de las CDNs, con características como la replicación de contenido a través de ubicaciones geográficamente dispersas que deberían ser transportadas sobre la nube. Las CCDNs pueden reducir el esfuerzo y el costo de producción de soluciones completas y dedicadas, en un entorno de desarrollo con tiempos de recuperación de la inversión más cortos durante su ciclo de mercado. Al ser CCDN un emergente y desafiante paradigma de computación, es necesario aprender a afrontar eficazmente su problemática de implementación, hasta que se introduzca o adopte un modelo apropiado de costos, comprendiendo los desafíos que se dan durante la representación gráfica de su particionamiento y la ubicación de los *replica servers* (*surrogates*) dentro de la nube, teniendo siempre en consideración la limitación de recursos físicos de CDN. La mezcla o adición de recursos de comunicación *intra* o *inter Cloud* conduce a todo el sistema a un *Networked Cloud Computing Environment* (NCE) para la distribución de servicios a través de modelos, como el de *infrastructure as a service* (IaaS) (Xin et al, 2011). CCDN proporciona características de distribución de contenido inigualables, junto con las características claves del campo de *Cloud Computing* como lo son el *Cloud Storage* y el *Platform as a Service* (PaaS), a más de esto también optimiza el algoritmo de balanceo de carga de la redirección por DNS (*Cloud Load Balancing*).

Ya en términos de una solución de entrega de contenido CDN a través de un *Networked Cloud Environment* (NCE), el *surrogate server* se convierte simplemente en otro recurso dentro de la nube, mientras que los *virtual servers* (VS) se conectan con las rutas de comunicación *intra* o *inter Cloud* utilizados normalmente para la distribución de contenidos. Dentro de un esquema NCE de multiproveedor, donde coexisten varios *cloud service providers* (CPs), algunos proveedores con una red de tránsito pueden ser utilizados para la interconexión de los *cloud sites*, ya que al interior del ecosistema, los *virtual surrogate servers* (VSS) se implementan sobre los *cloud sites*. Sin embargo, también se deben implementar eficientes rutas de distribución de contenido, aprovechando que la topología de la red física subyacente permite un mayor nivel de libertad en su diseño y lo que es más, estas

rutas pueden abarcan múltiples dominios administrativos. Por esta razón, ciertos diseños requerirán de un apropiado servicio de *cloud brokering*, de tal modo que las *multicloud-based* CDNs tengan una eficiencia operativa y de costo (Chen, 2012). Por otra parte, en términos de distribución de contenido, la redirección de la replicación se convierte en algo de suma importancia con respecto al coste de CCDN, ya que coexistente con el modelo de costos de *Cloud*, donde la carga y la descarga de información pueden variar el precio final del servicio.

Los principales objetivos de CCDN son:

- Proporcionar una alta escalabilidad y estabilidad del servicio a través de la infraestructura en la nube.
- Reducir el costo de comunicación entre el *edge server* y el servicio de infraestructura en la nube.
- Permitir que el *content service provider* (CSP) gestione los contenidos con facilidad.
- Ajustar el sistema de tal manera que los contenidos más populares siempre estén ubicados lo más cerca posible de los usuarios finales.

3.3 RETOS DE LA IMPLEMENTACIÓN DE CCDN

Existen algunas limitantes dentro de las CDNs comerciales, tales como los altos costos de implementación, el mantenimiento y la capacidad de servicio de un *surrogate*. Además, los proveedores de contenido deben prestar más atención a los niveles de latencia en la red, en especial cuando un *surrogate* ingresa contenidos desde el almacenamiento en la nube y cuando los usuarios no pueden obtener dicho contenido del *surrogate* (Chia-Feng et al, 2011). En tal sentido, hay varias cuestiones que requieren ser revisadas para la integración de CDN y *cloud computing:*

3.3.1 *Graph Partitioning*

Principalmente se refiere a los aspectos relacionados con la naturaleza de un esquema compartido de multiproveedor, debido a la necesidad de alcanzar un modelo escalable y eficiente; donde como punto de inicio, el área geográfica de despliegue de la infraestructura CCDN se subdividirá en áreas más pequeñas, llamadas *service clusters* y que como resultado formarán un *intercluster content distribution graph*. Asumiendo la coexistencia de varios CPs sobre la misma área de servicio, se lleva a cabo un particionamiento rentable del *intercluster content distribution graph*, por lo tanto, a cada *service cluster* se le será asignado a un CP que esté basado en la técnica *iterated local search* (ILS) (Papagianni et al, 2013) y que será de utilidad para la elaboración del mapeo de recursos sobre las redes tipo *Cloud*.

3.3.2 Latencia de Red

Dentro de la infraestructura CCDN, la funcionalidad de almacenamiento que se realiza sobre la nube (c*loud storage*) puede ser considerada como la funcionalidad que realiza el *origin server* dentro la arquitectura CDN, con la diferencia que el *cloud storage* tiene una elevada capacidad para ofrecer disponibilidad de contenidos y es altamente tolerante a fallos; por esta razón, su rendimiento general ante el problema de latencia general es mucho mejor que el de un simple *origin server* dentro de CDN (Chia-Feng et al, 2011). Con el fin de hacer un buen uso de esta ventaja y mejorar aún más el inconveniente de la latencia, la colocación e intercambio de contenido web se lo realiza lo más cerca de los usuarios finales. Teniendo en cuenta esto, cada uno de los *cloud providers* seleccionados dentro de CCDN empleará un apropiado *surrogate placement algorithm* para determinar el número y la ubicación de *surrogates* que serán alojados dentro del subconjunto de los *cloud sites* que estarán disponibles en el dominio, considerando esta estrategia como la mejor para reducir la latencia dentro de la red y minimizar el tráfico al momento que se efectúe la petición del cliente.

3.3.3 *Replica Placement*

A través de un correcto manejo de este mecanismo, la replicación de los contenidos se la puede realizar a los *datacenter* o *clusters* regionales, donde el servicio de almacenamiento en la nube podría tener un mecanismo de copia de seguridad remoto y acceso de contenido directo para el servicio de CDN, colocando el contenido lo más cerca posible del usuario (Chia-Feng et al, 2011). Este inconveniente finalmente influenciará la selección efectiva de los *surrogate servers* y su mapeo sobre el sustrato de infraestructura correspondiente dentro de un único dominio. Para este mecanismo se introducen dos soluciones:

- El primer enfoque viene de parte del *virtual surrogate placement* (VSP) y se ejecuta en dos fases: 1) la fase de selección del *surrogate server,* y 2) la fase de selección de la ruta de distribución de contenido dentro del dominio (Leivadeas et al, 2012).

- El segundo enfoque es a través del empleo de un mecanismo que está inspirado en *social network analysis* (SNA) y *greedy virtual surrogate placement* (SNA-GVSP); donde el objetivo final propuesto será el de asignar con eficiencia usuarios finales a los *virtual surrogate servers*, maximizando así el *shortest path betweeness centrality* (SPBC) (Papagianni et al, 2012) para el conjunto total de *surrogates* seleccionados, de tal manera que a cada usuario final se le asigne por lo menos un *cloud site* que satisfaga los requisitos de QoS para las solicitudes de los usuarios.

3.3.4 Balanceo de Carga en CDN

Hoy en día los propietarios del contenido Web se enfrentan a retos bastante significativos durante la entrega de contenido a sus usuarios, debido a que el número de usuarios de un servicio CDN se ha disparado debido a la interacción de conocimientos en línea entre las

culturas a través de las diferentes geografías. Por esta razón, ahora ya se tiene una alta variabilidad en el tipo de contenido compartido, donde un típico sitio web puede estar compuesto de una alta cantidad de texto enriquecido con tecnología web, imágenes, vídeos y animaciones; es por esta razón que existe una tremenda necesidad de entregar servicios personalizados los cuales den como resultado un aumento del contenido dinámico, resaltando que esta es una época donde ciertos usuarios generan gran parte de sus ingresos económicos a través del mercado en línea. Debido al continuo crecimiento del número de usuarios y con el fin de lograr un servicio escalable y confiable, una CDN debe considerar dos propiedades: *load awareness* y *locality awareness* (Chia-Feng et al, 2011) (fig. 79).

- ***Load Awareness.***

Esta propiedad atiende solicitudes a través de los *edge servers* globales empleando el contenido replicado, balanceo computacional y congestión de red. El mecanismo de *IP anycasting* no fue considerado inicialmente como un enfoque viable para la solicitud de redireccionamiento de las CDNs (Chia-Feng et al, 2011), esto se debió a los efectos secundarios no deseados durante los cambios de enrutamiento en dicho mecanismo sobre la Internet, toda esa falta de propiedades que vinieron con *load awareness* y que fue resulta con la arquitectura CDN basada en DNS.

- ***Locality Awareness***

Esta propiedad utiliza relaciones de proximidad dentro de la red: como la distancia geográfica, la latencia de ida y vuelta o la distancia topológica; cuando se redirecciona las peticiones de cliente a los *edge servers* (Hernansanz et al, 2009). Es importante comprender que se podrían emplear algoritmos simples como el *Round Robin* o algoritmos que funcionen con asignación por pesos, hasta algoritmos más complejos.

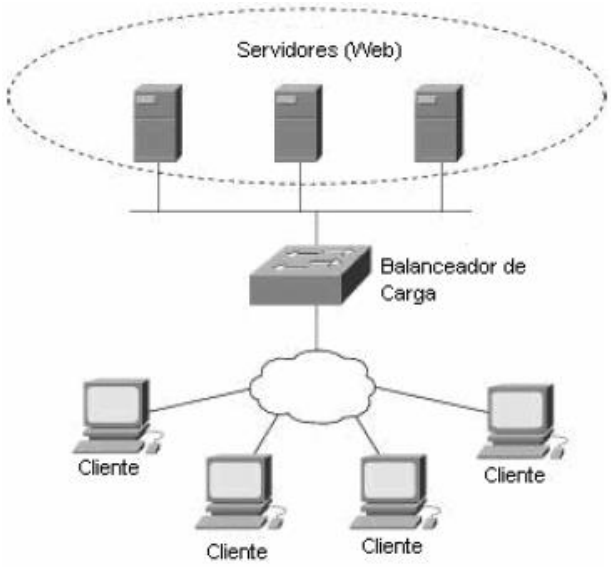


Figura 79. Balanceador de carga como front-end. (2009) (Fuente: Hernansanz J.)

Se podría describir algunos de los balanceadores de carga más populares:

3.3.4.1 POUND

Este programa es un *reverse proxy* (ya que pasa las peticiones que realiza el navegador del cliente a uno o más *back-end servers*); también es un balanceador de carga (ya que distribuye las solicitudes que realizan los navegadores del cliente entre varios *back-end servers*, mientras mantiene la información de una sesión) y también es considerado como un servidor *Web front-end HTTPS* (ya que las solicitudes HTTPS son desencriptadas y se pasan a los *back-ends servers* como HTTP de texto plano). *Pound* es distribuido bajo licencia GPL y se lo utiliza de manera gratuita; inicialmente fue desarrollado para permitir la distribución de carga entre varios servidores Web y permitir usar el protocolo de seguridad SSL sobre aquellos servidores Web que no lo ofrecían de forma nativa (Hernansanz et al, 2009). *Pound* necesita de los objetos *listeners, services* y *back-ends* para funcionar y dado que su uso es bastante concreto, ya que su utilidad prácticamente está limitada a la de un balanceador de

carga, su instalación y configuración es realmente sencilla, lo que permite tenerlo operativo en instantes.

3.3.4.2 HAProxy

Es una solución gratuita que ofrece rápidamente alta disponibilidad, balanceo de carga y servicios de *proxying* para aplicaciones TCP y HTTP. Es particularmente útil en *web sites* con una alta demanda de tráfico y un gran número de conexiones simultaneas, que con frecuencia se despliegan por defecto en plataformas *Cloud* (Hernansanz et al, 2009). Con los años se ha convertido en el balanceador de carga de código abierto estándar, ya que ahora se incluye con la mayoría de las distribuciones Linux y la forma más sencilla de instalarlo es directamente desde sus repositorios (fig. 80). Su modo de funcionamiento hace que su integración con arquitecturas existentes sea más fácil de configurar e instalar y no tenga tantos riesgos al igual que el *Nginx* o *Pound*, sin dejar de ofrecer la posibilidad de que queden expuestos a la red los servidores web.

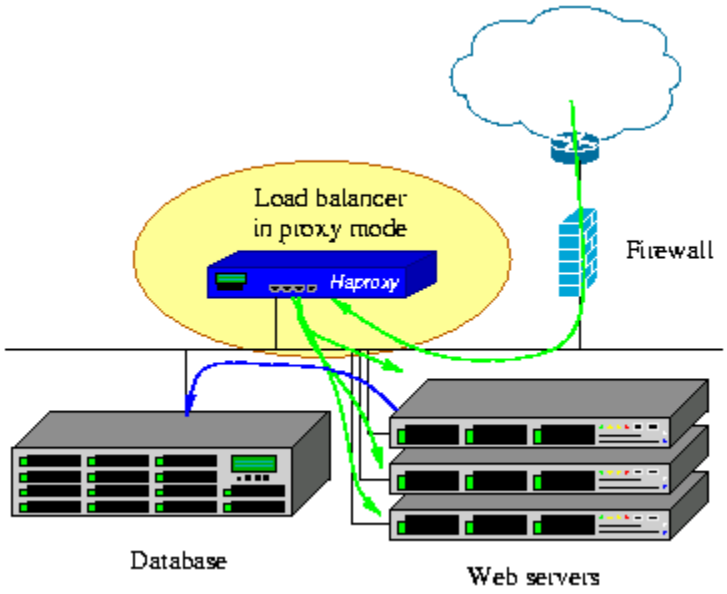


Figura 80. HAProxy como front-end y en modo proxy. (2015) (Fuente: haproxy.org)

3.3.4.3 NGINX (Engine x)

NGINX es conocido por su estabilidad, gran conjunto de características, configuración simple y bajo consumo de recursos, debido a que realiza el balanceo de carga a través de múltiples instancias de aplicación con ciertas modificaciones propietarias que las trae por defecto, lo cual maximiza su rendimiento, reduce la latencia y garantiza una configuración tolerante a fallos. Tiene la funcionalidad de ser un servidor con características HTTP (manejo de ficheros estáticos, índices y autoíndices), *reverse proxy* acelerado sin caché, servidor *proxy* IMAP/POP3, FastCGI, arquitectura modular, soporte SSL, etc. Fue desarrollado por *Igor Sysoev* para el sitio web ruso *Rambler.ru* y su código fuente se liberó después bajo una licencia de tipo BSD (Hernansanz et al, 2009).

Soporta algunos mecanismos de balanceo de carga, como: *round-robin* (donde las solicitudes a los servidores de aplicaciones se distribuyen en un esquema de selección equitativa y en un orden racional), *least-connected* (hace que la siguiente solicitud sea asignada al servidor con el menor número de conexiones activas) e *IP-hash* (que utiliza una función de *hash* para determinar qué servidor debe ser seleccionado para la siguiente solicitud, esto lo hace según la dirección IP del cliente). *Ngnix* tiene más opciones de configuración que *HAProxy* y *Pound*, sin embargo, cualquiera de estos puede ser una opción viable de balanceador de carga ante *Apache*, ya que su configuración es algo más extensa.

En resumen, un balanceador de carga fundamentalmente se presenta como un dispositivo de hardware o software que se pone al frente de un conjunto de servidores (*front-end*) y que atiende una aplicación; su principal función es la de distribuir la asignación o balancear las solicitudes que llegan desde los clientes hacia los servidores a través del uso de alguno de los algoritmos antes indicados. Sin embargo aunque las CDNs tradicionales estén optimizadas para ofrecer un gran rendimiento, escalabilidad y confiabilidad, no tienen la flexibilidad necesaria para permitir que los propietarios del contenido controlen el tráfico en tiempo real, por lo que en el modelo NCE se expone la implementación de un *Cloud Load Balancer* integrado con mecanismos *DNS-based Global Load Balancing* (CDNetworks, 2015).

3.4 CLOUD LOAD BALANCER

Dentro del concepto CCDN, el *cloud load balancer* trabaja en la capa de DNS y propone un mecanismo *cloud-based load balancing* diseñado principalmente con el propósito de resolver los inconvenientes específicos de las CDNs tradicionales. Es una solución de software potente y flexible que le permite al usuario gestionar su propio tráfico, dándole el control sobre la entrega del contenido a través de un mecanismo basado en políticas, con especificación de reglas técnicas y de negocio. Las políticas se aplican en tiempo real e incluyen su distribución entre las CDNs del sistema, balanceando la carga entre múltiples servidores, con un mecanismo de enlistado de usuarios basados en la geo-localización (*whitelist/blacklist*) o de manera dinámica como el porcentaje del tráfico del día y el enrutamiento del tráfico en función de las condiciones favorables de la infraestructura (CDNetworks, 2015). Escala automáticamente su capacidad de administración de solicitudes, para satisfacer las demandas de tráfico de las aplicaciones a fin de garantizar que dispone de capacidad *back-end* para satisfacer distintos niveles de tráfico sin recurrir a la intervención manual; permitiendo a los usuarios reaccionar rápidamente a nuevos mercados, personalizar la entrega de contenido basado en las condiciones de uso, controlando el precio y el rendimiento, mientras se incrementa la eficiencia operativa de un negocio.

Cloud Load Balancer distribuye automáticamente todo el tráfico entrante de las aplicaciones entre varias instancias y zonas de disponibilidad virtuales en la nube, permitiendo conseguir niveles más altos de tolerancia a errores en las aplicaciones, ya que ofrece de manera integral la capacidad de equilibrio de carga necesaria para distribuir el tráfico de las aplicaciones, basándose en la infraestructura *Cloud DNS* que es segura, confiable y escalable; ofreciendo un alto rendimiento bajo cualquier condición de tráfico y desde cualquier parte del mundo. Este método ejecuta aplicaciones en varias regiones y designa balanceadores de carga alternativos para conmutación por error en las distintas regiones, proporcionando sofisticadas y flexibles capacidades de gestión de tráfico a nivel DNS, está especialmente diseñado para atender las necesidades de rendimiento web de las empresas y sin mayor inversión de capital para la implementación o mantenimiento de sistemas complejos de hardware-software; en caso de que su aplicación no responda, este

método eliminará el punto de enlace del balanceador de carga no disponible del servicio y dirigirá el tráfico a un balanceador de carga alternativo en otra región. Cada balanceado de carga recibe un nombre por defecto del *Domain Name System* (DNS), por lo que este nombre de DNS incluye el nombre de la región en el que se crea el balanceador de carga. Entre algunas de las características claves, están:

3.4.1 Tráfico On-Demand y Failover Automático

La infraestructura *Cloud Load Balancer* proporciona un alto grado de disponibilidad, escalabilidad y un rendimiento superior para los usuarios, independientemente de donde sea su ubicación, a través de la propagación y la aplicación de políticas casi en tiempo real, lo que le faculta a tener una reacción inmediata a las necesidades del mercado. Ante la falla de un servidor, la respuesta DNS para ese servicio se sustituye por la atención de un servidor que está en modo *standby* hasta que el principal sea restaurado, permitiendo una mayor disponibilidad y redundancia al reducir las interrupciones costosas del servicio y mejorando la eficiencia operativa (CDNetworks, 2015).

3.4.2 Tráfico entre Servidores y Control de Tráfico (Geo-Based)

Permite un mejor rendimiento y control de costos, optimizando la utilización de las inversiones actuales, utilizando un único CDN, múltiples CDNs o la combinación de CDNs y servidores virtuales alojados en *datacenters*. Ciertas condiciones y acciones se pueden combinar de cualquier manera para generar un conjunto de políticas y acciones muy poderosas para la gestión del tráfico.

3.4.3 Whitelist / Blacklist

Permite denegar los servicios a determinados usuarios en función del negocio o para prevenir ataques. Mediante el uso de ciertas políticas, no se enviará ninguna respuesta a las solicitudes de servicio desde determinadas direcciones IP o regiones geográficas.

3.4.4 Control Avanzado e Informes

Puede implementar una gestión basada en Web para la administración fácil del DNS. Con su sistema de gestión casi en tiempo real, se pueden generar informes avanzados, que proporcionan una visión sin precedentes sobre el tráfico de DNS y el de los clientes.

3.5 ALTERNATIVAS DEL PLANTEAMIENTO GENERAL Y ESTRUCTURA

Como se habló durante el capítulo de CDN, en su esquema tradicional se involucra a los siguientes actores básicos: los usuarios finales, el proveedor de contenidos (CSP) y el proveedor de CDN. Los usuarios finales son los clientes que tienen acceso al contenido del sitio web de los proveedores de contenidos, estos proveedores a su vez son entidades que poseen o tienen licencia para vender contenidos que son almacenados inicialmente dentro de su *origin server* como activos del negocio. Por otro lado, el proveedor de CDN es el que despliega, mantiene y proporciona las instalaciones de infraestructura necesarias para una oportuna y confiable distribución del contenido, esencialmente consta de un conjunto de *edge servers* (*surrogates* y *replicas*) que entregan copias del contenido a los usuarios finales (Pathan, 2008). Pero ya dentro del contexto de un NCE, los proveedores de contenido pueden implementar sus propias *cloud-based CDNs,* arrendando servicios en la nube de tipo IaaS, de

alguno de los *Cloud Service Providers* (CPs) disponibles en el mercado (ej. Amazon y su EC2), o también, 2) arrendando servicios *Cloud* de alguno de los *Transit Network Providers* del mercado, que proporcionan servicios *intercloud virtual network* (ej. *On-demand QoS-guaranteed paths*) (Xin et al, 2011). Los *transit network providers* pueden utilizar tecnologías de red virtual de capa L2/L3 (ej. VLANs) para establecer la infraestructura de red y conectividad entre los múltiples CPs. Por lo tanto, un proveedor de contenido (CSP) actuará también como un proveedor de CCDN, donde el contrato de arrendamiento se puede negociar de forma directa con los CPs o los proveedores de la red de tránsito, con ayuda de algún intermediario (*CCDN Brokers*). Para esto es necesario considerar y tener en cuenta los niveles de desempeño y estadísticas favorables del proveedor, dentro del *cuadrante mágico de Gartner*, como los que se exponen a continuación en la figura 81:

Figura 81. AWS nombrado líder en el IaaS Magic Quadrant por cuarto año consecutivo (2015) (Fuente: aws.amazon.com)

3.5.1 Amazon CloudFront

Es un servicio web para la entrega de contenido, este servicio se integra con otros servicios web de Amazon (AWS) para proveer a desarrolladores y empresas de herramientas que faciliten la distribución del contenido a los usuarios, brindando baja latencia, alta velocidad de transferencia y sin mayor problema ante acuerdos de servicio (SLAs) (Amazon CloudFront, 2016). *CloudFront* puede entregar contenido estático y dinámico (*streaming),* como: .html, .css, .php, imagen y *media files;* utilizando una red global de *edge locations*, donde las peticiones de contenido que se solicitan, son enrutadas automáticamente a los *edge location* más cercano y pueden utilizar Amazon S3 o EC2 para convertirse en el *origin server* sobre el cual se pueda almacenar los archivos y entregarlos con el mejor rendimiento posible (Chia-Feng et al, 2011). Por lo tanto, el *origin server* y el *edge server* serán parte del servicio de Amazon, po0r lo que algunos protocolos y métodos de acceso previstos en Amazon S3 o servidores HTTP se pueden aplicar en *Amazon CloudFront*. Las solicitudes de contenido dinámico se devuelven a los *origin server* que se ejecutan en *Amazon Web Services* (ej. *Amazon EC2*, *Elastic Load Balancing*) a través de rutas de red optimizadas para conseguir una experiencia más fiable y coherente.

Una única llamada al *Application Programming Interface* (API) o interactuando con la consola de administración de AWS, permite empezar a distribuir contenido desde el *bucket* de S3, las instancias de EC2 u otro *origin server* a través de la red de *Amazon CloudFront*, pagando unicamente por el contenido que se distribuye a través de la red. Amazon monitoriza constantemente estas rutas de red y las conexiones de las ubicaciones de borde de *CloudFront,* ya que en el origen se vuelven a utilizar para ofrecer contenido dinámico desde la red de entrega de contenido (CDN) con el máximo desempeño posible, de tal manera que los cambios que se haga en la configuración existente se aplicarán en toda la red global en pocos minutos y sin tener la necesidad de crear dominios independientes para el contenido estático y dinámico. Si el origen es un servicio de AWS la transferencia de datos del origen a las ubicaciones de borde (“recopilaciones de origen” de *Amazon CloudFront*) será gratuita. Esto abarca la transferencia de datos desde todas las regiones de AWS a todas las ubicaciones de borde mundiales de *CloudFront*.

Tabla 7.

EC2 - Precios bajo demanda. (Fuente: Amazon.com)

Transferencias de datos regionales a Internet (por GB)

	Estados Unidos	Europa	Hong Kong, Filipinas, Corea del Sur, Singapur y Taiwán	Japón	América del Sur
Primeros 10 TB/mes	0,085 USD	0,085 USD	0,140 USD	0,140 USD	0,250 USD
Siguientes 40 TB/mes	0,080 USD	0,080 USD	0,135 USD	0,135 USD	0,200 USD
Siguientes 100 TB/mes	0,060 USD	0,060 USD	0,120 USD	0,120 USD	0,180 USD
Siguientes 350 TB/mes	0,040 USD	0,040 USD	0,100 USD	0,100 USD	0,160 USD
Siguientes 524 TB/mes	0,030 USD	0,030 USD	0,080 USD	0,080 USD	0,140 USD
Siguientes 4 PB/mes	0,025 USD	0,025 USD	0,070 USD	0,070 USD	0,130 USD
Más de 5 PB/mes	0,020 USD	0,020 USD	0,060 USD	0,060 USD	0,125 USD

Transferencias de datos regionales al origen (por GB)

	Estados Unidos	Europa	Hong Kong, Filipinas, Corea del Sur, Singapur y Taiwán	Japón	América del Sur
Transferencia de todos los datos	0,020 USD	0,020 USD	0,060 USD	0,060 USD	0,125 USD

3.5.2 ActiveCDN

Se basa principalmente en la prestación de servicios tradicionales de CDN, pero también permite ofrecer servicios bajo demanda, ya que cuenta un sistema de nodos de almacenamiento de contenido (tipo *pop-up*) que se activan en función de las necesidades del tráfico sobre Internet, sin que estos se hayan implementado previamente (Srinivasan et al, 2011). Estos nodos con propósito *pop-up* pueden almacenar dinámicamente el contenido de vídeo, para que después sean estos nodos quienes provean el contenido, en vez de que sea el *CDN replica server* quien atienda la solicitud. El resultado es la reducción del tráfico de *streaming* en los servidores CDN.

3.5.3 Akamai NetStorage

Es un servicio adaptado para la red de almacenamiento de contenido. Mediante el uso de este servicio de almacenamiento tolerante a fallos, los usuarios pueden tener su propio contenido multimedia disponible y de forma rápida para todos los que lo soliciten (servicio *on-demand*). Este servicio y sus derivados se compone de varios terabytes de capacidad de almacenamiento, replicación geográfica, una arquitectura escalable de forma masiva, con un mapeo y tecnología de enrutamiento propietarios (Akamai NetStorage, 2016). *NetStorage* replica el contenido del cliente para asegurar su disponibilidad, colocando otras copias de los archivos en dos de los muchos centros de almacenamiento de *Akamai* en todo el mundo. Estos servicios de entrega hacen uso de una masiva y distribuida red global de servidores conocida como *Akamai Platform* y que trabajan en conjunto para ubicar de manera inteligente el contenido lo más cercano de los usuarios web. A través del uso de *Akamai Global Traffic Management* (Chia-Feng et al, 2011) se determina la ubicación más óptima para el almacenamiento, desde donde se pueda hacer un *pull* con contenido fresco, lo que permite que este servicio sea eficiente, de mejor rendimiento y con alta confiabilidad.

3.5.4 MetaCDN

Propone un enfoque alternativo mediante el aprovechamiento de la infraestructura menos costosa de almacenamiento a través de *Cloud*. Proporciona una red integrada de superposición que elimina la complejidad que se presenta al tratar con múltiples proveedores de almacenamiento (Broberg, Buyya & Tari 2009). Su principal función es la de identificar la mejor propuesta de servicio que sea posible de entre todos los proveedores de almacenamiento en la nube asociados, basándose en la preferencia de su calidad de servicio, experiencia (QoS-QoE), cobertura y presupuesto.

3.5.5 Level 3 Cloud Connect

Propone un nuevo tipo de red privada y sin interrupciones de aplicaciones de misión crítica, diseñada para ofrecer flexibilidad con los servicios en la nube sin comprometer la productividad o la seguridad, permite formar conexiones seguras dentro de *datacenters* y *clouds* mediante servicios de red de fibra óptica altamente redundante y de bajo retardo. Es una tecnología de red específica de sus *datacenters* especializados, que permite combinar los beneficios de las redes públicas y privadas, para crear un servicio de red verdaderamente revolucionario y confiable en el cual las organizaciones puedan migrar y optimizar sus aplicaciones en la nube, con un uso de ancho de banda dinámico al pagar solo por lo que se consume (Peak10, 2015). Cuenta con servicios de interconexión con algunos de los mayores representantes *cloud* del mercado, como son: *Amazon Web Services Direct Connect*, *Windows Azure ExpressRoute*, *Google Cloud Interconnect, HP Helion Network* y *Blue Jeans Network*. La interconexión contra Amazon cuenta con conexiones privadas y dedicadas a los recursos de la nube de AWS globales, donde forma múltiples interfaces virtuales sin la dependencia de la red pública de internet (*Amazon VPC*), para brindar una solución de distribución de archivos B2B segura a los cliente que necesitan publicar, distribuir y transportar archivos grandes a una audiencia internacional.

3.5.6 Windows Azure

La plataforma *Windows Azure* es un servicio de PaaS sobre el medio del *Public Cloud*, donde los clientes crean aplicaciones sobre la plataforma *Azure* (Microsoft, 2015), mediante el uso de lenguajes de programación (ej. C#) y algunas herramientas como (ej. *Visual Studio*) soportado también por *Azure*. Al ser Microsoft un proveedor *PaaS Cloud*, este será responsable del sistema operativo *cloud* (OS) conocido como *Windows Azure* y que funciona como un medio de ejecución para las aplicaciones, donde la constitución de la red y su mantenimiento proporcionan un conjunto de servicios que permiten el desarrollo, gestión y alojamiento de aplicaciones fuera del establecimiento. La plataforma *Windows Azure* se encuentra en los *datacenters* de Microsoft y está formado por una colección de nodos interconectados que constan de servidores, conexiones de alta velocidad, *switches* y máquinas virtuales (VM) corriendo Windows Server 2008 como sistema operativo. A parte también cuenta con un servicio CDN conocido como *Azure Content Delivery Network* (Microsoft, 2015) y que se basa en la tecnología de CDN de *Akamai*, este de igual manera se integra con *Azure* Storage para proporcionar alrededor del mundo una mejor experiencia al usuario sobre las descargas de archivos de gran tamaño y la entrega de contenido estático o dinámico con baja latencia.

3.6 TECNOLOGÍAS CLOUD - OPEN-SOURCE vs. PROPRIETARY

Muchos de los nuevos participantes en el mercado *Cloud* prefieren utilizar la tecnología *open-source* para implementar sus plataformas. Sin embargo, los principales actores del mercado como Amazon, Google y Microsoft emplean sistemas propietarios que los fueron perfeccionando desde que el mercado *Cloud* inició y es la razón de porque estas grandes empresas desarrollaron sofisticadas tecnologías *Cloud* propietarias y obtuvieron su aclamada ventaja competitiva; es por esto que para este tipo de empresas no es factible liberar y

compartir su tecnología haciéndolas *open-source*, ya que perderían su ventaja técnica (Gorelik, 2013). Por otro lado, los nuevos participantes del mercado han optado por aceptar la realidad y tomar una posición diferente, al reconocer que no pueden perder tiempo desarrollando sus propias tecnologías, ante el ya establecido mercado de *cloud* y el cual crece contantemente. Por tal razón la implementación de mecanismos *open-source* han aliviado la presión competitiva del mercado y se esperaría ver a no tan a largo plazo, que ambas tecnologías propietarias y de código abierto compitan por los nichos del mercado *Cloud*, hasta que el medio se consolide, estabilice y que algunas de las tecnologías propietarias también puedan ser de código abierto.

Al tener en cuenta el período de estabilización del mercado final, nos podríamos dar cuenta que, las empresas están prestando más atención a la innovación de procesos en lugar de la innovación de productos. Entendiendo que la innovación de productos es la creación de nuevos mercados o servicios, y la mejora significativa de un producto existente o servicio. Mientras que, la innovación de procesos significa cambios menores o mejoras en las capacidades del producto o servicio, tales como un nuevo método de entrega de servicio o función de software. Como un gran ejemplo, se podría hablar sobre Hewlett-Packard, ya que es una de las principales compañías que utiliza plataformas *cloud open-source* desde que ingresó en el mercado *Cloud* público en el 2012. Esta decisión de ir por el lado *open-source*, la tomó por la necesidad de construir rápidamente su plataforma *Cloud* y ponerse al día con los otros competidores del mercado que ya tenían implementada su infraestructura (Gorelik, 2013). Se conoce que Hewlett-Packard utiliza para su ejecución la plataforma *Cloud open-source* llamada *OpenStack*.

También se puede hablar sobre Netflix, que es otro de los grandes exponentes del mercado en implementar eficientemente durante su desarrollo una infraestructura CDN asistida por *Cloud*, donde ha sabido aprovechar la elasticidad y el beneficio de disminuir sus costos y precios finales a través de una operación formada por *web servers* de alto rendimiento con un software propietario conocido como (*OpenConnect*) y que está integrado con su sistema (*Netflix Content Delivery*) con módulos de características personalizadas y *open-source (OS: FreeBSD, Web Server: NGINX y BGP daemon: BIRD)*. Al ser Netflix un proveedor de contenido (CSP) puede reducir sus costos de operación mediante la adopción

de recursos similares (Pathan, Sitaraman & Robinson, 2014). La arquitectura *Cloud/CDN* de Netflix está constituida en gran parte a su medida, debido a la complejidad del servicio de *streaming* que provee a gran escala; comprendiendo que cada elemento de la infraestructura apunta a un factor de rendimiento específico para las necesidades únicas de algunas de las funciones del servicio y sabiendo que dicha arquitectura al mismo tiempo utiliza sus propios *datacenters*, los cuales los ha venido implementado con los años al fortalecerse como marca (Adhikari et al, 2012) (fig. 82).

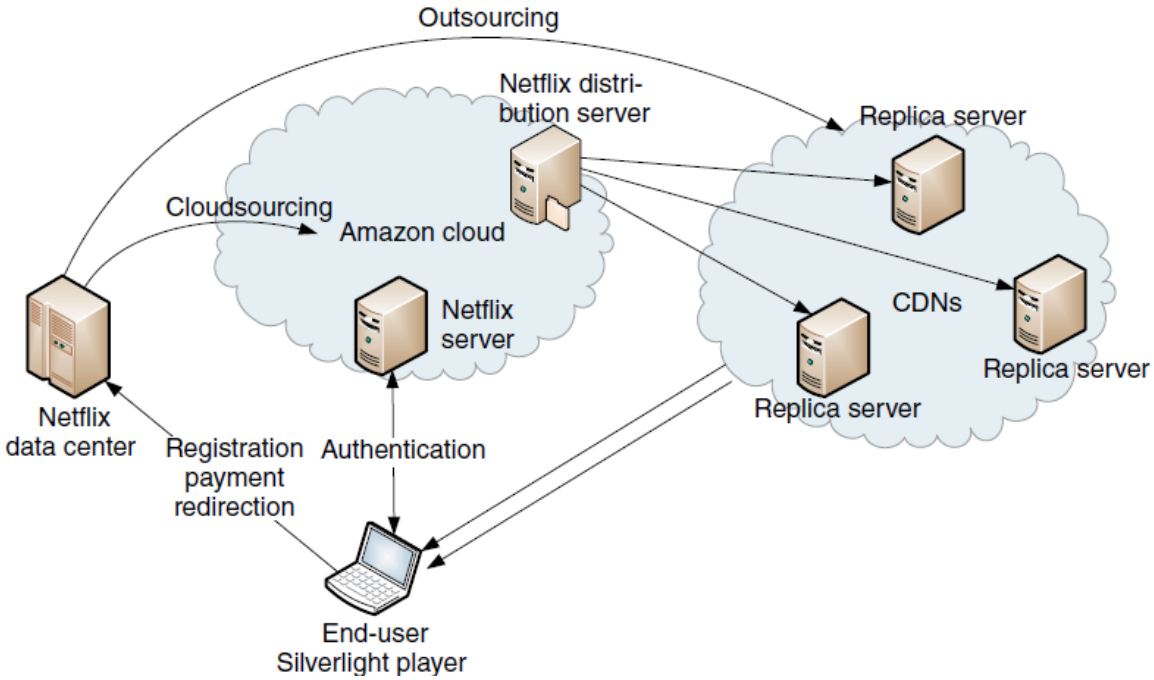


Figura 82. Video streaming platform de Netflix. (2012) (Fuente: Adhikari VK.)

Los *datacenters* de Netflix también son responsables del registro de nuevos usuarios, gestión de pagos y la redirección de los usuarios a un servidor de Netflix ubicado dentro del NCE de Amazon, en esta ubicación y servidores es donde se lleva a cabo la tarea de originar el contenido, mantiene un record de grabación y análisis, *Digital Rights Management* (DRM), enrutamiento CDN, registro de inicio de sesión del usuario, etc; haciendo que las infraestructuras de los diferentes CDNs desplieguen un número de *replica servers* que alojen

el mismo contenido de vídeo, pero con diferentes niveles de calidad (*bitrates*) y formatos. Dependiendo del ancho de banda disponible en el extremo del cliente, los servidores CDN pueden decidir si envían el contenido de vídeo con una velocidad alta o baja de bits, normalmente utilizando el protocolo *dynamic adaptive streaming over HTTP* (DASH) (Islam & Grégoire, 2012).

3.7 ARQUITECTURA CCDN

3.7.1 CCDN – Modelo 1

En este modelo de arquitectura de CCDN se empléa un *centralized brokerage*, donde el *CCDN broker* será un proveedor de servicios de terceros que actuará como interfaz entre el proveedor CCDN y los *Cloud service providers CPs/transit network provider*. El *CCDN broker* se encargará de crear el *intercluster content distribution graph* y el particionamiento entre los componentes del CP, que serán responsables de establecer una solución parcial del servicio CDN mediante la ejecución de una apropiada estrategia para el *surrogate placement*.

Las CDNs parciales que ya están interconectadas, son fusionadas por el *CCDN broker* para entregar la solución final de CCDN al solicitante, que para este caso será el *CCDN provider*. Cada uno de los actores involucrados dentro del planteamiento CCDN, tendrán sus propias prioridades en términos de costos, eficiencia y calidad de servicio QoS (Ericsson, 2012). Por otro lado, el *CCDN broker* deberá responder a los requerimientos planteados por el proveedor CCDN y reducir al mínimo el gasto total hasta el final de la implementación.

Tabla 8.

Resumen de conceptos básicos. (Fuente: Papagianni C.)

CCDN	Cloud-oriented Content Delivery Network.
CP	Cloud service Provider.
Transit network provider	Interconnects clouds belonging to different CPs with virtual links.
CCDN provider	Content Provider that deploys his own CDN over leased networked cloud resources.
Service Area	A geographically defined area where the CCDN provider grants content delivery services to end-users.
CCDN broker	Provides CCDN brokerage services to CCDN providers, based on the CCDN provider's service area, potential clientele, available CPs and transit network providers.
Service Cluster	Base CCDN service area unit as defined by the CCDN broker.
ASC	The Abstracted Service Cluster is a point representation of a service cluster, estimated as the center of mass of end-users with unit mass residing in the particular service cluster.

Sin embargo, al proporcionar un servicio CDN sobre una estructura de tipo *Cloud*, se requiere de la adopción de un apropiado esquema de colocación de los *edge servers*, teniendo en cuenta el entorno multiproveedor y la geolocalización de los potenciales usuarios. Esto se puede conseguir mediante el procedimiento de tres pasos, como se muestra a continuación:

- **Fase de Pre-diseño CCDN.**

Durante esta fase, la colocación del *surrogate server* sobre las *Clouds* interconectadas debe seguir la dispersión geográfica de los destinatarios del contenido, que es ya en los usuarios finales. Por lo tanto, el *CCDN provider* debe identificar a los potenciales clientes, en términos de ubicación dentro de un área de servicio geográficamente definida (Papagianni, 2013). El *CCDN broker* fragmentará el área de servicio *Grid* en una serie de subregiones más pequeñas, denominadas como *service clusters*. De acuerdo a este ejemplo de modelado de CCDN, se asumió un número de (k) *service clusters* sin considerar la eficiencia de la técnica de agrupación (*clustering*), esto debido a que se pueden utilizar varios enfoques alternativos de *clustering*, dependiendo de los detalles del servicio ofrecido, así como el modelo de negocio del proveedor o las bases impuestas por el cliente.

Cada *service cluster* se abstrae a un nodo llamado *Abstracted Service Cluster* (ASCk), el cual se caracteriza por las coordenadas de ubicación definidas por el centro de masa ASCk (x,y) de la agrupación del servicio estimado por la geolocalización de usuarios finales de CDN. Las propiedades adicionales de los ASCks son las características relacionadas con la

CDN de cada grupo de servicios, tales como: 1) los usuarios finales dentro del cluster, 2) tasa de solicitud y tamaño por usuario, y 3) el tamaño del contenido replicado. La abstracción mencionada anteriormente se representa en la figura 83, específicamente para el clúster de servicio 1. Para el conjunto de ASCks, un gráfico de malla parcial se establece en el marco propuesto, donde el ASCk resaltado indica el *origin server*, ya que proporciona rutas de distribución de contenidos redundantes desde el clúster del *origin server* hacia los *clusters* de servicios de toda la zona. Se pueden también adoptar topologías alternativas con diferentes características y propiedades (ej. topologías *treebased*).

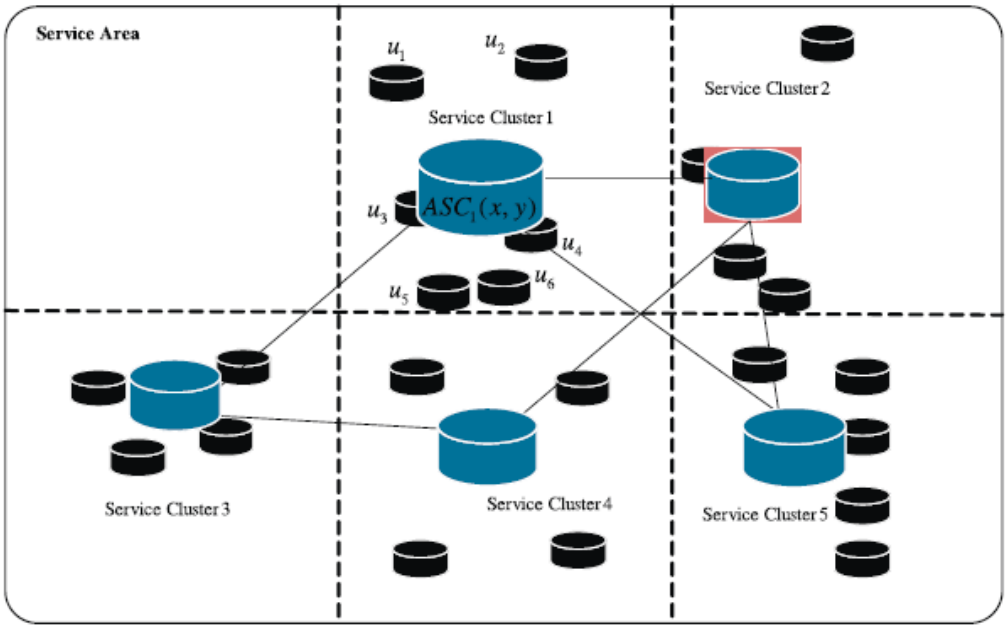


Figura 83. Descripción general de la arquitectura CCDN 1. Service area clustering e intercluster content distribution graph. (2013) (Fuente: Papagianni C.)

- **Clúster de Servicio en la Asignación de CP.**

Se asume la coexistencia de varios CPs competidores, proporcionando un sobrelapado total en la misma área de servicio. Durante este paso, la meta para el *CCDN Broker* será la de dividir de manera eficiente el gráfico de distribución de contenido *intercluster* entre los

distintos CPs. Cada *cluster* puede ser asignado a solamente un CP, mientras que cada CP puede servir a múltiples clústeres.

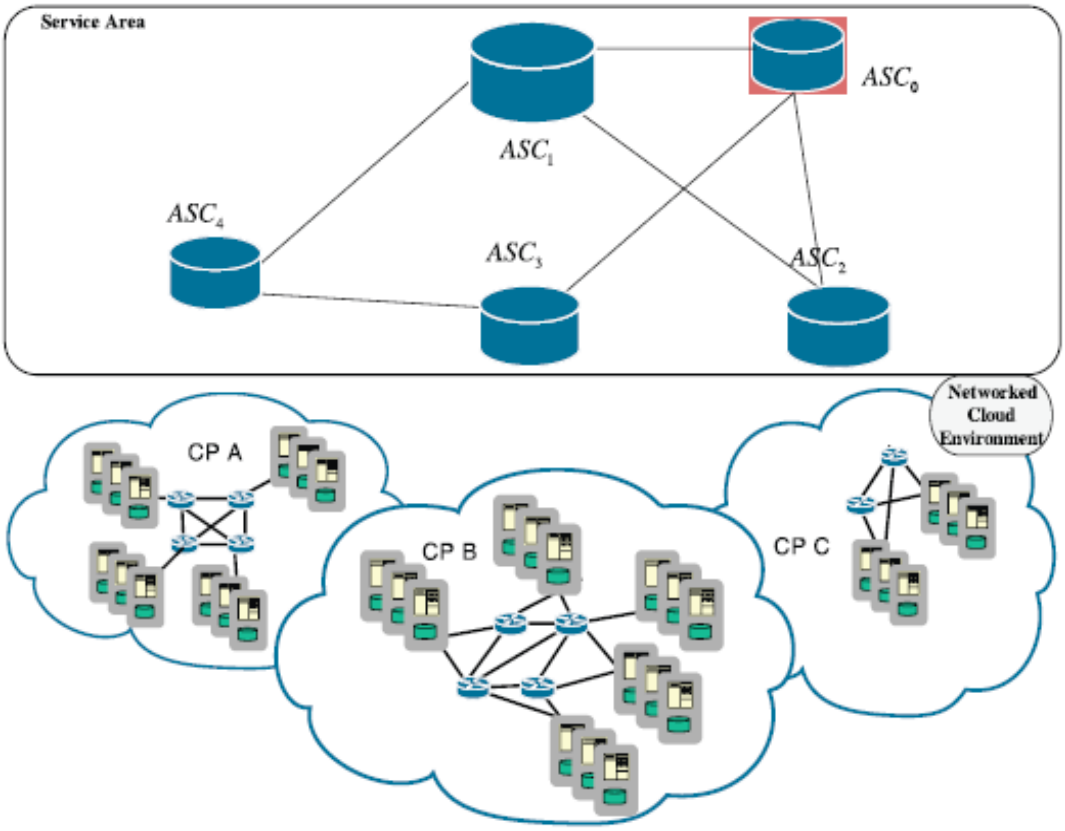


Figura 84. Intercluster content distribution graph over a networked cloud environment. (2013) (Fuente: Papagianni C.)

El *CCDN Broker* reenvía el gráfico de distribución de contenido *intercluster* extraído a los CPs (Fig. 84). Se pide a los CPs que provean al *broker* con información relacionada con el número de *cloud sites* que están disponibles por clúster de servicio, así como los costos de aprovisionamiento de unidades relacionadas a cada *cloud site*, lo que permite la fijación de precios diferenciados basados en la localidad del servicio. Teniendo como base esta información y el esquema de particiones gráfico adoptado (Ericsson, 2012), el *CCDN broker* llega a la conclusión de ser un *cluster* de servicio más rentable para la asignación al CP y

envía las partes correspondientes de la gráfica de distribución de contenido *intercluster* a los CPs seleccionados para su posterior procesamiento (Papagianni et al, 2013).

- **Ubicación de los VSS por Clúster de Servicio.**

Una vez que han sido asignados uno o más gráficos de distribución de contenido *intercluster* parcial con información sobre la ubicación de los usuarios y sus necesidades dentro de cada *cluster* de servicios, el CP ejecuta un adecuado algoritmo de ubicación de *surrogates* (fig. 85). En el marco propuesto, se introducen dos nuevos algoritmos: el VSP heurístico y el SNAGVSP heurístico. El objetivo del problema en cuestión es determinar el número y ubicación de las réplicas alojadas en un subconjunto de *cloud sites* disponibles dentro del dominio del CP, dentro del particular grupo de servicio, de tal modo que cada usuario final sea asignado a por lo menos un sitio y que los requisitos de calidad de servicio QoS para las peticiones de los usuarios finales sean satisfechas.

Uno de los *surrogate servers* dentro del *cluster* de servicio puede ser seleccionado para facilitar la comunicación entre el resto de los *surrogates* y el *origin server* que pertenece a un *cluster* de servicio diferente. Este servidor es seleccionado basándose en las proximidades de los ASC y los requerimientos de capacidad, al cual se lo conocerá de aquí en adelante como un *transit server*. El *CCDN broker* será responsable de informar al proveedor de CCDN y al proveedor de contenido el precio general y la información de implementación del servicio CCDN (Papagianni et al, 2013).

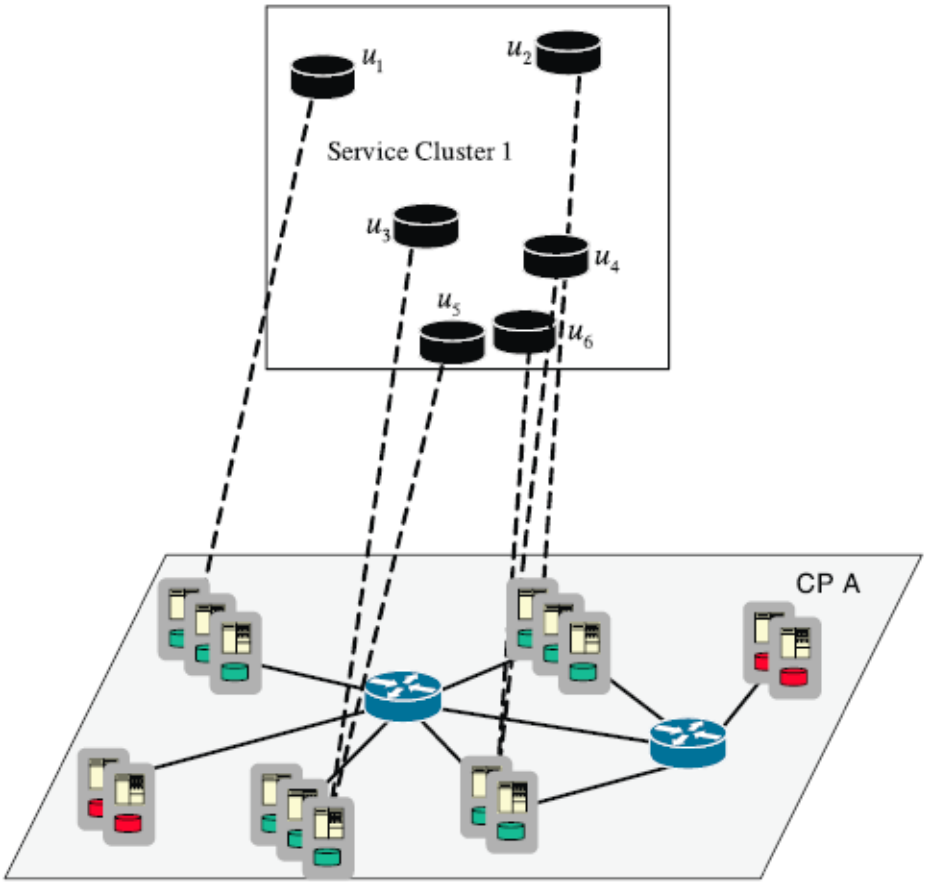


Figura 85. Replica placement in a networked cloud. (2013) (Fuente: Papagianni C.)

3.7.2 CCDN – Modelo 2

Este modelo de arquitectura consta de solo un *Master server* y varios *Partial servers* como se muestra en la figura 86. Cada uno de los *Partial servers* administrará tres o más *Point servers* y a su vez, un *point server* consta de *Cloud Storage servers* y *Cache servers*. Para este caso en particular, los *cloud storage server* pueden ofrecer un servicio de almacenamiento para los *Content Creators*, mientras que los *cache servers* podrían actuar como los "*edge points-servers*" para implementar el servicio CDN de los *content consumers* (Yuedui et al, 2011). Los archivos de contenido almacenados en los *Cloud storage servers* serán necesarios para que los *content creators* adopten el servicio de CCDN. Además, la entrega del contenido entre los *edge points* estará basada en la tecnología P2P.

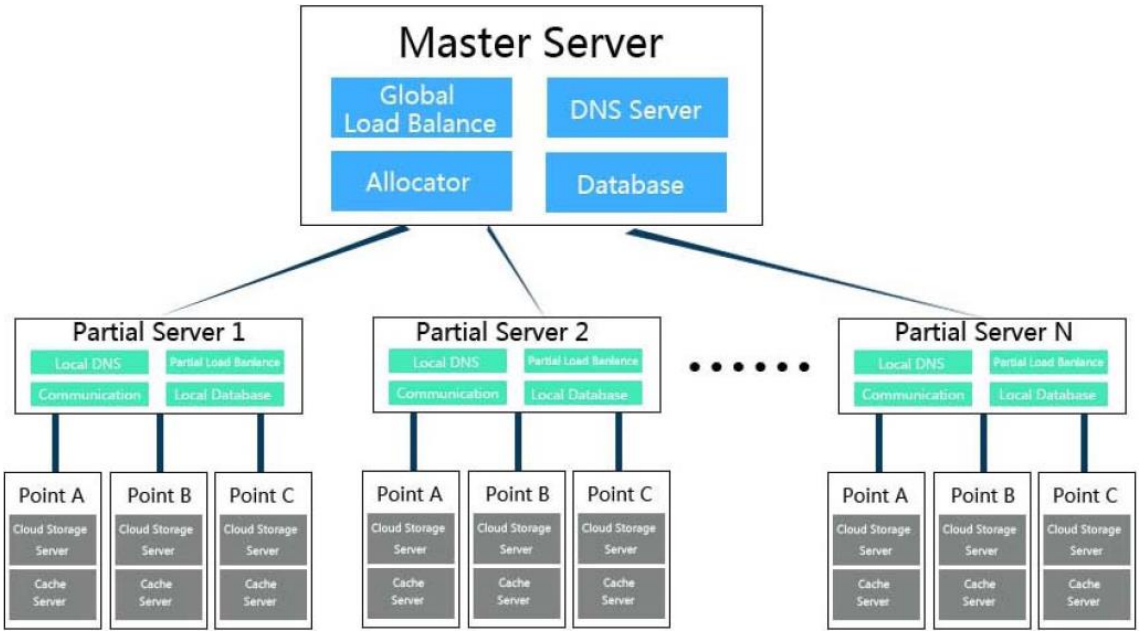


Figura 86. Descripción general arquitectura CCDN 2. (2011) (Fuente: Yuedui W.)

- ***Master Server***

Este servidor se compone de un módulo *Global Load Balance*, un módulo de *DNS server*, un módulo *Allocator* y de un módulo *Database*. El sistema debe tener una buena capacidad de ampliación o escalabilidad, por lo que el *Master Server* debe implementar un monitoreo en tiempo real del estado de los *Partial Servers* para asegurarse de que los puede detectar a tiempo a través de mensajes de unión y cancelación.

El módulo *Global Load Balance* implementa el balanceo de carga, de acuerdo con los mensajes de estado de los *Partial Servers*. El *Master server* no suele almacenar los archivos de usuario en sí, sino más bien todos los *Metadatos* asociados a los *Partial Servers* y los archivos de usuario, tales como las tablas de mapeo de la ubicación de los archivos de usuario, los mensajes de estado de los *Partial Servers* y su información de enrutamiento. Debido a que sólo existen flujos de control entre el *Master Server* y los *Partial Servers*, la carga de información en el *Master Server* se reduce considerablemente, lo que hace que el *Master Server* no se convierta en un cuello de botella para la información que maneja el sistema (Yuedui et al, 2011).

Cuando un usuario móvil realiza una visita *Wireless Application Protocol* (WAP) a través de las regiones, el *DNS Server* local del usuario es el encargado de cargar la solicitud analítica de ubicación del fichero visitado al módulo *DNS server*. Luego el módulo *DNS server* lo comparará con la tabla de mapeo de la ubicación de los archivos de usuario para asegurar la posición real del archivo y devolver un mensaje de localización al DNS local a través del módulo *Allocator*; una vez que se recibe el mensaje de la ubicación real del archivo, se levantará una comunicación con el *DNS server* al cual el archivo actual le pertenece.

- ***Partial Server***

Este servidor consta del módulo *Local DNS*, un módulo *Partial Load Balance*, un módulo *Communication* y del módulo *Local Database*. Este servidor no sólo almacena los archivos reales de los usuarios, sino también todos los *metadatos* asociados con los archivos y puntos de usuario, como las tablas de mapeo de las ubicaciones de los archivos de usuario, los mensajes de estado de los puntos de acceso y la información de enrutamiento de dichos puntos. A través del módulo *Partial Load Balance* implementa el balanceo de carga en el equipo local de acuerdo a los mensajes de estado de los puntos.

Cuando un usuario móvil realiza una visita WAP, el módulo *DNS local* será el primero en juzgar si la visita es una visita local o una visita regional, de acuerdo con el requerimiento analítico del DNS y la información de la ubicación de los archivos locales. Si la visita WAP es una visita local, el módulo *Local DNS* devolvería la ubicación del punto en el cual es grabado el archivo real para el usuario y entonces los usuarios pueden acceder a dicho archivo directamente en el punto local. Por otro lado, si la visita WAP es una visita regional, el *Local DNS* presentará una solicitud de análisis de ubicación al *Master Server*. Finalmente, el módulo *Communication* será el responsable de las comunicaciones específicas en el proceso de implementación del *content delivery* entre los puntos.

Con esta metodología, este sistema de CCDN puede mejorar el servicio de una CDN tradicional, mediante la difusión de múltiples *edge points* en todo el país. En este modelo los *edge points* pueden proporcionar servicio de CDN y también servicio de almacenamiento, haciendo que todas las visitas WAP puedan ser vistas como visitas locales, lo que hará que

los usuarios móviles gasten menos tiempo de espera para que se carge la página web en sus dispositivos. Mientras que, se empleará la tecnología P2P para la entrega de contenido entre los *edge points* del sistema, para garantizar la velocidad y calidad de los contenidos.

3.7.3 CCDN – Modelo 3

Este tercer esquema CCDN se muestra en la fig. 87, el cual estará basado en una Arquitectura *Master/Slave* dividido en diferentes regiones. En este esquema los *Content service providers* (CSP) ponen el contenido en la nube que este cerca de los usuarios finales y da acogida a por lo menos una réplica del mismo esquema en otra región. Por lo tanto, los CSP no necesitan mantener el servidor con el contenido de origen (*origin server*) porque los datos son copiados al *cloud storage*, el cual está en el nodo maestro. De acuerdo a la ubicación geográfica del mundo, el esquema CCDN se dividirá en varias regiones, donde cada región contendrá por lo menos un *surrogate* y un *slave;* y donde por lo menos uno de los *slaves* actuará como esclavo principal y será el responsable del almacenamiento en caché de un número fijo de contenido de *metadatos*, los cuales contendrán la información básica del contenido y su ubicación (Chia-Feng et al, 2011).

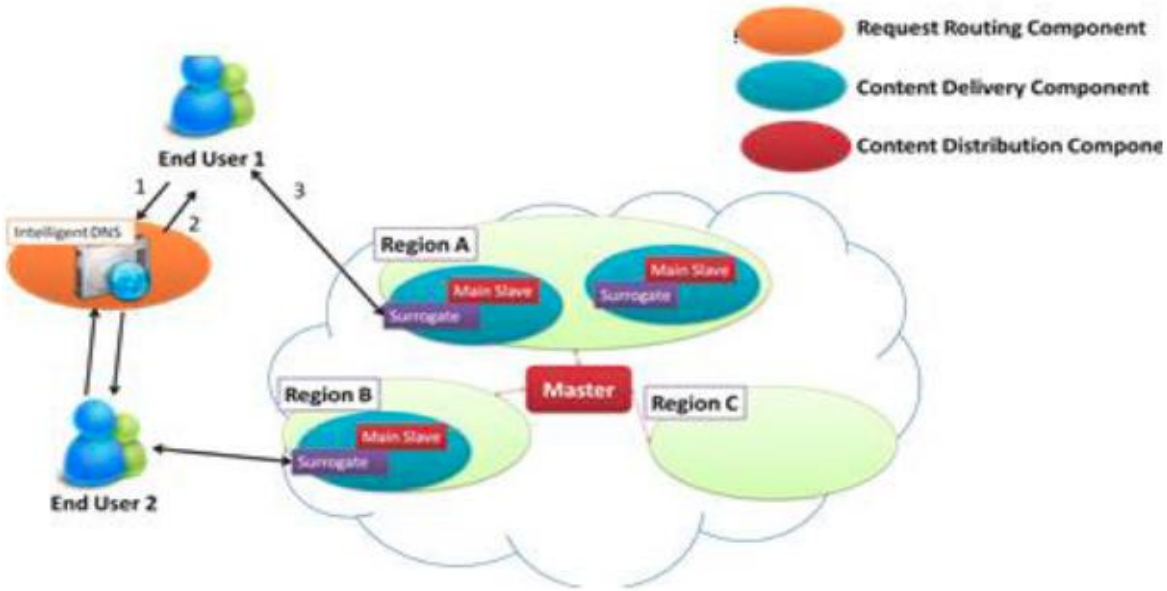


Figura 87. Descripción general de la arquitectura CCDN 3. (Chia-Feng L., 2011)

En este sistema hay tres componentes principales; el componente *Request Routing*, el componente *Content Delivery* y el componente *Content Distribution*.

- **Componente Request-routing**

Determina el enrutamiento optimizado para la solicitud del usuario final. Por lo tanto, la solicitud de acceso del usuario final es redirigida al *surrogate server* más cercano a través del método de enrutamiento de solicitudes basado en DNS (*DNS-based request routing*), el cual reduce el tiempo y mejora la calidad del acceso al servicio. El mecanismo inteligente de asignación por DNS entrega el contenido web al usuario desde el *surrogate* más cercano. Sin embargo, si hay más de dos *surrogates* en la región, el requerimiento del usuario es redirigido por el *load balancer* a un *surrogate* especifico, entonces el *surrogate* verificará si el contenido solicitado existe en el propio *Least Recently Used* (LRU) cache, a través de un *hash request url* que permite al sistema estar listo para evitar una pesada carga de procesamiento sobre un solo *surrogate* y optimizar el espacio en cache. (fig. 88).

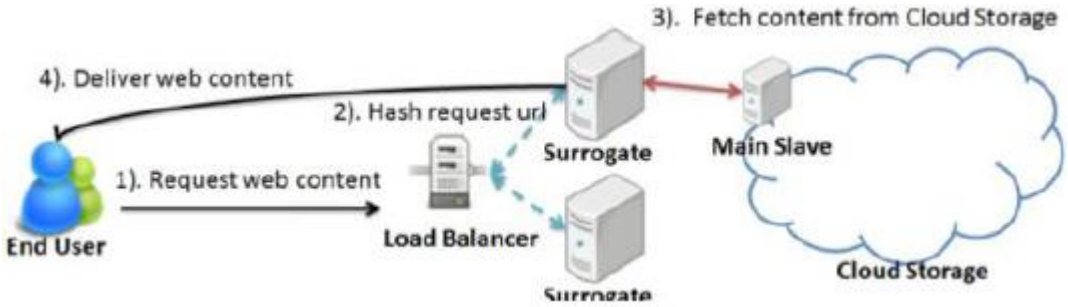


Figura 88. Procedimiento de recuperación de contenidos. (Chia-Feng L., 2011)

- **Componente Content Delivery**

La información descrita dentro de la figura 89, muestra la estructura del componente *Content Delivery*, el cual principalmente se encarga de entregar el contenido solicitado por los usuarios finales a través de un grupo de *surrogates*. En primer lugar, si hay una copia del contenido en la memoria caché del *surrogate,* este entregará directamente dicho contenido en respuesta a la solicitud que realizó el usuario final. Si no es así, el *surrogate* obtendrá el contenido solicitado del *cloud storage* del CCDN, a través del *slave* principal con las mejores condiciones de conectividad. Por último, el contenido se entregará de vuelta al usuario final y será almacenado en el caché del *surrogate*. El almacenamiento en cache de la *Metadata* también puede reducir el tiempo para encontrar donde está el contenido web, razón por la cual, si el cache lo solicita, el *surrogate* requerido no tendrá la necesidad de preguntar la localización del contenido en el nodo maestro. Para tolerar los fallos del *surrogate*, primero se chequea si el equipo *main slave* está respondiendo antes de que el *surrogate* entable comunicación (Chia-Feng et al, 2011). De ser el caso que el *main slave* no responda, el *surrogate* se comunicará con otro de los *main slaves* disponibles o el principal en la misma región. Luego que el *surrogarte* realizó la solicitud de requerimiento del contenido desde el *main slave* más saludable y cercano a él, entonces el *surrogate* replicará una copia en su propio LRU cache.

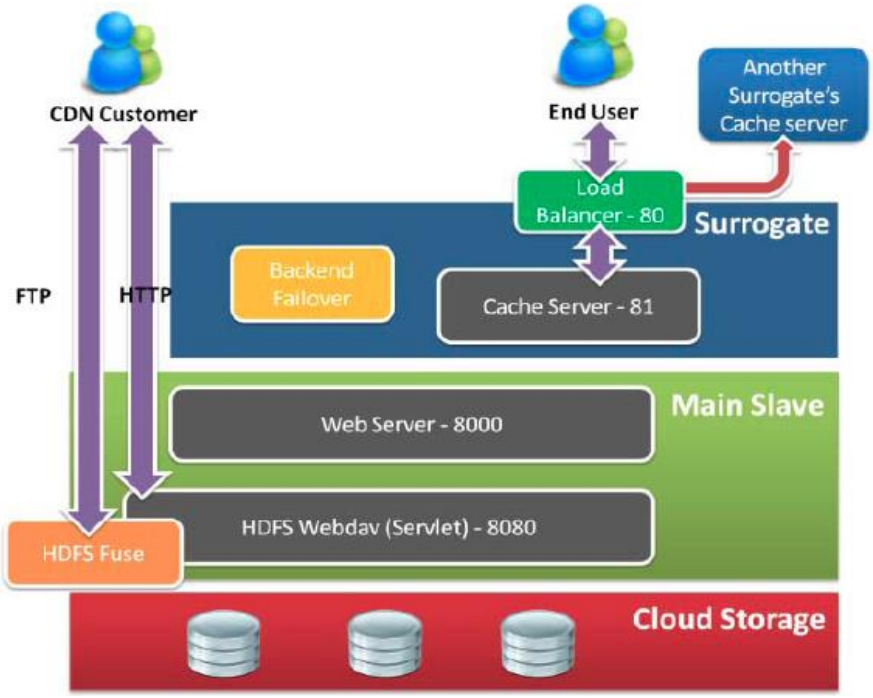


Figura 89. Componente Content Delivery. (2011) (Fuente: Chia-Feng L.)

Las condiciones del *surrogate* dependerán de las conversiones CDN que realice el *edge server* dentro de las regiones. Este componente es un *cache server* que es responsable de la comunicación con los usuarios finales y el equipo de replicación, siendo el papel clave de esta CCDN. Por otro lado, si una región tiene dos o más *surrogates*, entonces CCDN establecerá el mecanismo de balanceo de carga, el cual asegurará que los usuarios comunes de Internet con recursos informáticos limitados, no sean bombardeados con carga pesada y que dichas peticiones les sean asignadas por el método de *hash* (Chia-Feng et al, 2011). Además, CCDN tiene también un mecanismo a prueba de errores que asegura la continuidad del servicio en todo el sistema al momento que el *surrogate* busque el contenido solicitado desde el *cloud storage*.

Como procedimiento para resolver los problemas de *Flash Crowds*, la CCDN debe tener implementado el balanceo de carga sobre el *surrogate* y es por esto que se propone para este modelo el concepto de balanceo de carga regional, donde el *load balancer* redireccione las peticiones del usuario final a un *surrogate* especifico a través de una petición *hashing* de

URL, esto no solo incrementará el rango de precisión del cache, sino que tampoco tendrá la duplicación de una réplica de contenido en el cache, lo que es lo mejor en términos de un manejo adecuado y controlado del espacio. Con esta política, la réplica de un archivo casualmente no será distribuida a lo largo de las regiones, ya que un tercio de las réplicas están sobre un nodo, dos tercios de las réplicas están sobre una región y el otro tercio está comúnmente distribuido a lo largo de las regiones restantes. Por otro lado, esto no incrementa demasiado la latencia de red, debido a que los *surrogates* no estarán tan lejos los unos de los otros. Si el *hash* que realiza el *surrogate* falla, el *load balancer* va a recalcular el *hash* para asignar otro *surrogate* con buenas condiciones y garantizará que no existan cortes del servicio.

- **Componente Content Distribution**

Este componente garantiza por sí mismo la disponibilidad del contenido en el sistema, esto en razón a que el c*loud storage* replica el contenido a otros *slaves*, a través de la estrategia de replica propuesta *Region Awareness* (RA). Como se muestra en la fig. 90 cuando el factor de replicación es tres, la política de localización *RA* pondrá una réplica sobre un nodo de la región local, otra en un nodo diferente de la misma región y el último sobre el nodo en una región diferente y más cercana de la región local. Además este componente es utilizado con el fin de conseguir menores costos, ahorrar esfuerzos de mantenimiento en el despliegue del sitio web, aplicando procesos de *cloud storage* al host del contenido del cliente de CCDN y será quien entregue al cliente un nombre de dominio específico como el de la URL.

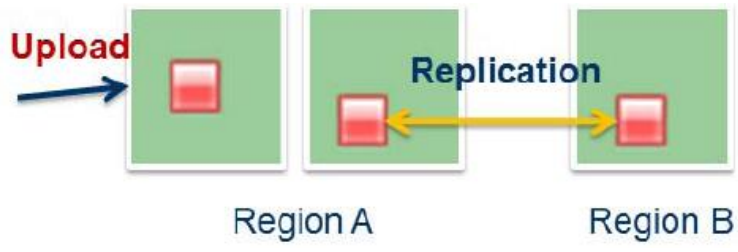


Figura 90. Procedimiento de detección de fallas. (2011) (Fuente: Chia-Feng L.)

Los CCDN *slaves* se encargarán de correr la operación de un cluster de computadoras que comúnmente están dispersas en muchas regiones y en donde la comunicación entre dos nodos de diferentes regiones tiene que ir a través de *switches*, haciendo que el ancho de banda de red entre las máquinas de la misma región sea mayor que el ancho de banda entre máquinas de diferentes regiones.

3.8 DISEÑO DEL NODO CCDN

Luego de comprender el potencial de la arquitectura CDN de Netflix, entender los diferentes métodos de integración hacia una infraestructura NCE analizados en la teoría de CDN y *Cloud Computing* realizados en los anteriores capítulos y haber comprendido la arquitectura de tres modelos Cloud/CDN durante el desarrollo de este capítulo referente al diseño del nodo CCDN; se ha conseguido sentar las bases para el diseño del concepto de nodo CCDN propuesto a través de la figura 91, para un *Contect Delivery Service Provider* (CDSP) con distribución optimizada de servicios multimedia, incluido el contenido interactivo, dinámico, estático y de streaming. Tomando como muestra la estructura del modelo 2, en este diseño el *content provider* pone el contenido en la nube, luego el *Master server* será el responsable de la distribución de dicho contenido, pero sin almacenar los archivos de usuario en sí, sino más bien todos los *metadatos* asociados a los *surrogates servers* de los *data nodes* y los archivos de usuario, tales como las tablas de mapeo de la ubicación de los archivos de usuario, los mensajes de estado y su información de enrutamiento. A diferencia del segundo modelo de arquitectura, este equipo es el que asigna un lugar apropiado para el contenido y ahorra un número fijo de copias en otras áreas, reduciendo la carga del equipo para manejar de mejor manera la información del sistema y el flujo del tráfico de control, haciendo que uno de los *surrogate servers* dentro del *cluster* de servicio pueda ser seleccionado para facilitar la comunicación entre el resto de los *surrogates* y el *origin server* que podría estar perteneciendo a un *cluster* de servicio diferente. El *content provider* no tiene que mantener el *origin server*, debido a que el *Master* será el responsable de la integridad de los datos

después de ser almacenados en la nube, donde para este caso, el *content provider origin server* que almacena las versiones originales y definitivas de sus archivos, puede ser visto como un cliente dentro de la plataforma *Cloud* tradicional (Ling, Xiaozhen & Yulan, 2013).

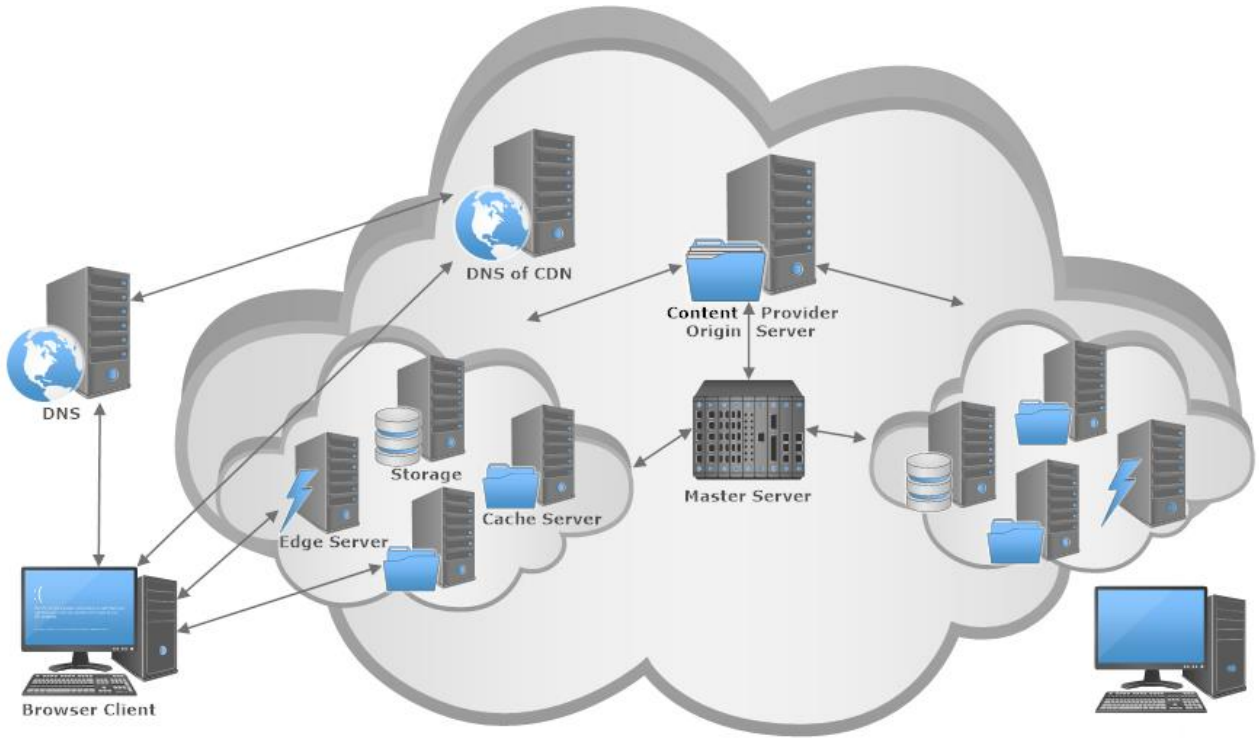


Figura 91. Descripción general del Nodo CCDN. (2016) (Fuente: Autor)

Cuando el *content provider origin server* quiere escribir datos en el *cache server*, este se comunica con el *Master* y obtiene la ubicación de los nodos de datos (*data nodes*) sobre los cuales se puede escribir la información y con esto ya se podría comunicar a los *data nodes* asignados para que almacenen el contenido. Para este modelo, los *data-nodes* se entienden como el *edge server* y el *cache server*, donde, dependiendo de las diferentes ubicaciones geográficas (red global de ubicaciones de borde), este CCDN estará dividido en diferentes zonas y cada zona tendrá al menos un *edge server* con un *cache server*; para reducir el tiempo y mejorar la calidad del acceso al servicio, a través de una forma sencilla de distribuir contenido a los usuarios finales con baja latencia y altas velocidades de transferencia de

datos; ya que a diferencia de lo que sucede en la plataforma *cloud* tradicional, este diseño del sistema CCDN se basa también en lo aprendido en el modelo 3, donde se agrega un *DNS server*, para lograr un mecanismo de redireccionamiento inteligente de asignación por DNS (*intelligent DNS*) al entregar el contenido web al usuario desde el *surrogate* más cercano a través del método de enrutamiento de solicitudes *DNS-based request routing* identificado durante el estudio de *Cloud Load Balancer*, haciendo que las solicitudes de contenido se redirijan de forma automática, a la ubicación de borde más cercanas, para que el contenido se entregue con el máximo desempeño posible.

Para iniciar con la descripción de los recursos lógicos y físicos del diseño del nodo CCDN, se seleccionó como mejor alternativa de integrador el modelo de servicios IaaS a través del empleo de la infraestructura EC2 de Amazon AWS, ya que cuenta con la mayor experiencia del mercado en este tipo de servicios y también por ser identificado como líder dentro del *cuadrante mágico de Gartner* para servicios de *cloud*; donde la parte privada del modelo de despliegue de esta *Hybrid Cloud* servirá de puente para la infraestructura entre los recursos virtuales del *content provider* (rebalanceo de cargas entre nubes), mientras que la parte pública será la que atenderá los requerimiento de los usuarios (entregando una aplicación SaaS privada que se basa en un IaaS público). De esta manera se emplea efectivamente lo aprendido en el modelo 1, donde a través de un correcto manejo de este mecanismo, la replicación de los contenidos se la puede realizar a los *datacenters* o *clusters* regionales mas cercanos, donde el servicio de almacenamiento en la nube podría tener un mecanismo de copia de seguridad remoto y acceso de contenido directo para el servicio CCDN, colocando el contenido lo más cerca posible del usuario sobre los *virtual surrogate servers*, de tal manera que a cada usuario final se le asigne por lo menos un *cloud site* que satisfaga los requisitos de QoS para las solicitudes. Para esto, dentro de la estructura de este modelo CCDN se pueden identificar algunos problemas que deben ser identificados en detalle, para poder sobrellevarlos:

- La entrega de contenido CDN puede elegir diferentes maneras de distribución de acuerdo a diversos contenidos: el contenido popular adopta el método *PUSH*; pero cuando la solicitud del contenido es dispersa, se adoptará el método *PULL*.

- En el sistema típico de CDN, cuando un usuario solicita contenido del *cache server* indicado y el contenido no se lo encuentra, el *cache server* tiende a realizar un *pull* del contenido especificado directamente del *origin server*. Este tipo de patrón hace que la tasa de utilización del ancho de banda del *origin server* aumente y también hace que se sobrecargue.

Con este análisis anterior, es en este momento que la tecnología de *Cloud Computing* aplicará un mecanismo llamado *pipelining copy*, el cual se emplea de tal manera que, cuando el cliente realiza la operación de escritura, el cliente transmite los datos al primer *data node* y por lo tanto el primer *data node* lo transmite al segundo *data node* y así sucesivamente hasta alcanzar el número regular de replicaciones (Ling, Xiaozhen & Yulan, 2013) lo que lo hace ser de característica CCDN. Ya teniendo identificada la operación de este mecanismo, si el *cache server* asignado carece del contenido solicitado y por otro lado el *edge server* tiene el contenido solicitado, el contenido puede ser obtenido desde este servidor y hace que este método pueda reducir el ancho de banda reservado de tal manera que minimiza los costos del ancho de banda del *origin server* y para esto se considerarán los siguientes pasos que fueron analizados en los apartados de *Request-routing algorithms* y *mechanism*s del capítulo de CDN:

- ***Global Load Balancing Algorithm***

Este algoritmo no solo debe considerar la información de ubicación geográfica para redirigir las peticiones de los usuarios, los otros factores también son importantes, tales como: la carga de nodos opcionales, la situación del tráfico de la red y si el *cache server* puede proporcionar el contenido solicitado. Todos estos factores se basan en el criterio de *Quality of Service* (QoS), mientras que desde otra perspectiva, de acuerdo al enfoque en el que se basa, dichos factores no sólo se basarían en criterios de QoS, sino que también en criterios de *Quality of Experience* (QoE) que a la final representa la percepción y satisfacción de los usuarios finales (Tran, Mellouk & Hoceint, 2011). Por ejemplo, los usuarios finales tendrán requisitos

específicos en términos de QoS, los proveedores CCDN necesitarán minimizar los costos de despliegue del servicio CCDN, teniendo como tope el precio del ecosistema NCE de multiproveedor, mientras que los CPs tendrán que utilizar eficientemente sus recursos para maximizar el reintegro de su inversión.

- ***Local Load Balancing Algorithm***

Con este algoritmo se puede alcanzar una apropiada aplicación en el sistema, ya que puede reflejar con precisión el estado de la red y la capacidad de procesamiento de cada servidor, evitando la sobrecarga del *cache server*. Utilizando el algoritmo *round robin* se asignará solicitudes a cada *cache server*, debido a que este algoritmo es simple y fácil de implementar, pero sólo puede ser aplicado a los servidores cuyas configuraciones de software y hardware sean similares, lo que hace que el buscar una optimización del *local load balancing algorithm* sea un problema importante que requiere ser considerado durante una implementación. Por lo que se podría utilizar como opción el servicio de *Elastic Load Balancing* de Amazon, el cual permite crear fácilmente un punto de entrada expuesto a Internet en su *Virtual Private Cloud* (VPC) o equilibrar el tráfico entre capas de la aplicación dentro de su VPC, ya que puede asignar grupos de seguridad al balanceador de carga para controlar qué puertos están abiertos a una lista de fuentes permitidas, por tal razón al crear un balanceador de carga en su VPC, se puede especificar si estará expuesto a Internet (opción predeterminada) o será nativo.

- ***Video Data Fragment Caching***

Aquí entra en marcha toda la teoría del ecosistema de video con CDN analizado durante el capítulo de CDN, ya que en este diseño del mecanismo de *Caching*, al integrarlo con el *Cloud Load Balancer* puede determinar la calidad del servicio del *streaming* multimedia de los medios de comunicación, de tal manera que se entregue el contenido multimedia bajo demanda usando el *streaming* del protocolo de mensajería en tiempo real (RTMP) a través de *Amazon CloudFront* y donde la copia original de los archivos multimedia se almacenará

en *Amazon S3* y se utilizará *Amazon CloudFront* para entregar contenido multimedia con baja latencia. Razón por la cual el *CCDN provider* tiene que elegir un mecanismo específico de caché, basado en las características de los medios de transmisión y el comportamiento parcial de cada una de las características del usuario. Poniendo como ejemplo, se conoce estadísticamente que la mayor parte del tiempo los usuarios no ven el vídeo completo, la mayoría de los usuarios de un servicio de vídeo generalmente ven una pequeña parte del contenido visual que se ha solicitado. Por lo tanto, cuando el *content provider origin server* almacena el archivo en el *cache server*, este sólo necesita almacenar el prefijo del audio y del vídeo del medio que se reprodujo y el resto del contenido será almacenado en caché hasta cuando el usuario reproduzca el contenido.

Este mecanismo puede hacer que se utilice más eficientemente el *cache server* reduciendo el tiempo de respuesta del usuario, pero para proceder con la implementación de este mecanismo de almacenamiento caché, el contenido proporcionado por el *origin server* debe dividirse en fragmentos. Por esto, para el mecanismo de *cloud storage* el *Master* tendrá divido el archivo en bloques de datos con tamaño fijo de 64MB y cuando un cliente introduzca los datos, este mecanismo será propicio para el despliegue del *fragment caching*.

3.8.1 Infraestructura del Nodo CCDN

Durante todo el desarrollo de los conceptos de la teoría se hizo referencia a que el modelo IaaS será el empleado para el diseño del nodo CCDN antes propuesto y que se utilizará el servicio *AWS EC2* de Amazon al ser el más óptimo en términos técnico-económicos, donde todas las características y beneficios del *Cloud Computing* estarían implícitos como servicio con características como (*On-Demand Capabilities*, *Elasticity*, *Multi-Tenancy*, etc) y harían la diferencia en la prestación de servicios CDN. Para esto se empleará *micro* instancias EC2 de Amazon que serán desplegadas en un principio por separado en diferentes localidades y finalmente donde se requiera el servicio. Se podría solicitar que la *micro* instancia EC2 de Amazon esté con la configuración por defecto: el sistema operativo Linux de 32 bits, 1 GB de memoria, 20 GB de disco. Tanto al *surrogate server* como al *slave server* ubicados dentro

de cada instancia de Amazon EC2 se los conocerá como los *edge servers*. Se requerirá también del uso de tres servidores que serán desplegados en el sistema, como: el *master server*, *edge server* y el *intelligent DNS* (Chia-Feng et al, 2011).

En tal razón, cada *cluster* lógico estará alojado en la infraestructura del *Cloud Provider (CP)*, a quien se solicitará un modelo de servicio (IaaS); para este ejemplo será el EC2 y con las siguientes características para los recursos físicos y lógicos (fig. 92, 93 y 94):

3.8.1.1 Recursos Físicos

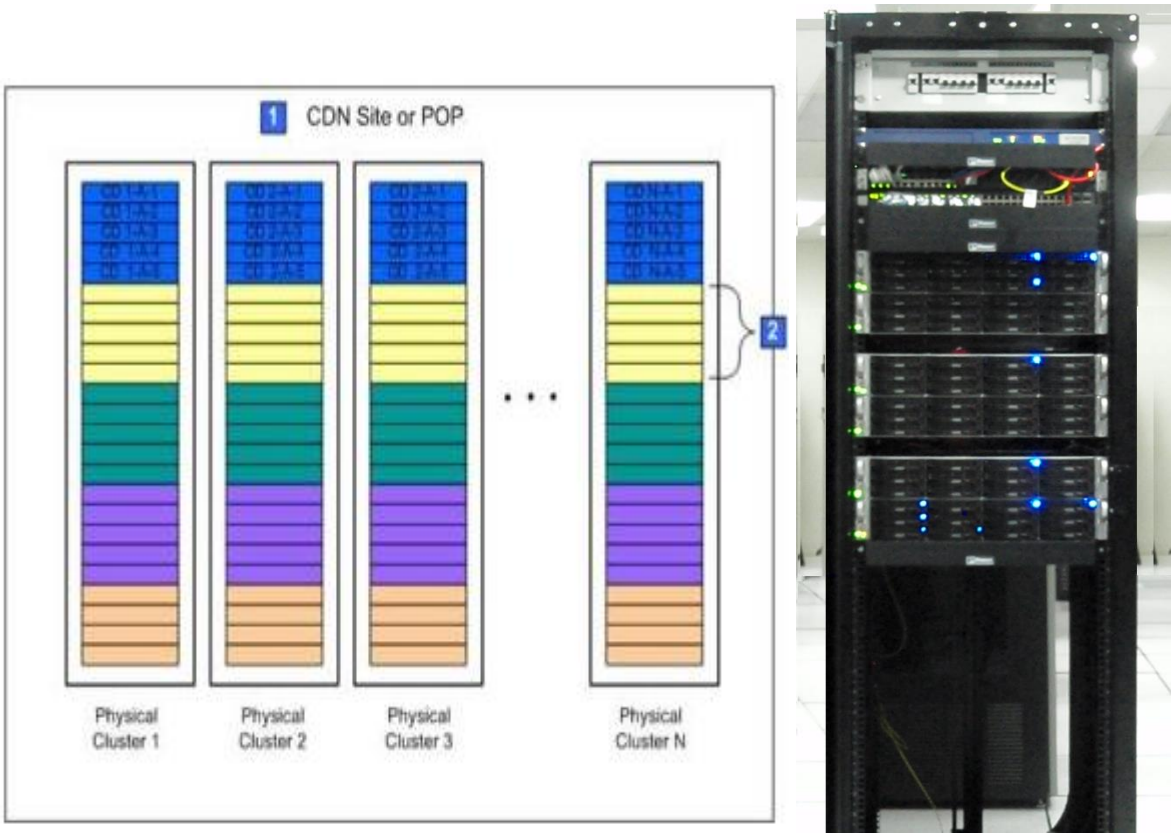


Figura 92. Recursos físicos del Nodo CCDN. (2015) (Fuente: Level3.com)

1) Nodo - *site/POP*

El Nodo será la ubicación física donde la infraestructura CDN estará implementada y esta dependerá de la estructura del proveedor que se contrate (ej. AWS), puede contar con una configuración con varios *clusters* físicos, en el que cada uno trabaje de forma independiente y autónoma como parte de un sistema cohesivo que atiende las solicitudes entrantes de contenido de los usuarios. Cada *cluster* físico está conectado al *backbone* del proveedor, que podría llegar a tener de momento una estructura que soporte hasta 20 Gbps por *cluster* como capacidad en la red y 1 Gbps por cada *Content Distributor server* (CD).

Un típico clúster físico completamente configurado podría estar compuesto de veinticuatro servidores de alta velocidad que ejecutan *Kernels* Linux de 64 bits altamente acondicionados y en muchos de los casos como Netflix con software CDN propietario; mientras que para el procesamiento de transacciones de almacenamiento en caché, existirán reglas inteligentes, informes, estadísticas y funciones administrativas (Level 3, 2015).

2) *Bindigns*

Son recursos que se describen como el tipo de unión que conceptualmente permite que varios propietarios utilicen los diferentes subconjuntos de toda la infraestructura de red:

- Para facilitar el aprovisionamiento, cada propietario será parte del *cluster* lógico.
- Grupos o *stripes* de *clusters* lógicos son pre-asignados en todas las regiones.
- Las políticas predeterminadas gestionan las agrupaciones a través de los *stripes*.
- Se hacen excepciones para contenido único, repositorios, tipos de tráfico y tamaños.
- Los grupos de unión pueden ser compartidos a través de los tipos de tráfico similares o complementarios de cliente.
- La capacidad puede ser reutilizada o ajustada para nuevas *stripes* si es necesario y para una mayor eficiencia en el tipo de tráfico que sirven.

3.8.1.2 Content Distributor Servers

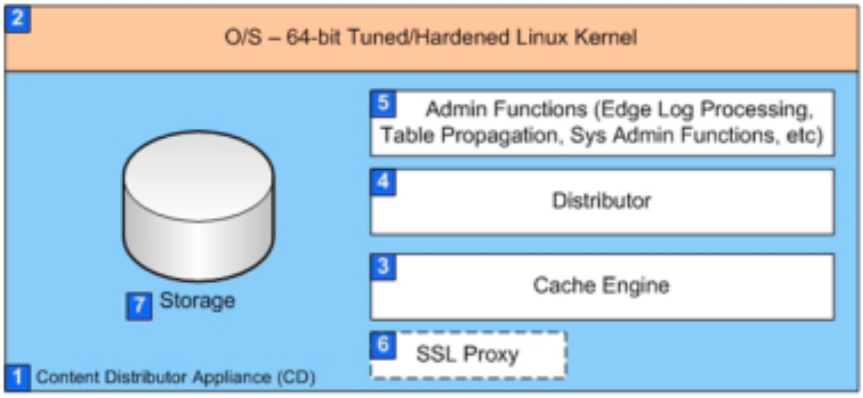


Figura 93. Estructura del servidor CCDN. (2015) (Fuente: Level3.com)

1) *Content Distributor Appliance* (CD)

Esta instancia proporciona procesos y funciones especializadas que entregan a la CCDN una flexibilidad y características superiores. Entre ellas se encuentra la *RuleBase* que asigna características especializadas de almacenamiento en caché y de entrega para los recursos seleccionados, en base a criterios flexibles, tales como patrones de URL, tipos de *MIME* y otras características. Cada *Content Distributor appliance* (CD) incluye un *Cache Engine* y el *Content Distributor* (Level 3 Media Portal, 2015). El *Cache Engine* es el que almacena y sirve a los recursos basados en la web, dando al servicio CDN su velocidad y eficiencia.

- Mejora de la gestión de activos basados en web.
- Proporciona una mayor eficiencia en CDN.
- Tiene una funcionalidad "*CDN-side*" especializada,
- Reduce la codificación HTML y los requerimientos de gestión del sitio web.

2) Sistema Operativo (OS)

Está basado en un *kernel* Linux de 64-bit modificado, con algunas características, como:

- Un alto nivel de optimización, dado que implementa balanceo de carga local personalizado y provee herramientas de *firewall* nativo.
- Es altamente acondicionado para manejar recursos de red, memoria, espacio en disco; con el propósito de mejorar su rendimiento y capacidad total.
- Puede contar con un paquete de instalación minimalistico, con capacidades de seguridad y administración para funciones propias de *caching*.

3) *Cache Engine*

Es un *software* altamente modificado que controla el *caching* del contenido y responde a los requerimientos entrantes. Está basado en *Squid proxy cache*.

4) *Distributor*

Es el *software* que controla la comunicación de los *Content Distributor* (CD) con toda la red y maneja varias funciones específicas de CDN (Level 3, 2015), como son:

- Identificativo del cliente.
- Establece conexiones persistentes de tipo "*side-channel*" con los "*Master Controllers*" para administrar las funciones de comando y control (tablas, invalidaciones, estadísticas, etc).
- Maneja la administración de la *metadata* y muchas funciones "*rulebase*".
- Maneja una condición especial del *cache* para el procesamiento y la autenticación.

5) Funciones Administrativas

Aquí se manejan varios *scripts*, programas, utilidades, con funciones especiales y administrativas.

- Realiza un pre-procesamiento de archivos de registro para consolidación, propagación y recuperación.
- Realiza funciones y rutinas de administración del sistema.

6) *SSL Proxy*

Dentro de la red algunos *Content Distributors* (CDs) están configurados para soportar SSL, para la entrega segura de contenido sensible a través de HTTPS y un componente extra con las siguientes características:

- Escuchan en el puerto SSL (443)
- Manejan sesiones de configuración, certificación, encriptación, etc.
- Maneja los requerimientos que se realizan al *Cache Engine* como un requerimiento normal de entrada (hereda la designación del *proxy*).

7) *Storage*

Cuenta con un almacenamiento en disco de los recursos de librería que entraron en el *cache* (Level 3 Media Portal, 2015).

- Los recursos dentro de su librería son indexados al interior del *cache engine* utilizando funciones de *hashing*, para una rápida recuperación.
- El almacenamiento es compartido entre todos los clientes del servidor.

- El sistema operativo y el *cache engine* están optimizados para minimizar la latencia del almacenamiento, recuperar y maximizar la capacidad de los requerimientos de ingreso que van a ser servidos.
- Para este caso el *storage* estará siendo soportado por la infraestructura *Cloud* y su flexibilidad de almacenamiento.

3.8.1.3 Recursos Lógicos

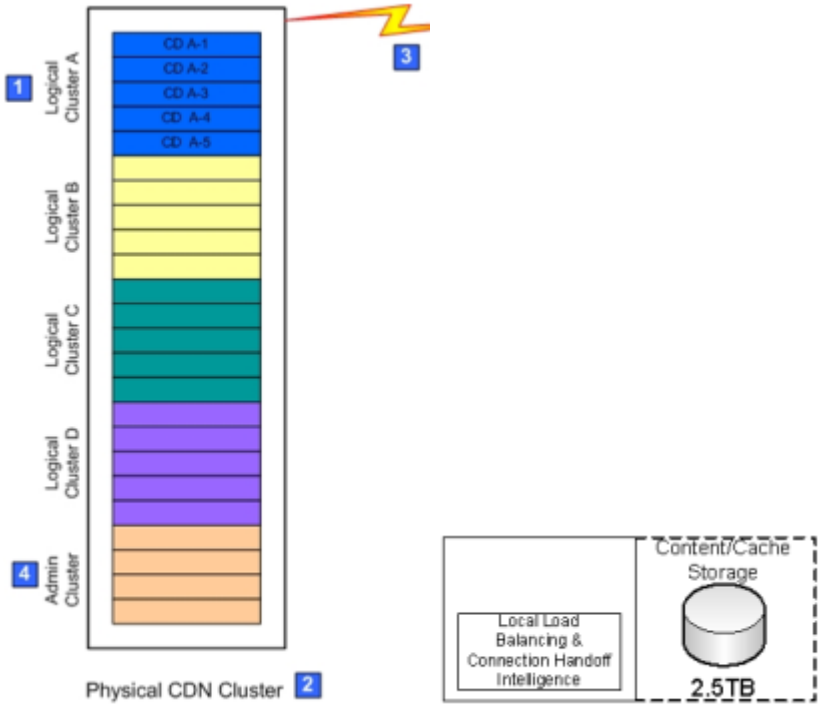


Figura 94. Recursos lógicos del Nodo CCDN. (2015) (Fuente: Level3.com)

1) Clúster Lógico

Cada *cluster* físico será sub-divido en varios *clusters* lógicos. Un *cluster* lógico representa la unidad direccionable más pequeña de la capacidad de CDN (Level 3, 2015). Será variable el

número de servidores dentro de un *clúster* lógico y por lo tanto el número de *clusters* lógicos dentro de un *clúster* físico; a su vez puede ser ajustado para acomodarse a las necesidades específicas de una aplicación y sus requerimientos de capacidad, trabajando en condiciones normales; sin embargo, un *cluster* lógico sería direccionado por una única dirección IP virtual (VIP) y contener cinco *load-balanced Content Distributor servers* (CDs). Además el registro de las cuentas de los clientes ingresara a la red a través de los *cluster* lógicos.

Cada servidor estará configurado con 2,5 terabytes de almacenamiento de contenido con funciones de alto rendimiento, donde el contenido almacenado en *caché* se distribuye eficientemente para una máxima capacidad y procesamiento.

2) *Site/POP, Bindigns* y *Content Distributor Servers* (analizados)

3) *Network Uplink*

Será la conexión de red de alta capacidad que se tiene contra el *service provider* (variable).

- Se podría exigir al proveedor un puerto de *Uplink* de 10Gps de capacidad, por cada *cluster* físico completo.
- Normalmente tiene una capacidad de 1Gps dependiendo de la capacidad física del *Content Distributor*.

4) *Admin Cluster*

Está constituido por un *pool* de cuatro servidores configurados para funciones administrativas y coordinaciones de red (Level 3 Media Portal, 2015):

- Consta de servidores *Best Distributor Selection* (BDS) y de servidores *Domain Name Services* (DNS), para medición de red y servidores de generación de tablas.
- También tiene servidores para registros, recuperación y funciones de *Intelligent Traffic Management* (ITM)
- Puede proporcionar funciones *Web Services* para servicios *GeoIP*.
- Todos los servidores utilizan la misma configuración de *hardware*, esto proporciona una flexibilidad importante durante la configuración y capacidades de conmutación ante errores de implementación.

3.9 RESULTADOS Y ANÁLISIS DE LA INTEGRACIÓN

La metodología de integrar los servicios de CDN con un entorno Cloud (NCE) abre muchas oportunidades de negocio y servicios de valor agregado, como los que se describe a continuación:

Dentro del entorno *Computing*, los recursos informáticos estarán siempre disponibles mediante el aprovechamiento de la infraestructura de *Cloud Computing* y su característica de ser flexible y adaptarse ante los diferentes escenarios "*Elasticity*", donde el acceso a un mayor poder de computo de procesos y la capacidad de almacenamiento es casi ilimitada, debido a que no estará únicamente limitado por un sistema de "caches" como lo hace el CDN tradicional, sino que despliega recursos adicionales a medida que la demanda de servicios lo requiere, logrando un sistema CCDN integral y que brinda servicios lo más cerca de donde se origina un requerimiento de usuario.

Como se explicó, la integración a CCDN trae beneficios a nivel del almacenamiento (*Storage*) ante un tipo especial de servicio con gran exigencia, como es el despliegue y manejo de contenido de video, ya que se utilizará mayores capacidades de almacenamiento, a medida que el contenido se almacena a diferentes resoluciones y formatos. Y algo que trae un enorme beneficio al entorno CDN tradicional, es que los servidores CDN tendrán un mejor

desempeño al no tener una sobre carga de procesamiento en almacenar dicho contenido de video, ya que estará soportado en todo momento por el entorno *Cloud*.

La constitución de un entono CCDN facilitará la distribución del contenido interactivo con baja latencia y menores tiempos de respuesta, ya que tiene una baja utilización de recursos al procesar el contenido desde a dentro de la infraestructura *cloud CPaaS*, ya que no tiene que tratar únicamente con un proveedor de servicio *Cloud/CDN* independiente con presencia global, para replicarlo en todo el mundo, permitiendo finalmente que múltiples instancias de CDN desplieguen el contenido interactivo.

Cuando se habla de costos, CCDN permite un desempeño equilibrado entre seguridad y facturación para no incurrir en gastos adicionales a través de herramientas de *cloud computing*, donde un usuario final siempre estará siendo autenticado por el proveedor ISP para ingresar a una red con contenido restringido, facilitando el establecimiento y control de los derechos de acceso y *Digital Rights Management* (DRM); al contrario del servicio de facturación de un proveedor comercial de CDN tradicional. Teniendo en consideración que el ancho de banda total de CCDN es la suma de todos los anchos de banda proporcionados por todos los *replica servers* del entorno *Cloud*, así se logrará una mayor calidad de servicio (QoS) y experiencia (QoE) para el usuario final, al tener el acceso al contenido con cobertura prácticamente local en todo momento.

3.10 FUTURAS INVESTIGACIONES

CCDN será escalable al desplegar servicios en lugares con mayor eficiencia de accesibilidad, razón por la cual será importante manejar el inconveniente de costos de CDN durante las decisiones relacionadas en la ubicación de los *surrogate servers*, se podrían desplegar nuevos servicios como *Edgesuite* y al mismo tiempo utilizando múltiples *edge clouds* o *multitier cloud* que abrirán caminos a nuevas formas de elasticidad para *cloud*.

Teniendo en cuenta los recientes avances en el *mobile content networking* (ej. WAP, IPv6) del modelo CCDN 2, donde la infraestructura de *mobile* CDNs puede desempeñar un papel de liderazgo en términos de explotación de nuevos avances tecnológicos sobre la Web para conexiones inalámbricas 4G. Al ofrecer servicios integrales y sin interrupciones donde el contenido y cálculo computacional deban ser desplegados cerca de los usuarios, mientras que el soporte podría ser centralizado para evitar la duplicación de información.

Finalmente, con la experiencia de buenas implementaciones se podrían lograr varios enfoques y acuerdos de interconexión (*Peering CDN*), los cuales de momento ya se están llevando a cabo y ganando popularidad entre los investigadores de la comunidad científica. Sin embargo, durante futuras investigaciones se tratarán varios temas críticos, sobre como evaluar las mejores prácticas de distribución, las nuevas técnicas para la medición de la carga, el manejo de las solicitudes de redireccionamiento y la replicación dentro del *Peering CDN*.

4. CONCLUSIONES Y RECOMENDACIONES

Durante el desarrollo de este trabajo se buscó, investigó y analizó una gran cantidad de material de estudio con referencias relacionadas a las ventajas de servicio que ofrece *Cloud Computing* y la tecnología CDN, para así lograr descubrir todos los beneficios de integrar ambas tecnologías y sus mecanismos internos en un sistema con arquitectura CCDN de bajo costo, escalable, con alto rendimiento y tolerante a fallos, para una adecuada distribución del contenido. La integración de tecnologías permitirá que las aplicaciones con amplia difusión de redes y gran utilización de ancho de banda que tienen por sentado una base sólida para el servicio de *video-on-demand* (VoD), continuen creciendo y atrayendo nuevas inversiones hacia esta importante y rentable industria, la cual se espera que supere el 94% por ciento del tráfico de consumo mundial para el año 2016.

La problemática general que se estudió y concluyó durante el marco jerárquico para el despliegue de CCDN, esencialmente se fue descomponiendo en subproblemas como su diseño, la correcta ubicación de los *web servers* réplicas (*surrogates*) dentro del entorno *networked cloud environment* (NCE) de multiproveedor, junto con la construcción de eficientes vías de distribución y baratos servicios de *storage* dentro de las infraestructuras lógico/físicas de alguno de proveedores CDN (*brokers*) del mercado, para así minimizar el costo incurrido en los servicios prestados. A través del diseño de este nodo CCDN, se planteó una arquitectura multi-nivel que permitirá optimizar los factores relacionados con la red, incluyendo la ubicación distribuida de los *virtual surrogate servers* (VSS) dentro de la nube de red y los factores propios relacionados con *Cloud*, donde se incluyen la capacidad del sistema virtualizado, es decir, el número requerido de servidores, switches, routers virtuales y cantidad de almacenamiento (*storage*) necesario en cada *point of presence (POP)* y utilizando de manera eficiente el ancho de banda de los *edge nodes*, para controlar proactivamente la capacidad de los *surrogates* inactivos de la red; de esta manera se identificó, que se puede reducir significativamente la carga de tráfico en los nodos de *back-bone*, disminuyendo en gran medida los requisitos de los servidores y los terminales de usuario, haciendo que el costo del sistema se reduzca significativamente, de tal modo que la arquitectura CCDN sea muy escalable y tenga un gran potencial dentro el futuro de los sistemas de vídeo de gran escala.

Con base sobre todo lo analizado, se concluyó que el diseño del nodo CCDN es más adecuado para los sitios web que utilizan aplicaciones de tráfico crítico como el vídeo, al igual que para otros sitios web con una alta demanda y utilización de tráfico dinámico. Haciendo que los usuarios que visualizan los sitios web de estos clientes (*content creators*), utilicen esta tecnología CCDN para que puedan experimentar un menor retardo, con un ancho de banda más amplio, distribuido y una mayor calidad de servicio (QoS) para todas sus peticiones, al mismo tiempo que se improvisa la calidad de la experiencia (QoE) percibida por el usuario. En tal razón, se concluyó que todo el sistema CCDN entrega un mejor rendimiento, al utilizar eficientemente los algoritmos y mecanismos de balanceo de carga utilizados en CDN dentro de la plataforma en la nube, a través de un "*Cloud Load Balancer*", debido a que se puede conseguir que los saltos del contenido desde una red a otra, entre el servidor y el cliente (*data nodes*) resulten ser más óptimos, logrando la mayor eficiencia

posible para implementar un servicio CDN sobre una plataforma *Cloud*. De esta manera resultó interesante comprender que mediante el empleo del método de *DNS-based request routing* sobre los servidores CCDN, los cuales para este diseño estarían configurados dentro de las instancias virtuales del servicio *Cloud EC2* de Amazon, se consigue un eficaz y veloz mecanismo de recuperación de contenidos, de tal manera que se mantiene siempre un adecuado equilibrio entre el volumen de las solicitudes y el número de *surrogates* disponibles para manejar dichas peticiones; debido a que en esta arquitectura CCDN se consideró que, serán los usuarios en gran parte quienes tendrán la responsabilidad para mantener la infraestructura CCDN, ya que a mayores exigencias, se lograrán alcanzar mejores estándares de servicio con buenos beneficio, grandes ingresos y la recuperación de la inversión.

De igual manera se identificó que cualquier cliente de CCDN al tener este esquema de acceso continuo a la nube, podría administrar fácilmente los contenidos mediante el uso de cualquiera de los protocolos FTP o HTTP, por lo que es importante considerar que para todo el sistema es imprescindible proteger la integridad y la seguridad del contenido que estará siendo distribuido a través de la CCDN, ya que el contenido está descentralizado y es vulnerable ante los ataques de *hackers*; por tal razón se consideró aplicar efectivas medidas de seguridad y protocolos contra las posibles formas de manipulación del contenido sobre los *surrogates*. En conclusión, un cliente CDN podría fácilmente subir sus contenidos a la *CCDN Cloud Storage*, donde se distribuirá el contenido desde los *surrogates* esclavos, de acuerdo con la estrategia regional del diseño y la infraestructura lógico-física planteadas, así los resultados del modelo muestran que al tomar en cuenta la popularidad del contenido, se puede replicar el contenido en ciertos lugares de la nube, de tal modo que el tráfico generado se entregue a los usuarios finales a través del número mínimo de saltos, obteniendo una solución más eficiente e inteligente para el servicio de contenido, optimizando recursos y la utilización de energía (*green computing*) en comparación con la centralización de todo el servicio en una sola ubicación, especialmente en períodos de carga máxima.

Por otro lado, se recomienda profundizar los estudios referentes a CCDN, ya que se ha logrado que este tipo de fusión tecnológica muestre la dirección correcta para el desarrollo de cualquier red futura y nuevas formas de elasticidad, dentro o entre las nubes, ya que ante todos los beneficios expuestos, este servicio tendrá un espacio de aplicación más amplia

dentro de varios sectores de investigación y comercialización, como el que se ofrece dentro de alguna de las próximas soluciones de *multicloud, multiple edge clouds* o *multitier clouds* como el *Distributed Rate Limiting* (DRL) o varios sistemas *P2P-based VoD* (ej. *PPLive*, *PPStream y Joost*) y todo el soporte para aquellos subscriptores que se basen en dichas soluciones. Además, se recomienda la implementación de CCDN sobre muchos sitios web pequeños que no pueden pagar los altos costos de la experiencia tradicional de CDN, para que así puedan experimentar los beneficios de este tipo de tecnología de aceleración, fomentando su utilización, lo que hará finalmente que incremente aún más su popularidad y se estandarizen procesos más sencillos de implementación que lleguen a más usuarios.

CRONOGRAMA

El Cronograma fue elaborado en un diagrama de Gantt en Microsoft Project. Para una mejor la visualización, observar en Anexo.

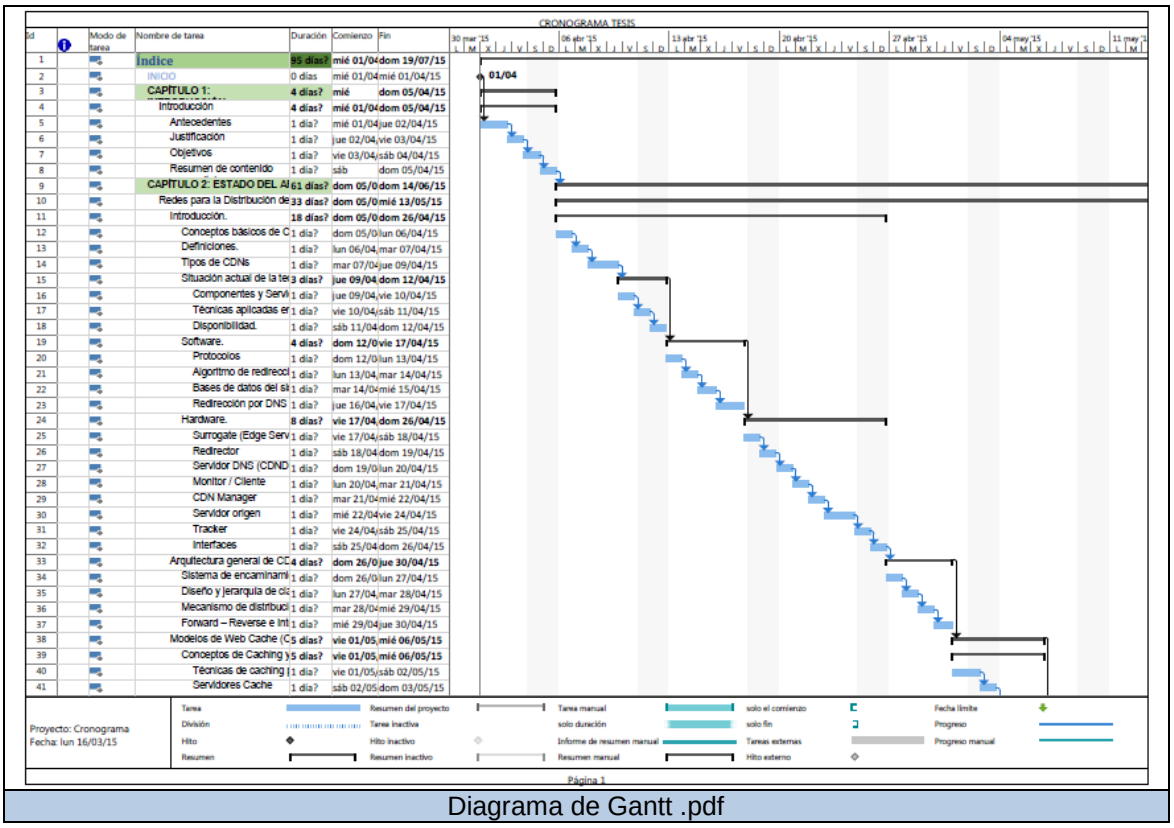


Diagrama de Gantt .pdf

REFERENCIAS BIBLIOGRAFICAS:

[1] Ling L., Xiaozhen M. & Yulan H. (Trad.) (2013). CDN Cloud: *A Novel Scheme for Combining CDN and Cloud Computing*, Institute of Communication Engineering, Jilin University.

[2] C., Leivadeas A. & Papavassiliou S. (Trad.) (2013). IEEE Transactions On Dependable And Secure Computing, *A Cloud-Oriented Content Delivery Network Paradigm: Modeling and Assessment*, VOL. 10, NO. 5, septiembre/octubre

[3] Chen F., Guoy K., Liny J. & La Porta T. (Trad.) (2012). Proceedings IEEE INFOCOM. *Intra-cloud Lightning: Building CDNs in the Cloud. Department of Computer Science and Engineering.* Pennsylvania State University.

[4] Zhang Z. (Trad.) (2014). Feel Free to Cache. *Towards an Open CDN Architecture for Cloud-based Content Distribution*. Dept. of Computer Science & Engineering, University of Minnesota.

[5] Lin C., Leu M., Chang C. & Yuan S. (Trad.) (2011). International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery, *The Study and Methods for Cloud based CDN*. Department of Computer Science, National Chiao Tung University, Hsinchu, Taiwan.

[6] Ramachnadran N. & Sivaprakasam P. (Trad.) (2010). *The Imminent Convergence of the Technology Trio: Demystifying the Super Potential of 4G, CDN and Cloud Computing*. Computer Centre, Indian Institue of Management, Kozhikode, India

[7] Yale L., Yushi S. & Yudong L. (Trad.) (2012). *Utilizing Content Delivery Network in Cloud Computing*. Western Washington University, USA.

[8] Selim E., Bertrand M., Tayeb L. (Trad.) (2013). Next-Generation Networking Symposium. *A Bidirectional Network Collaboration Interface for CDNs and Cloud Services Traffic Optimization*, Orange Labs.

[9] Li H., Zhong L., Liu J., Li B. & Xu K. (Trad.) (2011). IEEE 4th International Conference on Cloud Computing. *Cost-effective Partial Migration of VoD Services to Content Clouds*, Hong Kong University of Science & Technology.

[10] Okamoto Y., Noguchi S., Matsuura S., et al (Trad.) (2012). IEEE/IPSJ 12th International Symposium on Applications and the Internet. *Koshien-Cloud: Operations of Distributed Cloud as A Large Scale Web Contents Distribution Platform*. Nara Institute of Science and Technology, Japan.

[11] Buyya R. (Trad.) (2009). *Market-Oriented Cloud Computing: Vision, Hype, and Reality of Delivering Computing as the 5th Utility*. The University of Melbourne, Australia.

[12] Wang Y., Wen X., Sun Y., et al (Trad.) (2011). International Conference on Network Computing and Information Security. *The Content Delivery Network System based on Cloud Storage*. Beijing University.

[13] Duan Q., Yan Y. & Vasilakos A. (Trad.) (2012). IEEE Transactions On Network And Service Management. *A Survey on Service-Oriented Network Virtualization Toward Convergence of Networking and Cloud Computing*. VOL. 9, NO. 4, diciembre.

[14] Kepes B. (Trad.) (2011). *Cloud U. Revolution Not Evolution*, How Cloud Computing Differs from Traditional IT and Why it Matters. Diversity Limited.

[15] Rackspace Support Network (2015), *Understanding the Cloud Computing Stack: SaaS, PaaS, IaaS*. Recuperado el 20 Julio de 2015.
http://www.rackspace.com/knowledge_center/whitepaper/understanding-the-cloud-computing-stack-saas-paas-iaas

[16] Kugler K., Chard K., Caton S., Rana O. & Katz S. (Trad.) (2013). *Constructing a Social Content Delivery Network for eScience*, Social Data Sharing. Cardiff Univ., UK

[17] Jarabek C. & Wang M. (Trad.) (2011). *Can P2P Help the Cloud Go Green?*, Energy Model, A Hybrid P2P-Cloud System.

[18] Halpert B. (Trad.) (2011). *Auditing Cloud Computing, A Security and Privacy Guide - Introduction to Cloud Computing*.

[19] Buyya R., Pathan M. & Vakali A. (Trad.) (2008). *Content Delivery Networks* - Chapter 1 Content Delivery Networks: State of the Art, Insights, and Imperatives. Berlin.

[20] Hofmann M. & Beaumont L. (2005). *Content Networking - Architecture, Protocols, and Practice* - pp. 13.

[21] Cardellini V., Casalicchio E., Colajanni M., & Yu P. (2002). *The state of the art in locally distributed web-server systems*. ACM Computing Surveys, 34(2):263–311.

[22] Bartolini N., Casalicchio E. & Tucci S. (2004). *A Walk through Content Delivery Networks* - Motivations for Content Delivery - pp. 4.

[23] Verma D. (2002). *Content Distribution Networks*: An engineering approach. Wiley Inter-Science.

[24] Gilmore S., Hillston J. & Kloul L. (2004). PEPA Nets. In M. Calzarossa and E. Gelenbe, editors, *Performance Tools and Applications to Networked Systems*, volume 2965 of Lecture Notes in Computer Science. Springer.

[25] Shen W. (2010). "*A New Streaming Media Network Architecture Based on the Fusion of P2P and CDN*". China.

[26] Drake C. (2012). *Content delivery networks: Market dynamics and growth perspectives*. Transforming CDN market brings new opportunities for all participants, Informa Telecoms & Media UK.

[27] Schwarz B., (2012). *Content Delivery Networks 3.0* - Historical CDN perspective. Broadpeak.

[28] Pathan M., Sitaraman R., & Robinson D. (2014). *Advanced Content Delivery, Streaming, And Cloud Services.* Types of CDNs, United States.

[29] Generation Equity Advisors, LLC TechMediaMergers.com (2014). Aaron Solganick, CEO - *Strategic Buyer Universe for CDN's*. United States.

[30] Molina B. M. (2013). *Estudio, análisis y desarrollo de una red de distribución de contenido y su algoritmo de redirección de usuarios para servicios web y streaming*, Valencia, España.

[31] Pathan M. & Buyya R. (2009). *A Taxonomy and Survey of Content Delivery Networks - Grid Computing and Distributed Systems (GRIDS)*, Taxonomy. Australia.

[32] Pallis G. & Vakali A. (2006). *Insight and Perspectives for CONTENT DELIVERY NETWORKS: Historical Background*. Greece, Vol. 49, No. 1.

[33] CDN Market Share: *A Complete Business Analysis 2004 and 2005*. AccuStream iMedia Research; www.researchandmarkets.com | www.irg-intl.com.

[34] Jung, Y. et al. (2002). *Flash crowds and denial of service attacks*: *Characterization and implications for CDNs and Web sites*. In Proceedings of the 11th International World Wide Web Conference, pp. 293–304. Hawaii.

[35] Cieslak M., Foster D., Tiwana G., & Wilson R. (2013). "*Web Cache Coordination Protocol Version 2*". Recuperado Agosto 2015.
http://www.webcache.com/Writings/Internet-Drafts/draft-wilson-wrec-wccp-v2-00.txt
http://ietfreport.isoc.org/all-ids/draft-wilson-wrec-wccp-v2-00.txt

[36] Leech M., Ganis M., Lee Y., Kuris R., Koblas D. & L. Jones, (1996) "*SOCKS Protocol Version 5*" Internet Engineering Task Force RFC 1928. www.ietf.org/rfc/rfc1928.txt

[37] Vixie P., & Wessels D. (2000). "*Hyper Text Caching Protocol (HTCP/0.0)*" Internet Engineering Task Force RFC 2756. www.ietf.org/rfc/rfc2756.txt

[38] Valloppillil V. & Ross K. W. (1998) "*Cache Array Routing Protocol v1.0*" Internet Draft.

[39] Wessels D. & Claffy K. (1997). "*Internet Cache Protocol (ICP) version 2*" Internet Engineering Task Force RFC 2186. www.ietf.org/rfc/rfc2186.txt

[40] Hamilton M., Rousskov A. & Wessels D. (1998). "*Cache Digest Specification – version 5*". http://www.squid-cache.org/CacheDigest/cache-digest-v5.txt

[41] Bartal Y. (1996). *Probabilistic approximation of metric space and its algorithmic applications*. In Proc. of 37th Annual IEEE Symposium on Foundations of Computer Science.

[42] Jamin S., Jin C. et al. (2000). *On the placement of Internet instrumentation*. In Proc. Of IEEE INFOCOM, Tel-Aviv, pp. 295–304. Israel.

[43] Krishnan P., Raz D. et al (2000). *The cache location problem*. IEEE/ACM Transaction on Networking, Vol. 8, No. 5.

[44] Jamin, S., Jin C. et al (2001). *Constrained mirror placement on the Internet*. INFOCOM 2001. 20th Annual Joint Conference of the IEEE Computer and Comm. Societies. Proceedings, vol.1, pp.31-40, doi: 10.1109/INFCOM.2001.916684, Alaska, USA.

[45] Radoslavov P., Govindan R. et al (2001). *Topology-informed Internet replica placement.* In Proc. of Sixth International Workshop on Web Caching and Content Distribution, Boston, Massachusetts.

[46] Qiu L., Padmanabhan, V. N. et al (2001). *On the placement of Web server replicas*. In Proc. of IEEE INFOCOM, Anchorage, Alaska, USA, pp. 1587–1596.

[47] Li B., Golin, M. J. et al (1999). *On the optimal placement of Web proxies in the Internet.* In Proc. of IEEE INFOCOM, pp. 1282–1290. NY, USA

[48] Chen Y., Katz, R. H. et al (2002). *Dynamic replica placement for scalable content delivery.* In Proc. of International Workshop on Peer-to-Peer Systems (IPTPS 02), LNCS 2429, Springer-Verlag, pp. 306–318.

[49] Vakali A., Pallis G (2003). *Content delivery networks*: status and trends. IEEE Internet Computing, 7(6), IEEE Computer Society, pp. 68–74.

[50] Douglis F. & Kaashoek M. F. (2001). *Scalable Internet services*. IEEE Internet Computing, Vol. 5, No. 4, pp. 36–37.

[51] Akamai Technologies (2007), Inc., http://www.akamai.com/

[52] Dilley B. & Maggs M., et al. (2002). *Globally distributed content delivery*, IEEE Internet Computing, pp. 50-58.

[53] Pallis, G. & Vakali, A. (2006). *Insight and perspectives for content delivery networks.* Communications of the ACM, 49(1), ACM Press, pp. 101–106. NY, USA.

[54] Kangasharju J., Roberts J., et al. (2002). *Object replication strategies in content distribution networks.* Computer Communications, Vol. 25, No. 4, pp. 367–383.

[55] Chen Y., Qiu L. et al. (2003). *Efficient and adaptive Web replication using content clustering*. IEEE Journal on Selected Areas in Communications, vol.21, no.6, pp. 979-994,doi: 10.1109/JSAC.2003.814608.

[56] Fujita N., Ishikawa Y. et al (2004). *Coarse-grain replica management strategies for dynamic replication of Web contents*. Computer Networks: The International Journal of Computer and Telecom. Networking, 45(1), pp. 19–34.

[57] Johnson K.L., Carr, J.F., et al. (2001). *The measured performance of Content Distribution Networks*, In 5th International Workshop on Web Caching and Content Distribution, Vol. 24, No. 2, pp. 202-206, Lisbon, Portugal.

[58] Katsaros D., Pallis G. et al (2009). *CDNs Content Outsourcing via Generalized Communities*. IEEE Transactions on Knowledge ands Data Engineering, vol. 21, n. 1.

[59] Freedman M., Freudenthal E. & Mazières D. (2004). "*Democratizing Content Publication with Coral*" In Proceedings of 1st USENIX/ACM Symposium on Networked Systems Design and Implementation, San Francisco, CA.

[60] Tse, S.S.H. (2005). *Approximate algorithms for document placement in distributed Web servers*. 7th International Symposium on Parallel Architectures, Algorithms and Networks, Vol. 16, No. 6, pp. 489-496 & pp. 61- 66, doi: 10.1109/ISPAN.2004.1300458.

[61] Pallis, G., Vakali, A. et al. (2005). *A Latency-Based Object Placement Approach in Content Distribution Networks*. Third Latin American Web Congress, pp. 8, 140-147, doi: 10.1109/LAWEB.2005.3. IEEE Press, Buenos Aires, Argentina.

[62] G. Pallis, K. Stamos et al. (2006). *Replication Based on Objects Load under a Content Distribution Network*. 22nd International Conference on Data Engineering Workshops, pp.53, doi: 10.1109/ICDEW.2006.127. Web Information Retrieval and Integration (WIRI), Atlanta, Georgia, USA.

[63] Stamos K., Pallis G. et al (2006). *Integrating Caching Techniques on a Content Distribution Network*. In Proc. of 10th East-European Conference on Advances in Databases and Information Systems (ADBIS 2006), Springer-Verlag, Thessaloniki, Greece.

[64] Stamos K., Pallis G. et al (2006), "*A Similarity-Based Approach for Integrated Web Caching and Content Replication in CDNs*," In Proceedings of 10th International Databased Engineering and Applications Symposium, IEEE Press, New Delhi, India.

[65] Bakiras S. & Loukopoulos T. (2005). *Combining Replica Placement and Caching Techniques in Content Distribution Networks*. Computer Communications, Vol. 28, No. 9, pp. 1062-1073.

[66] Jian N., Tsang, D.H.K. et al (2003). *Hierarchical Content Routing in Large-Scale Multimedia Content Delivery Network*. IEEE International Conference on Communications, vol.2, pp. 854- 859, doi: 10.1109/ICC.2003.1204454

[67] Wessels D. & Claffy K. (1997). *Internet Cache Protocol (ICP), version 2*. Internet Engineering Task Force RFC 2186. www.ietf.org/rfc/rfc2186.txt

[68] Rousskov & D. Wessels, (1998) "*Cache Digests*," Computer Networks and ISDN Systems, Vol. 30, No. 22-3, pp. 2155-2168.

[69] Gadde S., Rabinovich M., et al (1997). *Reduce, Reuse, Recycle: An Approach to Building Large Internet Caches*. In Proc. of 6th Workshop on Hot Topics in Operating Systems, pp. 93–98.

[70] Karger D., Sherman A. et al (1999). *Web Caching with Consistent Hashing," Computer Networks.* Computer networks, ol. 31, No. 11-16, pp. 1203-1213.

[71] Valloppillil V., & Ross K. W. (1998). "*Cache Array Routing Protocol v1.0*" Internet Draft.

[72] Ni J. & Tsang D., (2005). "*Large Scale Cooperative Caching and Application-level Multicast in Multimedia Content Delivery Networks*" IEEE Communications, Vol. 43, Issue. 5, pp. 98-105.

[73] Ni J., et al. (2003). "*Hierarchical Content Routing in Large-Scale Multimedia Content Delivery Network*" In Proceedings of IEEE International Conference on Communications, (ICC '03), Vol. 2, pp. 854-859.

[74] Gayek P., Nesbitt R. et al (2004). *A Web content serving utility*. IBM Systems Journal, 43(1), pp. 43–63.

[75] Chen C., Ling Y. et al (2005). *Scalable Request-Routing with Next-Neighbor Laod Sharing in Multi-Server Environments*. 19th International Conference on Advanced Information Networking and Applications, IEEE Computer Society, vol.1, pp. 441- 446, doi: 10.1109/AINA.2005.303. Washington, USA.

[76] Sivasubramanian S., Szymaniak, M. et al (2004). *Replication of Web Hosting Systems. ACM Computing Surveys systems*, ACM Press, Vol. 36, No. 3. NY, USA.

[77] Wang L., Pai V. S. & Peterson L., (2002). *The Effectiveness of Request Redirection on CDN Robustness*. 5th Symposium on Operating Systems Design and Implementation, pp. 345-360. Boston, MA, USA.

[78] Aggarwal A. & Rabinovich M., (1998) "*Performance of Dynamic Replication Schemes for an Internet Hosting Service*" Technical Report, HA6177000-981030-01-TM, AT&T Research Labs, Florham Park, NJ, USA.

[79] Balakrishnan H., Kaashoek, M. et al (2003). *Looking Up Data in P2P Systems*. Communications of the ACM Press, Vol. 46, No. 2, pp. 43–48. NY, USA.

[80] Delgadillo K., (1997) "*Cisco Distributed Director*," Cisco White Paper, Cisco Systems.

[81] Pierre G. & Van Steen, M. (2006). *Globule: A Collaborative Content Delivery Network*. IEEE Communications Magazine, vol.44, no.8, pp.127-133, doi: 10.1109/MCOM.2006.1678120.

[82] Szymaniak M., Pierre G., et al, (2003). *Netairt: A DNS-based Redirection System for Apache.* In Proceedings of International Conference WWW/Internet, Algrave, Portugal.

[83] Huffaker B., Fomenkov M. et al (2002). *Distance metrics in the Internet*. Proc. of IEEE International Telecommunications Symposium, IEEE CS Press, Los Alamitos, CA, USA.

[84] Dilley, B. Maggs, et al. (2002). *Globally Distributed Content Delivery*, IEEE Internet Computing, pp. 50-58.

[85] Andrews M., Shepherd, B. et al (2002). *Clustering and Server Selection Using Passive Monitoring*. INFOCOM 2002. Proceedings IEEE Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies., vol.3, pp. 1717- 1725 vol.3, doi: 10.1109/INFCOM.2002.1019425. NY, USA.

[86] Ardaiz O., Freitag, F. et al (2001). *IImproving the Service Time of Web Clients Using Server Redirection*. ACM SIGMETRICS Performance Evaluation Review, Vol. 29, No. 2, ACM Press, pp. 39–44. NY, USA.

[87] Hofmann M., Beaumont, L. R (2005). *Content Networking: Architecture, Protocols, and Practice*. Morgan Kaufmann Publishers, pp. 129–134. San Francisco, CA, USA

[88] Barbir A., Cain, B. et al (2003). *Known Content Network Request-routing Mechanisms*. Internet Engineering Task Force RFC 3568. www.ietf.org/rfc/rfc3568.txt

[89] Mao Z. M., Cranor, C. D. et al (2002). *A Precise and Efficient Evaluation of the Proximity between Web Clients and their Local DNS Servers*. Proc. of the USENIX 2002 Annual Technical Conference, pp. 229–242. CA, USA.

[90] Shaikh A., Tewari, R. et al (2001). *On the effectiveness of DNS-based server selection*. INFOCOM 2001. Proceedings of the IEEE 20th Annual Joint Conference of the IEEE Computer and Communications Societies, vol.3, pp.1801-1810, doi: 10.1109/INFCOM.2001.916678. Anchorage, AK, USA.

[91] Krishnamurthy B., Willis, C. et al (2001). *In the Use and Performance of Content Distribution Network*. Proc. of 1st International Internet Measurement Workshop, ACM Press, pp. 169–182.

[92] Peng, G (2003). *CDN: Content Distribution Network*. Technical Report TR-125, Experimental Computer Systems Lab, Department of Computer Science, State University of New York, StonyBrook, NY. Recuperado Agosto 2015.

http://citeseer.ist.psu.edu/peng03cdn.html

[93] Partridge C., Mendez, T. et al (1993). *Host anycasting service*. Internet Engineering, Task Force RFC 1546. www.ietf.org/rfc/rfc1546.txt

[94] Pathan M., Broberg, J. et al (2007). *An Architecture for Virtual Organization (VO)-Based Effective Peering of Content Delivery Networks*. UPGRADECN' 07. Proc. of the 16th IEEE International Symposium on High Performance Distributed Computing (HPDC), Monterey, California, USA.

[95] Fei Z.M., Bhattacharjee, S. et al (1998). *A novel server selection technique for improving the response time of a replicated service*. INFOCOM '98. Proceedings. IEEE Seventeenth Annual Joint Conference of the IEEE Computer and Communications Societies, vol.2, no., pp.783-791. doi: 10.1109/INFCOM.1998.665101

[96] Milojicic D. S., Kalogeraki, V. et al (2002). *Peer-to-peer computing*. Technical Report, HP Laboratories, Palo Alto, CA, HPL-2002-57.

[97] Stoica I. & Morris, R. et al (2003). *Chord: a scalable peer-to-peer lookup protocol for Internet applications*. IEEE/ACM Transactions on Networking (TON), 11(1), ACM Press, NY, USA, pp. 17–32

[98] Byers J., Considine, J. et al (2003). *Simple load balancing for distributed hash tables*. In Proc. of 2nd International Workshop on Peer-to-Peer Systems (IPTPS'03), pp. 31–35

[99] Berer, K. Hauswirth, M. (2002). *An overview on peer-to-peer information systems*. In Proc. Of the Workshop on Distributed Data and Structures (WDAS), France.

[100] Wang Y., Yun X. et al (2007). *Analyzing the Characteristics of Gnutella Overlays*. Fourth International Conference on Information Technology (ITNG '07), vol., no., pp.1095-1100, doi: 10.1109/ITNG.2007.39

[101] Douglis F., Kaashoek M. F. (2001). *Scalable Internet services*. IEEE Internet Computing, 5(4), pp. 36–37.

[102] Gadde S., Chase, et al. (2000). *Web Caching and Content Distribution: a view from the interior.* In Proc. of the Fifth Int. Web Caching and Content Delivery Workshop.

[103] Krishnamurthy B., Willis, C. et al (2001). *On the use and performance of content distribution networks*. Proc. of 1st International Internet Measurement Workshop, ACM Press, pp. 169–182.

[104] Keynote Systems, http://www.keynote.com/

[105] Vakali A., Pallis G (2003). *Content delivery networks: status and trends*. IEEE Internet Computing, 7(6), IEEE Computer Society, pp. 68–74.

[106] Jenny M., at al (2004). *Patterns: Service-Oriented Architecture and Web Services*. IEEE Internet Computing, 7(6), pp. 30–32 IBM Corp. 2004.

[107] Skillsoft, "*Introducing Cloud Computing*", 2011 - http://store.healthstream.com/product.aspx?zpid=31604 http://www.skillsoft.com/catalog/detail.asp?CourseCode=cl_clbp_a01_it_enus

[108] Torres, O. (2009). "*Evolución y tendencia de la tecnología de streaming en Internet*". Última consulta: 15/12/15 http://exa.unne.edu.ar/depar/areas/informatica/SistemasOperativos/estructura.pdf

[109] CISCO. (2009). "*HTTP versus RTMP: Which Way to Go and Why?*. Última consulta: 15/12/15 http://www.cisco.com/en/US/prod/collateral/video/ps11488/ps11791/ps11802/white_paper_c11-675935.pdf http://www8.cs.umu.se/education/examina/Rapporter/UsmanAfzaal.pdf

[110] Ponce, G. (2013). *Diseño de una Red CDN/P2P Orientada al Live Streaming De Materiales Académicos en el Perú*. Protocolos, codecs y modos para live streaming, Cap. 2. pp. 15-20. Perú.

[111] SyT. Códec (2010). *Códec de video.* Última consulta: 16/12/15 http://www.figge.com.ar/index.htm/ApuntesSyT/C%F3decs%20y%20formatos.pdf

[112] UPNA. (2009). *Servicios en la Web y Distribución de Contenidos*. Video Digital. Última consulta: 16/12/15. https://www.tlm.unavarra.es/~daniel/docencia/doctorado/2008-09/slides/Streaming.pdf

[113] GENETEC (2010). "*Three Simple Ways to Optimize Your Bandwidth Management in Video Surveillance*". Última consulta: 18/07/15. http://www.genetec.com/Documents/EN/Whitepapers/EN-Genetec-Three-Simple-Ways-to-Optimize-Your-Bandwidth-Management-in-Video-Surveillance-WhitePaper.pdf

[114] Level 3 - Over-The-Top Video (2012). Video Delivery. Última consulta: 16/12/15 http://www.level3videocloud.com/story/over-the-top-video

[115] Olivé M. (2008). *Algoritmo de Compresión de Imagenes de Alta Resolución Sin Pérdidas.* La compresión de imágenes, pp. 11–12.

[116] EDII (2011) - *Compresión de la información de vídeo. Redundancia espacial y Temporal* http://edii.uclm.es/~jmlova/Archivos/VD/Archivos/VdCompresion.pdf Última consulta: 16/12/15.

[117] JPEG - *Overview of JPEG* | JPEG 2000 (2010). Core coding system and Motion JPEG 2000 - http://jpeg.org/jpeg2000/

[118] Macaulay A., et al (2005) "IP Streaming of MPEG-4: Native RTP vs MPEG-2 *Transport Stream*". Última consulta: 16/12/15. http://www.envivio.com/files/white-papers/RTPvsTS-v4.pdf https://iq.polskaszerokopasmowa.pl/g2/oryginal/2012_11/3ce0434ebfb2e6dde5bc616a84cfa839.pdf

[119] Halpert B. (2011). Auditing Cloud Computing - A Security and Privacy Guide. Introduction to Cloud Computing, Hoboken, New Jersey, pp. 1–15.

[120] Parkhill, D. (1996). *The Challenge of the Computer Utility*. Addison-Wesley Educational Publishers Inc., US.

[121] Gorelik E. (2013). *Cloud Computing Models*. Comparison of Cloud Computing Service and Deployment Models, CISL# 2013-01, Massachusetts, pp. 7–20.

[122] Mell P., Grance T. (2011). *The NIST Definition of Cloud Computing*. NIST Special Publication 800-145.

[123] Rosales E. (2010). *UNACLOUD: Infraestructura como Servicio para Cloud Computing Oportunista*. Ventajas y desventajas de Cloud Computing. pp 26-28.

[124] Prabhakar C. (2008). *Cloud computing with Amazon Web Services*. consulta: 15/12/15.

http://www.ibm.com/developerworks/websphere/techjournal/1011_amato/1011_amato.html

[125] Hamilton, J. (2008). *Internet-Scale Service Efficiency*. s.l: In Large-Scale Distributed Systems and Middleware (LADIS) Workshop.

[126] VMware, (2015. Inc. *Transform your Business with Virtualization.*
http://www.vmware.com/virtualization/what-is-virtualization.html

[128] Franco F. (2009). *Consolidación de servidores con vmware esx, en el centro de datos de la dirección de cómputo y comunicaciones*. ¿Qué es la Virtualización? - pp.16

[129] ORSI, Junta de castilla y León. (2010). *Cloud Computing. Observatorio Regional de la Sociedad de la Información.* La tecnología como servicio. Recuperado noviembre 2015.
http://issuu.com/orsicyl/docs/cloud_computing?mode=a_p

[130] Jinesh V. (2010). *Architecting for the Cloud*: Best Practices

[131] Nerion. (2012). *El blog de Virtualizamos*. Recuperado octubre 2015
http://blog.virtualizamos.es/2011/09/21/%C2%BFque-tecnologia-de-hypervisor-de-virtualizacion-elegir/

[132] Campos D. y Zamorategui E., (2012). *Laboratorio de prácticas, modelado en tecnologías libres*. Capítulo 2: Fundamentos teóricos, pp. 17-21

[133] Figueroa C., (2013). *Analisis, Comparación e Implementación de una infraestructura virtual open source, con alta disponibilidad basada en Clusters*. Capítulo 1: Técnicas de Virtualización, pp. 6-8

[134] Fernandez, M. (2009). *El Blog de Marcelo*! Recuperado octubre 2015
http://blog.marcelofernandez.info

[135] Utterback, James M., (1994) "*Mastering the Dynamics of Innovation*", Harvard Press.

[136] Amazon EC2. (2014). Cloud is made up of almost half-a-million Linux servers
http://www.zdnet.com/blog/open-source/amazon-ec2-cloud-is-made-up-of-almost-half-a-million-linux-servers/10620

[137] Skillsoft, "*Introducing Cloud Computing*", 2011
http://store.healthstream.com/product.aspx?zpid=31604

http://www.skillsoft.com/catalog/detail.asp?CourseCode=cl_clbp_a01_it_enus

[138] Christodoulou G, Georgiou C, Pallis G. (2012). *The role of twitter in youtube videos diffusion*. Proceedings of the 13th International Conference on Web Information System Engineering (WISE); Paphos, Cyprus.

[139] Welcome to the Cloud Services, (2015). *Cloud Computing, Demystifying Cloud Myths.* Capítulo 1-5, pp. 5-30. / Toll Free: 18002122022. Website: www.cloudoye.com

[140] Hernansanz J., Requejo J. y Mirón J. (2009). *Arquitecturas de red para servicios en Cloud Computing*. Capítulo 4: Principales caracteristicas, pp. 18-19.

[141] Melendez M., (2012). *Diseño de Metodología para la Gestión de la Seguridad de la información en el Cloud Computing para Universidades Públicas*. Capítulo 4: Modelos de implementación en el cloud computing, pp. 9-11

[142] J. D. Lasica. (2009). *Identity in the Age of Cloud Computing*. Recuperado mayo 2015 http://www.aspeninstitute.org/sites/default/files/content/docs/pubs/Identity_in_the_Age_of_Cloud_Computing.pdf

[143] Ullauri G., (2013). *Diseño e Implementación de una Red de Servicios Basada en los conceptos de Cloud Computing*. Capítulo 4: Modelos de implementación en el cloud computing, pp. 15-27

[144] Revelo M., (2013). *Servicio de virtualización de infraestructura tecnológica basado en cloud computing*. Capítulo 2: Fundamento Teórico, pp. 9-25

[145] Cisco, (2014). *Private Cloud Computing For Enterprises*. Recuperado en Junio 12 de 2015 de http://www.cisco.com/c/en/us/solutions/collateral/data-center-virtualization/cloud-computing/white_paper_c11-543729.html

[146] Salazar R., (2013). *Análisis Comparativo de Tecnologías de Infraestructura como Servicio en Cloud Computing y su Aplicación de un Modelo para ea EIS*. Capítulo 2: Tipos de Cloud Computing, pp. 41-46

[147] Figueroa C., (2013). *Análisis, Comparación e implementación de una infraestructura virtual open source con alta disponiblidad basada en Clusters*. Capítulo 2: Fundamento Teórico, pp. 9-15

[148] Kamaraju A. & Nicolas P. (2009). *Cloud Storage*. Recuperado Diciembre 2015

http://www.snia.org/sites/default/education/tutorials/2009/spring/applications/AshvinKamaraju_Cloud_Storage_Introductionsv16.pdf

[149] Mendoza A., (2007). *Utility Computing Technologies*. Recuperado en Diciembre 10 de 2015 de Standards, and Strategies, EEUU: Artech House, Inc.

[150] Leung A., Pasupathy S., Goodson G., Miller E. (2008). *Measurement And Analysis Of Largescale Network File System Workloads*. Recuperado en Diciembre 10 de 2015 de http://www.ssrc.ucsc.edu/Papers/leung-usenix08.pdf

[151] Computing, (2016). *La nube híbrida domina las estrategias de negocio* http://www.computing.es/informatica-profesional/noticias/1085873001701/nube-hibrida-domina-estrategias-negocio.1.html

[152] Eludiora S., at al. (2011). *A User Identity Management Protocol for Cloud Computing Paradigm.* Capítulo 2: Security Issues of Cloud Computing, pp. 156-160. EEUU: Houston, USA.

[153] So S. (2009). *Cloud Computing and Information Security.* Info-Security Project

[154] Cox, P. (2009). *How to Use Software as a Service Securely - SaaS*. Search Cloud Computing.com. Recuperado diciembre 15, 2015.

http://searchcloudcomputing.techtarget.com/tip/Mitigating-the-top-three-Software-as-a-Service-security-concerns

[155] Cox, P. (2009). *How to Use Platform as a Service Securely - PaaS*. SearchCloud Computing.com. Recuperado diciembre 15, 2015.

http://searchcloudcomputing.techtarget.com/tip/Top-threats-in-a-PaaS-cloud-service-and-how-to-avoid-them

[156] Cox, P. (2009). *Securing IaaS Operating Systems Vulnerabilities.* SearchCloud Computing.com. (recuperado noviembre 2015).

http://searchcloudcomputing.techtarget.com/tip/The-most-important-threat-facing-an-IaaS-cloud-service

[157] Quan D. (2008). *From Cloud Computing To The New Enterprise Data Center*.

http://www-958.ibm.com/software/data/cognos/manyeyes/datasets/from-cloud-computing-tothe-new-ente/versions/1.txt Recuperado diciembre 15, 2015

[158] Xin Y., et al, (2011). *Embedding Virtual Topologies in Networked Clouds.* Proc. Sixth Int'l Conf. Future Internet Technologies (CFI '11), pp. 26-29.

[159] Pathan A-M.K., Buyya R., y Vakali A., (2008). *Content Delivery Networks: State of the Art, Insights, and Imperatives.* Content Delivery Networks, vol. 9, no. 1, pp. 3-32.

[160] Leivadeas A., Pappagianni C, & Papavassiliou S. (2012). *Efficient Resource Mapping Framework over Networked Clouds via Iterated Local Search Based Request Partitioning*," published in IEEE Trans. Parallel and Distributed Systems.

[161] Ericsson (2012). *The Telecom Cloud. Opportunity Operator Opportunities in Cloud Service Delivery*. White paper, Ericsson. Recuperado enero 2016.
http://www.ericsson.com/res/site_AU/docs/2012/ericsson_telecom_cloud_discussion_paper.pdf

[162] Peng G. (2008). *CDN: Content distribution network*. Stony Brook University, Technical Report TR-125.

[163] Armbrust M, et al. (2009). *Above the clouds: A Berkeley view of cloud computing.* University of California at Berkley, USA, Technical Report No. UCB/EECS-2009-28.

[164] Adhikari VK et al. (2012). *Unreeling Netflix: Understanding and improving multi-CDN movie delivery*. INFOCOM. p 1620–1628.

[165] NETFLIX (2014). *OpenConnect Appliance deployment guide*. Netflix. Recuperado enero 2016. http://oc.nflxvideo.net/docs/OpenConnect-Deployment-Guide.pdf

[166]. Cisco. 2013. *Cisco visual networking index forecast*. Ultimo ingreso diciembre 2015 http://www.cisco.com/go/vni/

[167] Chen F., et al, (Mar. 2012) "*Intra-Cloud Lightning: Building CDNs in the Cloud*," Proc. IEEE INFOCOM, pp. 433-441.

[168] Chia-Feng L., et al, (2011). *The Study and Methods for Cloud based CDN*. International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery. Integrated CDN and Cloud Storage, pag 470 - 473.

[169] Papagianni C., et al, (2012) *On the Optimal Allocation of Virtual Resources in Cloud Computing Networks*. to be published in IEEE Trans. Computers.

[170] Hernansanz J., Requejo J. y Mirón J., (2009). *Arquitecturas de red para servicios en Cloud Computing*. Capítulo 10: Balanceador de carga, pp. 55-62

[171] CDNetworks (2015). *Cloud Load Balancer. DNS-based Global Load Balancing*, https://www.cdnetworks.com/wp-content/uploads/2016/01/CDNetworks-CloudLoadBalancer-BR-EN-US1.pdf

[172] Broberg J., Buyya R. & Tari Z. (2009). *MetaCDN: Harnessing 'storage clouds' for high performance content delivery*. J Netw Comput Appl; 32: 1012–1022.

[173] Srinivasan S., et al (2011). *ActiveCDN: Cloud computing meets content delivery networks*. Columbia University; Computer Science Technical Reports.

[174] Amazon CloudFront (2016). http://aws.amazon.com/cloudfront/

[175] Akamai NetStorage (2016).
http://www.akamai.com/html/technology/products/netstorage.html

[176] Gorelik E. (Enero 2013). C*loud Computing Models. Comparison of Cloud Computing Service and Deployment Models*, Open-Source vs. Proprietary Cloud Technologies. CISL# 2013-01, Massachusetts, pp. 75.

[177] Pathan M., Sitaraman & Robinson (2014). *Advanced Content Delivery, Streaming, and Cloud Services.* Cloud-Based CDN, Capitulo 12, pp. 246-247. United States.

[178] Islam S & Grégoire JC. (2012). *Giving users an edge: A flexible cloud model and its application for multimedia.* Future Gener Comp Syst; 28(6):823–832.

[179] Yuedui W., et al. (2011). *The Content Delivery Network System based on Cloud Storage*. International Conference on Network Computing and Information Security. pp 98-100.

[180] Ling L., Xiaozhen M. & Yulan H. (2013). *CDN Cloud: A Novel Scheme for Combining CDN and Cloud Computing*. 2nd International Conference on Measurement, Information and Control. pp. 688-690.

[181] Papagianni C., Leivadeas A. y Papavassiliou S. (Trad). (2013). *A Cloud-Oriented Content Delivery Network Paradigm: Modeling and Assessment*. IEEE Transactions On Dependable And Secure Computing, VOL. 10, NO. 5, pp. 288-291.

[182] Level 3 (2015). *Content Delivery Network Overview*. Cluster Architecture. Server Architecture https://mediaportal.level3.com/webhelp/help/Content/ContentDeliveryNetworkOverview.htm

[183] Level 3. Media Portal (2015). *CDN-based functions*. Physical Site y Logical Clusters https://mediaportal.level3.com/webhelp/help/Content/ServicesDocs-Caching/CachingServiceDescription/Architecture.htm

[184] Peak10 (2015). Level 3 - *cloud connect solutions*. Recuperado en Enero 16 de 2015 www.peak10.com/wp-content/uploads/2015/03/marketing-sheet-level-3-cloud-connect-solutions-ethernet.pdf

[185] Microsoft (2015) "*What is Windows Azure?*", Recuperado 3 Enero 2016 del sitio web de Microsoft: http://www.microsoft.com/windowsazure/

[186] Microsoft (2015) "*Windows Azure Content Delivery Network*", Recuperado 3 Enero 2016: https://azure.microsoft.com/en-us/blog/introducing-the-windows-azure-content-delivery-network/

[187] Adhikari VK. et al. (2012). *Unreeling Netflix: Understanding and improving multi-CDN movie delivery*. INFOCOM; 2012. p 1620–1628.

[188] Tran H., Mellouk A. y Hoceint S. (2011). *QoE Content Distribution Network for Cloud Architecture*. 1st International Symposium on Network Cloud Computing and Applications, pp. 14-19.

ANEXOS

Nodo CDN - Content Distributor Servers

Appliance front view

Options	
Rack Size	2 Post rack rails (aka Telco cabinet - regular 19" wide) OR 4 Post rack rails (aka Server cabinet - regular 19" wide)
AC or DC Power Supply (Peak power draw: 560 watts) **Note:** Power Supplies are not hot swappable.	**AC Power supplies (2N redundant):** • C14 connector • Voltage: 180 - 240V • Input current 5-3.8A • Frequency: 60-50Hz • Included in package: 2x C13 to C14 and 2x C13 to NEMA 5-15 *OR* **DC Power supplies (2N redundant)** 10A fuse per power supply recommended.
Network	SFP+ 10GE, LC Connector • 10GBASE-SR Multimode LC Connector OR • 10GBASE-LR Monomode LC Connector **Included in package**: Fiber patch cables (upon request), provide length and connector type on the router side

Appliance back view

Physical Specifications	
Appliance Weight	31lbs (14Kg)
Shipping Weight	Appliance: 31lbs (14Kg) Rails: 3lbs (1.4Kg) Packaging: 8lbs (3.6Kg) Total: 42lbs (19Kg)
Dimensions	1.7"x19"x23.5" (4.32cm x 48.26cm x 59.7cm)
Power Draw (peak)	165W
Operational Throughput	Quad-port 10GBE NIC 13-17Gb/s using 2 10GBE ports in LACP bundle
Storage Capacity	14TB MLC SSD
Rack space	1 Rack unit (1U)

Load Balancer – Ej. Nginx (engine x)

Instalación:

```
ubuntu@mv01:~$ wget http://sysoev.ru/nginx/nginx-0.7.59.tar.gz
ubuntu@mv01:~$ tar xvfz nginx-0.7.59.tar.gz
ubuntu@mv01:~$./configure
ubuntu@mv01:~$ make
ubuntu@mv01:~$ sudo make install
```

También lo podríamos hacer de forma automática con:

```
ubuntu@mv01:~$ apt-get install nginx
```

En su modalidad de balanceador de carga, que es la que se podría utilizar en este caso, su configuración es bastante sencilla, quedando el nginx.conf de esta forma:

```
user www-data www-data;
worker_processes 2;
error_log logs/error.log debug;
pid logs/nginx.pid;
events {
worker_connections 10240;

}
http {
# include conf/mime.types;
default_type application/octet-stream;
log_format main '$remote_addr - $remote_user [$time_local]
$status '
'"$request" $body_bytes_sent "$http_referer" '
'"$http_user_agent" "http_x_forwarded_for"';
access_log logs/access.log main;
sendfile on;
tcp_nopush on;
tcp_nodelay on;
upstream backends{
server 10.1.2.10:80;
server 10.1.2.11:80;
server 10.1.1.12:80;
server 10.1.1.13:80;
}
server {
listen 80;
access_log logs/host.access.log main;
location / {
proxy_pass http://backends;
#proxy_pass http://10.1.1.69:80;
proxy_redirect off;
proxy_set_header Host $host;
proxy_set_header X-Real-IP $remote_addr;
proxy_set_header X-Forwarded-For
$proxy_add_x_forwarded_for;
client_max_body_size 10m;
client_body_buffer_size 128k;

proxy_connect_timeout 90;
proxy_send_timeout 90;
proxy_read_timeout 90;
proxy_buffer_size 4k;
proxy_buffers 4 32k;
proxy_busy_buffers_size 64k;
proxy_temp_file_write_size 64k;
}
}
}
```

Por último, para iniciar el Nginx, se debe ejecutar:

```
ubuntu@mv01:~$ /etc/init.d/nginx start
```

CDN LINUX KERNEL

Amazon EC2 Kernels

El servicio de Amazon AWS tiene anunció que ahora los clientes pueden ejecutar los kernels de usuario proporcionados en EC2. Con esta característica, Amazon permite cargar un para-virtual Linux kernel dentro de una Amazon Machine Image (AMI) o volumen de Amazon EBS. Por lo tanto, también tiene la opción de crear imágenes que contienen un kernel y el initrd, el cual se comporta lo más cercano a las instalaciones físicos o virtuales tradicionales de Linux (Amazon, 2016).

Hay una serie de distribuciones de Linux que tienen kernels EC2 compatibles:

- SLES/openSUSE 10.x, 11.0, 11.1 Xen
- SLES/openSUSE 11.x EC2 Variant
- Ubuntu EC2 Variant kernels
- RHEL 5.x kernels
- CentOS 5.x kernels

Para iniciar una instalación, tendrá que realizar los siguientes pasos de alto nivel:

1. Install an EC2 compatible kernel.
2. Generate an initrd.
3. Populate /boot/grub/menu.lst referencing your kernel.
4. Select an appropriate AKI ID from the “Amazon Kernel Image IDs” section below.
5. For new AMIs or Amazon EBS volumes, bundle the AMI and set the default to your chosen AKI.
6. Upload and register your new AMI.
7. For existing AMIs, you can simply specify the appropriate AKI ID when you call RunInstances or from within the AWS Management console.

1. Install an EC2 compatible kernel from the command line on your running Linux instance.

```
# rpm –ivh /tmp/kernel-ec2-2.6.35-rc4.8.1.x86_64.rpm
warning: /tmp/kernel-ec2-2.6.35-rc4.8.1.x86_64.rpm: Header V3 DSA
signature:
NOKEY, key ID a29f6635
Preparing... ########################################
[100%]
1:kernel-ec2 ########################################
[100%]
Kernel image: /boot/vmlinuz-2.6.35-rc4-8-ec2
Initrd image: /boot/initrd-2.6.35-rc4-8-ec2
Root device: /dev/sda1 (mounted on / as ext3)
Features: block
```

```
14807 blocks
```

2. Generate an initrd on your running Linux instance.

```
# mkinitrd
Kernel image: /boot/vmlinuz-2.6.35-rc4-8-ec2
Initrd image: /boot/initrd-2.6.35-rc4-8-ec2
Root device: /dev/sda1 (mounted on / as ext3)
Features: block
14806 blocks
```

3. Populate /boot/grub/menu.lst referencing your kernel on your running Linux instance.

```
default 0
timeout 3
title EC2
root (hd0)
kernel /boot/vmlinuz-ec2 root=/dev/sda1
initrd /boot/initrd-ec2
```

4. Select an appropriate AKI ID from the “Amazon Kernel Image IDs” section below.

For this host, “aki-427d952b” is being chosen, because we are bundling an AMI.

5. For new AMIs or Amazon EBS volumes, bundle the AMI and set the default to your chosen AKI from your running Linux instance.

```
# ec2-bundle-vol -r x86_64 -d /mnt -p openSUSE-11.2-pvGrub -u
[AWS-ID] -k
/mnt/pkey.pem -c /mnt/cert.pem -s 10240 -e /mnt,/root/.ssh --
kernel aki-
427d952b
```

6. Upload the bundle from your running Linux instance to S3.

```
# ec2-upload-bundle -b MyReallyCoolBucketLocation -m
/mnt/oepnSUSE-11.2-
pvGrub.manifest.xml -a MyAccessKey -s MySecretKey
```

7. Register the AMI with the AKI chosen in step 4 from your desktop using the EC2 command line tools.

```
$ ec2-register --name openSUSE-11.2-pvGrub
MyReallyCoolBucketLocation/oepnSUSE-11.2-pvGrub.manifest.xml
```

Otros - Linux Kernel CDN

Production Kernels

KERNEL VERSION	RELEASE NOTES	STATUS
2.6.35.10-18ab	• TCP optimization (Google PRR & Increased Rcv window). • Raid 10 support.	• Eng. test on 20348.x (IL, General).
2.6.35.10-16ab	• PMTU Bug fix for IPv6.	• FP5.x
~~2.6.35.10-14ab~~	• ~~IPv6 support~~ • ~~Stability bug fixes.~~	• ~~In trial on FP5.x.~~ • Deprecated in favor 2.6.35.10-16ab
2.6.35.10-10ab	• TCP Optimization • Increaed ethernet ring buffer sizes (4096). • Throughput improvements for Netflix.	• GA for Netflix • In trial for other bindings.
2.6.28.3-18ab	• TCP Optimization	• GA for all bindings except: BLUE (based on direction from Product).
2.6.28.3-11ab	• Stability bug fixes.	• ~~GA for all bindings~~ • Deprecated for all bindings except Blue.

Release Candidate Kernels

KERNEL VERSION	FEATURE LIST	STATUS
2.6.35.10-16ab	• LX00196. PMTU Bug fix for IPv6.	• IN FIELD TRIAL

New Feature Requests

FEATURE ID	DESCRIPTION	REQUESTOR	REVIEWED BY	APPROVED
---	• [Description of the feature and why it is needed]	• [Who]	[Eng, Dev]	[Y/N reason if rejected.]
	RAID10 support: • Wanted for PCI 5.8/9 images • Wanted for DEC 5.8/9 images • Wanted for CA-DB 5.8/9 images • Wanted for ITM 5.8/9 images INITRD support: • Wanted to support moving forward to CentOS 5.8/9 in a standard fashion • Wanted to facilitate futher modularization of the kernel SCSI driver support: • Needed for current vendor 4U • Needed for alternate vendor4U (Dell) • Needed for alternate vendor 2U (Dell) Attached for review:	pmcdonne	pm,wp	

FEATURE ID	DESCRIPTION	REQUESTOR	REVIEWED BY	APPROVED
	kernel.config.x86_64 kernel.spec			
	Patch to igb driver to support new Dell platform. At present, the 4 igb card will come up with a single MAC address for all 4 ports. There is an existing work around for the issue, but the work around adds complexity at installation time. http://patchwork.ozlabs.org/patch/120978/ From: Carolyn Wyborny <carolyn.wyborny@intel.com> In 82580 and later devices, the alternate MAC address feature is completely handled by the option ROM and software does not handle it anymore. This patch changes the check_alt_mac_addr function to exit immediately if device is 82580 or later.	cyoder		
	Device-mapper support: CONFIG_BLK_DEV_DM=y CONFIG_DM_DEBUG=y			

FEATURE ID	DESCRIPTION	REQUESTOR	REVIEWED BY	APPROVED
	CONFIG_DM_BUFIO=m CONFIG_DM_PERSISTENT_DATA=m CONFIG_DM_CRYPT=m CONFIG_DM_SNAPSHOT=m CONFIG_DM_THIN_PROVISIONING=m CONFIG_DM_MIRROR=m CONFIG_DM_RAID=m CONFIG_DM_LOG_USERSPACE=m CONFIG_DM_ZERO=m CONFIG_DM_MULTIPATH=m CONFIG_DM_MULTIPATH_QL=m CONFIG_DM_MULTIPATH_ST=m CONFIG_DM_DELAY=m CONFIG_DM_RAID45=m CONFIG_DM_UEVENT=y CONFIG_DM_FLAKEY=m			

DISEÑO DE UN NODO CON TECNOLOGÍA CLOUD-ORIENTED CONTENT DELIVERY NETWORK, UTILIZANDO LA INTEGRACIÓN CON MODELOS DE SERVICIO DE CLOUD COMPUTING, PARA EL DATACENTER DE UN PROVEEDOR DE SERVICIOS DE CONTENIDO.

Santiago Gálvez Huerta, Quito - Ecuador, santy20sn@gmail.com, 0984493595