IMPLEMENTACIÓN DE UNA PLATAFORMA DE JUEGO EN LA NUBE EN UNA RED DEFINIDA POR SOFTWARE CONFIGURADA EN GENI

FRANNIER DANIEL TREJOS MARTÍNEZ

UNIVERSIDAD CATÓLICA DE PEREIRA

FACULTAD DE CIENCIÁS BÁSICAS E INGENIERÍA

RISARALDA

PEREIRA

2016

IMPLEMENTACIÓN DE UNA PLATAFORMA DE JUEGO EN LA NUBE EN UNA RED DEFINIDA POR SOFTWARE CONFIGURADA EN GENI

FRANNIER DANIEL TREJOS MARTÍNEZ

Trabajo de Grado presentado para optar al título de Ingeniero de Sistemas y Telecomunicaciones

UNIVERSIDAD CATÓLICA DE PEREIRA

FACULTAD DE CIENCIÁS BÁSICAS E INGENIERÍA

RISARALDA

PEREIRA

2016

CONTENIDO

LISTA DE ILUSTRACIONES

LISTA DE TABLAS

RESUMEN

Los videojuegos se han convertido en un mercado que se ha expandido con rapidez, pero para poder ejecutar los videojuegos actuales se requiere de un computador o consola de última generación, donde el primero aunque tiene un mejor desempeño gráfico requiere de actualizaciones de hardware que conllevan a costos mayores. Lo anterior puede evitarse con la implementación de nubes basadas en videojuegos, donde los servidores brindan el procesamiento central y gráfico de aplicativos de altos requerimientos gráficos, permitiendo que todo usuario con conexión a Internet pueda ejecutar en cualquier tipo de dispositivo videojuegos exigentes gráficamente de forma remota, sin la necesidad de preocuparse por los requisitos de hardware de éstos, este paradigma se conoce como juego en la nube. En este paradigma existen diversas plataformas y servicios, en su mayoría soluciones privadas, creadas para minimizar la latencia que experimenta el usuario final, aunque con poca documentación de ellos.

Este proyecto de grado presenta un estado del arte del juego en la nube con el fin de aportar a la documentación de esta tecnología, así como la implementación de una plataforma de este tipo en una red definida por software configurada en GENI.

Palabras clave—Juego en la nube, juego como servicio, SDN, GENI, GaaS, juego remoto.

ABSTRACT

Video games have become a market that has expanded rapidly, but to run the current games it requires a computer or console of last generation, where the first has better graphics performance but requires hardware upgrades that lead to higher costs. This can be avoided with the implementation of cloud-based video games, where the servers provide the central and graphics processing for applications with high graphics requirements, allowing that any user with an Internet connection can execute in any type of device, graphically-demanding games remotely without the need to worry about the hardware requirements of these, this paradigm is known as cloud gaming. In this paradigm there are several platforms and services, mostly private solutions, created to minimize latency experienced by the end user, but with little documentation of them.

This degree project presents a survey about the cloud gaming, in order to contribute to the documentation of this technology, and the implementation of a cloud gaming platform in a software defined network configured in GENI.

Descriptors—Cloud gaming, game as a service, SDN, GENI, GaaS, remote gaming.

1. INTRODUCCIÓN

Como resultado de los recientes avances en Computación en la Nube (Cloud Computing) y en los centros de datos, se ha producido un aumento significativo en la demanda de servicios que estos pueden ofrecer, ya que la centralización de los recursos y los procesos de cálculo de los centros de datos crean soluciones a los usuarios de bajo costo y flexibles para una gran cantidad de aplicaciones. El cloud gaming (juego en la nube) es una de las aplicaciones que pueden beneficiarse de los avances de la computación en la nube, permitiendo ofrecer un juego como servicio (Game as Service), donde el servidor se encargará de todas las tareas de procesamiento de un videojuego y vía streaming el cliente podrá acceder a él. Un motivo por el cual se creó esencialmente el Juego en la Nube fue porque debía existir una solución en la cual se pudiera virtualizar altos requerimientos gráficos con la mayor fluidez posible, en entornos donde no sólo importaba tener el menor retardo posible sino el procesamiento gráfico.

Pero el juego en la nube tiene poca investigación debido a su uso primordialmente privado, además hay que tener en cuenta que una plataforma de Cloud Gaming requiere fluidez, retardo y latencia mínima, compresión de audio y video, optimización según la conexión de red del usuario y sobre todo una buena experiencia para el usuario; los cuales son problemas difíciles de solucionar debido a la escasa documentación sobre el tema.

Existe una solución que podría ayudar significativamente a los problemas anteriores y esa es la implementación de las redes definidas por software (SDN) en una plataforma de juego en la nube. Las SDN son una manera de abordar la creación de redes en la cual el control se desprende del hardware y se le da el mismo a una aplicación de software llamada controlador, se han venido trabajando en los últimos años en las mayores plataformas del mundo como Amazon, Google, Microsoft, entre otras. Hacen referencia a una arquitectura de red que permite separar el plano del control, del plano de datos, para conseguir redes más programables, automatizables y flexibles.

Gracias a los beneficios que traen las SDN, se implementará una plataforma de juego en la nube con esta tecnología, haciendo dos contribuciones: En primer lugar, la aplicación de las SDN en un sistema de juego en la nube para la disminución de retardos, en segundo lugar un avance en la investigación y documentación de las diferentes plataformas de juego en la nube y controladores SDN que pueden usarse en el servicio.

2. ÁREA PROBLEMÁTICA

Los videojuegos se han convertido en un mercado que se ha expandido con rapidez, viniendo desde consolas especializadas hasta computadores de alta tecnología. Pero para poder ejecutar los videojuegos actuales, los usuarios precisan de un computador o consolas de última generación, donde esta última no tiene el mismo desempeño gráfico de un computador, sin embargo este requiere de actualizaciones de hardware y una mayor inversión, haciendo que el usuario se incline por la adquisición de una consola sacrificando de esta manera gráficos, velocidad, accesibilidad y comodidad. También hay que tener en cuenta que la conexión a Internet se ha vuelto más asequible con el paso de los años, por lo cual debe haber una alternativa que permita a los usuarios ejecutar cualquier videojuego desde cualquier dispositivo con conexión a Internet, como lo es el Juego en la Nube.

Adicionalmente, en la infraestructura para proveer el servicio de streaming de los videojuegos y aplicaciones en el sistema de Juego en la Nube, los servidores consumen mucha energía y requieren de un buen hardware para soportar los juegos y transmitirlos con rapidez. Así que al conectar los diferentes equipos que proveerán el servicio, se debe pensar en una solución para reducir el consumo eléctrico, aumentar la velocidad de conexión con el cliente para que no hayan retardos, la latencia sea mínima, y que la plataforma que proveerá el servicio de cloud gaming tenga estabilidad en el streaming para que el videojuego sea “jugable” tanto offline como online.

También hay que tener en cuenta que en la última década se han implementado diferentes aplicaciones de las redes definidas por software, dando así una inteligencia a la red para diferentes propósitos, entre ellos la computación en la nube. Las SDN además permiten dar un control algorítmico de la red de elementos de red (switches/routers) facilitando su configuración y gestión, su reutilización y el despliegue de aplicaciones, servicios e infraestructuras en el menor tiempo posible.

Podemos concluir que el usuario requiere de una alternativa para poder acceder a aplicativos de altos requerimientos gráficos por medio de una conexión a Internet, y existe un modelo de computación en la nube denominado Juego en la Nube diseñado para este propósito. Pero este modelo tiene poca documentación, además que para su implementación requiere de una buena gestión de recursos y de conexión para que los juegos o aplicativos tengan la mayor fluidez posible, y el usuario tenga una buena experiencia del servicio. Las SDN pueden ser una solución a este problema.

4. OBJETO DE ESTUDIO

El objeto de estudio de este proyecto es la computación en la nube aplicada en la ejecución de videojuegos y la implementación de redes definidas por software para este tipo de servicio.

5. OBJETIVOS

5.1. OBJETIVO GENERAL

Implementar una plataforma de Juego en la Nube en una red definida por software configurada en GENI.

5.2. OBJETIVOS ESPECÍFICOS

- Realizar una búsqueda bibliográfica para seleccionar una plataforma que provea el servicio de Juego en la Nube.
- Seleccionar el protocolo y controlador que se van a configurar en la SDN para dar soporte a la plataforma de Juego en la Nube.
- Diseñar y configurar una infraestructura SDN en GENI para proveer el servicio de Juego en la Nube,
- Evaluar el desempeño de la implementación de la plataforma.

6. DELIMITACIÓN Y/O ALCANCES DEL PROYECTO

Este proyecto se centrará en la implementación de una plataforma funcional de cloud gaming con redes definidas por software, incluyendo lo siguiente:

- Un estudio previo de la tecnología del juego en la nube.
- La exploración de soluciones propuestas por diferentes autores para lograr mejoras en un sistema de cloud gaming.
- La investigación y selección de una plataforma de cloud gaming y un protocolo de redes definidas por software para usarlos en la infraestructura del servicio.
- La implementación del servicio en una nube privada para al menos dos usuarios simultáneos.
- La optimización de la plataforma con técnicas para mejorar su desempeño.

En caso de tener problemas de conectividad con los diferentes equipos que servirán de servidores, se implementará el servicio con máquinas virtuales o los equipos del laboratorio de telecomunicaciones de la Universidad Católica de Pereira.

7. APORTE PRÁCTICO

Con este proyecto se beneficiará la industria de los videojuegos y la virtualización, ya que se usará el modelo del cloud computing para la virtualización de videojuegos y transmitirlos en la nube. Por lo cual cualquier usuario que quiera acceder al videojuego o aplicativo sólo necesitaría de un dispositivo con conexión a Internet para ejecutarlo, sin preocuparse por las especificaciones técnicas de éste. Además que los productos de la investigación, servirán para otro tipo de implementaciones, como en la aplicación de redes definidas por software para servicios similares en los que se requiera de una mejor gestión de recursos o de consumo eléctrico, la ejecución de software de altos requerimientos gráficos a través de la nube, entre otras.

8. APORTE TEÓRICO

Adelantar la exploración, utilización y documentación del juego en la nube; avanzar en la investigación de las redes definidas por software aplicadas en un sistema de juego en la nube.

9. FACTIBILIDAD

9.1. FACTIBILIDAD TÉCNICA

La implementación de un sistema de “juego en la nube” es viable técnicamente, ya que se cumplen con las tecnologías o conocimientos, que se muestran a continuación:

- Una plataforma que sea compatible con el servicio de “cloud gaming”. En la actualidad existen diversas plataformas que pueden usarse para dicho fin que no existían en la última década, entre ellas GamingAnywhere, SteamMyGame, In-Home Streaming y NVIDIA GameStream.
- Una tecnología para darle una inteligencia a los diferentes servicios de una red, de una manera dinámica para mejorar la fluidez de la plataforma, la conectividad, la reducción del consumo eléctrico u otro aspecto. Para lo anterior existen las redes definidas por software, las cuales son un concepto bastante reciente pero con muchas posibilidades, ya que separan el plano de control del plano de datos y se pueden instalar en el servidor que proveerá el “juego en la nube”.
- Equipos que tengan alto procesamiento gráfico y dispositivos que sean compatibles con las redes definidas por software. A la fecha, existen switches y dispositivos compatibles con las SDN, asimismo las tarjetas de aceleración gráfica y el procesamiento en general de los computadores ha mejorado exponencialmente en los últimos años, por lo cual pueden servir para la infraestructura del servicio.

9.2. FACTIBILIDAD ECONÓMICA

Para brindar el servicio de juego en la nube, se requiere de equipos de última generación porque estos tienen un gran procesamiento gráfico y pueden soportar varios usuarios, pero ya que se hará un prototipo del sistema para soportar al menos dos usuarios, se necesitan aproximadamente tres equipos de este tipo con buena conexión a Internet. También se requieren uno o dos dispositivos compatibles con redes definidas por software para la conexión de los diferentes equipos que harán el papel de servidores.

El laboratorio de Telecomunicaciones de la Universidad Católica de Pereira, tiene equipos de gama alta, switches compatibles con OpenFlow (protocolo de redes definidas por software) y una conexión a Internet rápida para implementar el servicio, por lo cual no se cubriría el gasto de ellos en la investigación.

Además para el proyecto se tiene acceso al laboratorio virtual de redes denominado GENI (Global Environment for Network Innovations), en el cual se podrá implementar el servicio de forma gratuita.

De igual manera para la plataforma de juego en la nube, se seleccionará una libre y gratuita por lo cual no se tendrán problemas económicos o de licenciamiento.

10. MARCO TEÓRICO

10.1. ANTECEDENTES

Los primeros indicios de investigación del juego en la nube datan del año 2000 por la empresa finlandesa G-Cluster [1] en la “Electronic Entertainment Expo” (Exposición de Entretenimiento Electrónico) mostrando un prototipo de esta tecnología, y no fue hasta el año 2005 cuando se implementó por primera vez un servicio comercial de juego en la nube para “Cyprus Telecommunications Authority” en una red IPTV [2], por lo cual la aplicación de esta tecnología es reciente. En marzo de 2008 G-cluster anunció la inclusión de soporte para alta definición en su solución con Amino STB IPTV. En noviembre de 2010, después de pruebas a fondo, el operador francés SFR lanzó un servicio comercial basado en la tecnología G-cluster [3]. El 11 de octubre de 2012 Orange lanza un servicio de juego bajo demanda utilizando la tecnología G-Cluster.

A parte de G-Cluster, se crearon otras empresas de juego en la nube de uso exclusivamente comercial o privado, como se mostrará a continuación:

En el 2008 la empresa denominada Gaikai [4], transmitió demos de videojuegos que se podían acceder por un navegador, con la posibilidad de acceder al videojuego completo pagando una cantidad de dinero.

En el año 2009 el juego en la nube fue difundido por diferentes compañías de videojuegos bajo demanda como Onlive, Gaikai y Playcast en la “Game Developers Conference” (Conferencia de Desarrolladores de Juegos) [5]. OnLive empezó a comercializar su servicio de juego bajo demanda en el año 2010, con un catálogo de videojuegos y diferentes capacidades de Juego en la Nube alojados desde sus servidores, con bajos requerimientos de procesamiento y posibilidad de jugar desde dispositivos móviles por medio de Internet. En Abril de 2013 se creó la primera plataforma de juego en la nube de código abierto “GamingAnywhere” [6].

Las empresas más populares que han ofrecido este servicio como Gaikai, OnLive, Playcast, CiiNow, entre otras [7], han sido compradas y asimiladas por otras empresas como Sony, Microsoft, Amazon y Google para uso privado.

10.2. MARCO CONTEXTUAL

Aunque el “juego en la nube” ofrece un panorama comprometedor a la industria de los videojuegos, el lograr una buena experiencia para el usuario sin inversión excesiva del hardware es un problema difícil. Esto se debe a que los jugadores son difíciles de complacer y los videojuegos requieren cada vez más requerimientos técnicos, teniendo en cuenta que conforme pasa el tiempo la demanda de los videojuegos hace necesaria una alta capacidad de respuesta y una alta calidad de vídeo, que deben brindarse sin pagar demasiado. Por lo tanto, los proveedores del servicio de “cloud gaming” no necesitan solamente el diseño de los sistemas que satisfarán las necesidades de los diferentes usuarios, sino la capacidad de recuperación de errores, la escalabilidad y la asignación de recursos. Esto hace que el diseño e implementación de sistemas de “juego en la nube” sean extremadamente desafiantes.

Por otro lado, aunque la transmisión de vídeo es una tecnología madura a primera vista, los sistemas de “juego en la nube” tienen que ejecutar juegos, por lo cual deben manejar las entradas del usuario y llevar a cabo la renderización, la captura, codificación, tráfico, transmisión, decodificación y visualización en tiempo real del juego, haciendo que este tipo de sistemas sean más difíciles de optimizar.

También hay que tener en cuenta que la mayoría de las plataformas de “juego en la nube” están cerradas o son propietarias (ver **Tabla 1:** Comparación de plataformas de Cloud Gaming), y a la fecha la única plataforma de este tipo de código abierto “GamingAnywhere” fue lanzada recientemente en Abril de 2013.

Hong, et al. [8] y Chen, et al. [9] hablan de la complejidad de lograr un equilibrio entre la calidad de experiencia del usuario (QoE) y el beneficio del proveedor de servicio de juego en la nube, por lo cual estudió el problema de la optimización de un sistema de este tipo usando diferentes métricas de QoE y rendimiento en máquinas virtuales. Los autores propusieron un algoritmo de juego en la nube para infraestructuras dedicadas, presentando un sistema prototipo y unas pruebas en un software de virtualización.

Guan, et al. [10] expresan uno de los grandes problemas del juego en la nube: El consumo eléctrico. Mostrando diferentes propuestas para la reducción del consumo, entre ellas la reducción de las frecuencias de la GPU, sin embargo al reducir la frecuencia de un GPU físico se baja considerablemente el rendimiento de todas las máquinas virtuales ejecutadas en un equipo físico. Así que los autores propusieron una arquitectura de control de dos capas llamada EvGPU (Garantías de SLA para la conservación de energía para GPU virtualizado) basada en técnicas de control de retroalimentación. Para la arquitectura propuso un bucle de control adoptado por un controlador proporcional-integral (PI) para asegurar las garantías del SLA, usando como una de sus métricas los cuadros por segundo (FPS) para cada juego, el bucle de control secundario haría un escalamiento de la frecuencia y cambiaría el voltaje para así tener un ahorro energético.

Chuah, et al. [11] hablan de la tendencia de requerir hardware de gran alcance para poder ejecutar los videojuegos de última generación, igualmente menciona el creciente deseo de los usuarios de acceder a un videojuego de alta calidad en cualquier lugar y con cualquier dispositivo con conexión a Internet. Luego hace una visión general del juego en la nube y soluciones verdes para mejorar la experiencia del juego.

Li, et al. [12] demuestran que el envío de las solicitudes en los servidores de un sistema de “cloud gaming” afecta en gran medida su funcionamiento. Por lo cual evalúa diferentes algoritmos para reducir el desperdicio de recursos de los servidores que proveen el servicio, concluyendo que un algoritmo de predicciones basadas en una red neuronal es la mejor opción para esta tarea, especialmente para juegos basados en partidas como Dota, LoL y World of Tank.

Hou, et al. [13] indican que la limitación del ancho de banda y el soporte para múltiples clientes concurrentes se ha convertido en el cuello de botella para el desarrollo de juegos en la nube. Los autores hablan de la importancia de la investigación en la codificación y comprensión de audio y video en el "cloud gaming", al igual que la implementación de un servidor de acceso concurrente. En el artículo se evalúa la latencia y concurrencia en un sistema de juego en la nube con la integración de NVIDIA GRID.

Amiri, et al. [14] y Osman, et al. [15] referencian los beneficios de implementar redes definidas por software para solucionar los problemas del juego en la nube, hablando de un controlador de SDN [16] basado en POX [17] para disminuir el retardo.

Teemu, et al. [18] exponen que en los servidores de la nube aunque se utilizan técnicas de virtualización para aislar a los usuarios y compartir recursos entre los servidores dedicados, estas técnicas pueden infligir una notable sobrecarga proporcional al rendimiento, que limita el número de usuarios para un único servidor, por lo cual proponen el uso de instancias de virtualización a nivel de sistema operativo (contenedores) en un sistema de juego en la nube, ya que estos no necesitan virtualizar todo el sistema operativo aportando un mejor rendimiento y reduciendo el consumo eléctrico. Los autores compararon el uso de máquinas virtuales con contenedores, usando QEMU [19] y Docker [20, 21] respectivamente por medio de GamingAnywhere, concluyendo que el uso de contenedores de software ofrece una mayor optimización y velocidad en los videojuegos, al igual que más usuarios concurrentes.

Shi, et al. [22] abordan el tema de la transmisión de video en tiempo real del modelo RR-GaaS, proponiendo mejoras en renderización, codificación y tecnologías de compresión, por medio de un codificador de video que selecciona un conjunto de fotogramas clave y utiliza un algoritmo de 3D image-warping (deformación de imágenes 3D), para interpolar tramas no críticas y reducir el retardo.

Wang and Dey [23] y Huang, et al. [24] exponen un deterioro en la experiencia del usuario en juegos móviles, especialmente por las redes inalámbricas que afectan negativamente el rendimiento de plataformas de juego en la nube, y proponen un conjunto de técnicas de optimización orientadas para hacer frente a los desafíos que enfrenta una red inalámbrica, en relación al tiempo de respuesta en una sesión de juego.

Lu, et al. [25] mencionan el crecimiento de juegos visualizados en 3D y la necesidad de dispositivos móviles de alta potencia de cálculo y batería. Hacen experimentos de juegos accedidos en redes 4G-LTE con móviles usando técnicas de renderización y codificación, evaluando el rendimiento de los juegos por medio de un modelo para relacionar la tasa de bits del video con los cambios en la visualización, y finalmente proponen un algoritmo de optimización para mejorar la experiencia del usuario. Concluyen que hace falta optimización de las plataformas de juego en la nube en cuanto redes móviles.

10.3. MARCO CONCEPTUAL

10.3.1. CLOUD COMPUTING

La computación en la nube es un paradigma que permite ofrecer servicios de computación a través de una red, usualmente en Internet. Las nubes suelen apoyarse en tecnologías como la virtualización, técnicas de programación como el multitenancy y/o habilidades para la escalabilidad, balanceo de carga y rendimiento óptimo, para conseguir ofrecer el recurso de una manera rápida y sencilla. [26]

10.3.2. VIRTUALIZACIÓN

En el ámbito del cloud computing, es la tecnología que a partir de harware físico permite ofrecer máquinas (CPU+ memoria del hardware físico) o almacenamiento virtual (“trozos” del disco duro físico) en cuestión de minutos y por lo tanto ofrece la flexibilidad de añadir o disminuir recursos en la infraestructura según las necesidades. El cloud computing suele apoyarse en esta tecnología para hacer un mejor uso y aprovechar los recursos del proveedor de una forma más óptima. Un ejemplo claro de uso de virtualización son las máquinas EC2 y el servicio de almacenamiento S3 de Amazon.

10.3.3. ESCALABILIDAD

Es la propiedad que cualquier sistema debería poseer para añadir nuevos componentes y así dar cobertura a un crecimiento de la demanda. Una de las ventajas más importantes del cloud computing en el nivel de infraestructura (IaaS) es la facilidad y rapidez para poder escalar los sistemas en función de las necesidades, es importante esta propiedad como la posibilidad de "desescalarlos" que también provee el cloud computing.

10.3.4. REDES DEFINIDAS POR SOFTWARE

Es un conjunto de técnicas relacionadas con el área de redes computacionales, cuyo objetivo es facilitar la implementación e implantación de servicios de red de una manera determinista, dinámica y escalable, evitando al administrador de red gestionar dichos servicios a bajo nivel. Todo esto se consigue mediante la separación del plano de control (software) del plano de datos (hardware). [27]

10.3.5 CLOUD GAMING

El juego en la nube, también llamado juego bajo demanda, es un tipo de juego en línea que permite la transmisión (streaming) directa y bajo demanda de juegos en un ordenador mediante el uso de un cliente, donde se almacena el juego real en el servidor de la compañía del juego y es transmitido directamente a las computadoras con acceso al servidor a través del cliente. Esto permite el acceso a los juegos sin necesidad de una videoconsola y en gran medida hace que la capacidad de la computadora del usuario sea una cuestión sin importancia, ya que el servidor es el sistema que está ejecutando las necesidades de procesamiento. Los controles y las pulsaciones de los botones por parte del usuario se transmiten directamente al servidor, donde se registran, y el servidor envía de vuelta la respuesta del juego a los controles de entrada. Este proceso funciona con rapidez, ya que debe haber el mínimo retardo posible para permitir una experiencia de juego casi perfecta. [28]

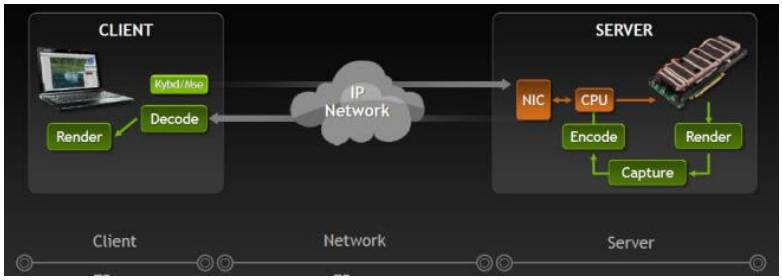


Ilustración 1: Arquitectura cliente-servidor básica de un sistema de juego en la nube

10.3.6. SERVICIOS DE CLOUD GAMING

Un servicio de juego en la nube se refiere a una plataforma comercial que brinda contenidos de videojuegos con diferentes características al público, por medio de una suscripción o forma de

monetización que debe pagar el usuario, se accede a estos juegos con un cliente ofrecido por la empresa. Este tipo de plataformas normalmente tiene convenios con las empresas desarrolladoras de videojuegos, ya que ofrecen juegos de forma masiva. Estos servicios aunque son plataformas de juego en la nube, tienen un servicio completamente consolidado, el cual no se puede personalizar ni configurar para que el usuario pueda crear su propio sistema de juego en la nube, por lo cual no se incluyó en la sección anterior. Los servicios de este tipo más característicos son los siguientes:

10.3.6.1. ONLIVE

OnLive fue un sistema de distribución de videojuegos bajo demanda de alquiler (se puede alquilar el juego durante unos días hasta 3 años o siempre) que fue estrenado en Estados Unidos el 17 de Junio de 2010. El servicio que ofrecía la compañía equivalía a usar la computación en la nube para acceder a juegos guardados en sus servidores, encargándose así del procesamiento, renderizado y almacenado en línea. El servicio era compatible con cualquier Mac basado en Intel, computadores con Windows Vista o Windows 7, celulares con Android o iOS y también se transmitían los juegos a través de una consola llamada "OnLive MicroConsole" conectada a un televisor. Para poder acceder al servicio se requería de una conexión de banda ancha mínimo de 1.5Mbps para una calidad de imagen similar a Wii, mientras que se requerirán 4-5Mbps para obtener alta resolución.

La empresa tenía convenios con diferentes marcas como Electronic Arts, Take-Two Interactive, Ubisoft, Epic Games, Atari, Codemasters, THQ, Warner Bros., 2D Boy y Eidos Interactive. En Diciembre 10 del 2010, OnLive obtuvo una licencia para juego en la nube por parte de la oficina de patentes de Estados Unidos.

El 5 de abril de 2015, fue realizada la compra de OnLive por parte de Sony, adquiriendo todos los patentes de OnLive y programando el cierre del servicio para el 30 de abril del 2015.

Ilustración 2: Modelo del juego como servicio de OnLive

10.3.6.2. GAIKAI

Gaikai era un servicio de juego basado en la nube que permitía a los usuarios jugar demos de los juegos de PC más importantes y aplicaciones de forma instantánea y gratuita desde una página web o dispositivo conectado a Internet. Al final del demo, el cliente tenía la opción de comprar el juego o producto a un distribuidor local, tienda en línea, descarga digital o continuar con el streaming del producto con un pago por uso.

La tecnología propietaria de Gaikai funcionaba en parte mediante el uso de plug-ins instalados previamente tales como Java o Adobe Flash. Un vídeo de demostración del servicio en la GDC San Francisco 2010 mostró Call of Duty 4: Modern Warfare, World of Warcraft, EVE Online, Spore, Mario Kart 64 y Adobe Photoshop todos ejecutándose en Adobe Flash Player [29]. El 2 de julio de 2012, Sony Computer Entertainment anunció que había alcanzado un acuerdo para adquirir el servicio basado en la nube por el equivalente a 380 millones de dólares estadounidenses y se cerró la compañía.

10.3.6.3. PLAYSTATION NOW

PlayStation Now (PS Now) [30] es un servicio de juego en la nube desarrollado por Sony Interactive Entertainment y Gaikai. Los usuarios de este servicio pagan por el acceso a una selección de juegos originales de PlayStation 3, a través de una suscripción o alquiler de juegos por un tiempo limitado. PS Now es compatible con los diferentes productos de Sony como Playstation Vita, Playstation TV y algunos Smart TV de la marca. Para el correcto funcionamiento del servicio se requiere una de un control DualShock 3 ó 4 y una conexión a Internet de mínimo 5 Mbps.

10.3.6.4. GAMENOW

Es un servicio de juegos en la nube impulsado por Ubitus, ofrece acceso a un catálogo de juegos bajo demanda en línea a través de un software de la empresa, con la posibilidad de probar de forma gratuita juegos antes de comprarlos. GameNow está disponible en una amplia gama de dispositivos Over-the-Top (OTT), incluyendo Smart TV, Google TV, PC y Mac. Además de eso, una versión móvil de GameNow está disponible exclusivamente para usuarios de Verizon Wireless LTE para disfrutar del mismo nivel de calidad de consola de juegos directamente desde sus redes LTE.

10.3.6.5. GAMEFLY

GameFly es una empresa americana de alquiler de videojuegos, que tiene un servicio de suscripción que da acceso a un suministro específico de discos de videojuegos, enviando éstos a la casa del usuario. Posee un servicio de streaming de videojuegos especializado para Smart TV's de LG, Samsung y Philips, por medio de una cuota mensual, disfrutando de una colección de juegos sin necesidad de tener una consola conectada. Para una calidad de imagen estándar (SD)

requiere de una conexión mínima de 4 Mbps, para obtener una calidad de alta definición (FHD), necesita 8 Mbps. Sólo está disponible para Smart TV's [31].

10.3.7. PLATAFORMAS DE JUEGO EN LA NUBE

En un enfoque simple para soportar un juego en la nube se debe usar un cliente de streaming, y aunque existen clientes ligeros de streaming de escritorio genéricos como LogMeIn, TeamViewer y UltraVNC, éstos alcanzan velocidades bajas de fotogramas, provocando una mala experiencia del usuario y juegos relativamente lentos. Por ello se requiere de un cliente ligero diseñado específicamente para tener menos retardos y más fluidez, estos existen en las plataformas de juego en la nube, las cuales se encargan del procesamiento y codificación de las escenas del juego, incluyendo la entrada del usuario y la transmisión de video desde el servidor.

Sin embargo, el juego en la nube tiene poca investigación debido a su uso primordialmente privado [6], ya que existen pocas plataformas [32] que puedan ofrecer este servicio y están limitadas con ciertos sistemas operativos en el servidor o el cliente, hardware, funcionalidades y costos. Pese a lo anterior existen dos plataformas de código abierto: GamingAnywhere y MoonLight Game Streaming. En la Tabla 1, se pueden observar algunas características de plataformas de éste tipo. A continuación se describirá las plataformas de juego en la nube más características en orden de importancia:

10.3.7.1. GamingAnywhere

Es una plataforma de juego en la nube extensible, portable y tiene capacidad de reconfiguración debido a que se tiene acceso a su código fuente, con posibilidad de instalarse en un servidor Windows, Linux y OS X, y accederse desde un cliente de los sistemas operativos anteriores incluyendo iOS y Android. Fue el producto de la investigación de Huang Hsu & Chen [6].

10.3.7.2. Steam In-Home Streaming

Es una funcionalidad del cliente de Steam que permite hacer streaming de videojuegos (uno a la vez), pero el servidor solamente puede tener sistema operativo Windows .

10.3.7.3. Nvidia GameStream y Moonligh Game Streaming

Nvidia GameStream es una plataforma que transmite los juegos de un computador con sistema operativo Windows o Mac a un dispositivo NVIDIA SHIELD. Para ello, aprovecha la potencia de tarjetas gráficas GeForce GTX que envían las imágenes en streaming [13].

Moonlight (anteriormente conocido como Limelight) es una implementación de código abierto del protocolo de GameStream de NVIDIA, con la diferencia que el cliente puede ser cualquier dispositivo con Android, Windows o Linux, y éste es gratuito.

10.3.7.4. Remote Play

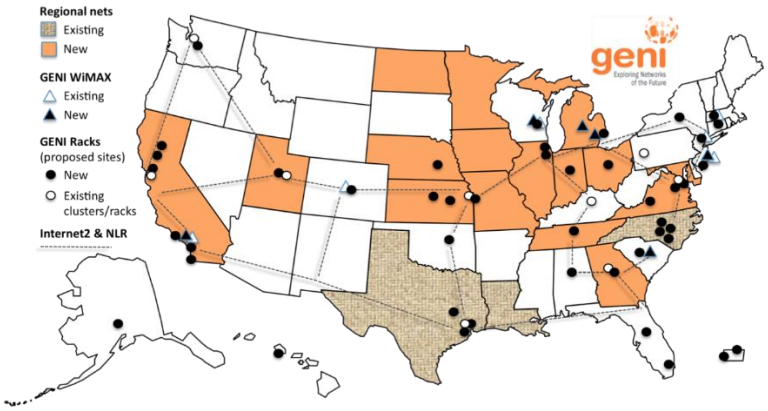
Es un servicio de Sony, por el cual se puede acceder a un juego de Playstation 3 ó 4 vía streaming en una red local, a través de una Playstation TV y Vita o con un Sony Xperia que sea soportado por la marca.

10.3.8. GENI

GENI (Global Environment for Network Innovations) proporciona un laboratorio virtual para la investigación y educación en redes y sistemas distribuidos. Es adecuado para explorar redes a escala, promoviendo así innovaciones en la ciencia, seguridad, servicios y aplicaciones de la red. GENI permite:

- Obtener recursos computacionales de ubicaciones alrededor de los Estados Unidos
- Conectar recursos de computación utilizando redes de Capa 2 en las topologías que mejor se adapten a experimentos.
- Instalar software personalizado o incluso sistemas operativos personalizados en estos recursos de computación
- Controlar el cómo los conmutadores de red en los experimentos manejan los flujos de tráfico
- Ejecutar protocolos de Capa 3 y superiores instalando software y proporcionando controladores de flujo para los conmutadores.
- Los recursos disponibles para los experimentadores de GENI incluyen Racks GENI, redes backbone regionales y nacionales, estaciones WiMAX, entre otros.

GENI es un banco de pruebas federado, lo que significa que los recursos de GENI son propiedad y son operados por diferentes organizaciones, como se muestra en la siguiente imagen:



Algunas características especiales que tiene GENI son las siguientes.

- Ofrece una infraestructura de experimentación a gran escala con equipos completamente reales, además puede potencialmente proporcionar más recursos de los que normalmente se encuentran en cualquier laboratorio.

- GENI da acceso a cientos de recursos ampliamente distribuidos, incluyendo recursos de computación tales como máquinas virtuales y "bare-machines", recursos de red como enlaces, conmutadores y estaciones WiMax.

- Conectividad no IP entre recursos. GENI permite configurar conexiones de capa 2 entre recursos de cálculo y ejecutar nuestros protocolos capa 3 o superior que conecten estos recursos.

- Programación profunda. Con GENI se puede programar no sólo los hosts finales de una red experimental sino también los switches en el núcleo de la red. Esto permite experimentar con nuevos protocolos de la capa de red o con nuevos algoritmos de enrutamiento IP.

- Reproducibilidad. Se puede obtener acceso a ciertos recursos exclusivos, incluidos recursos de CPU y recursos de red. Esto da control sobre el ambiente de los experimentos y por lo tanto la capacidad para que cualquier persona pueda replicar experimentos en condiciones idénticas o similares.

- Instrumentación y herramientas de medición. GENI tiene dos sistemas de instrumentación y medición que se pueden usar para instrumentar los experimentos. Estos sistemas proporcionan sondas para mediciones activas y pasivas, almacenamiento de datos de medición y herramientas para visualizar y analizar datos de medición.

Tabla 1: Comparación de plataformas de Cloud Gaming

Nombre	Estado	Estado de desarrollo	Código fuente disponible	Plataformas o S.O.	Licencia	Precio
GamingAnywhere	Activo	Activo	Si	**Server:** Windows 7 o superior, OS X, Linux **Cliente:** Windows XP o superior, OS X, Android 4.1+, Linux	Open-source	Gratis
Steam In-Home Streaming	Activo	Activo	No	**Server:** Windows Vista o superior **Cliente:** Windows Vista o superior, OS X, Steam OS, Linux	Propietaria	Freeware
Remote Play y Playstation Now	Activo	Activo	No	PlayStation 3, PlayStation 4, PlayStation Vita, PlayStation TV, Sony Xperia	Propietaria	Freeware
Ubitus GameCloud	Activo	Activo	No	**Server:** Windows Vista o superior **Cliente:** Smart TVs, GoogleTV, OS X, Smartphones, Windows Vista o superior	Propietaria	N/A

Nombre	Estado	Estado de desarrollo	Código fuente disponible	Plataformas o S.O.	Licencia	Precio
StreamMyGame	Activo	Caído	No	**Server/Cliente:** Windows XP o superior, Linux	Propietaria	**Premium:** $9.99 por año **Ilimitado:** $19.99 por año
Cloud Gaming eXtreme	Activo	Activo	No	**Server:** Windows Vista o superior, Windows Server 2012 R2 **Cliente:** Windows XP o superior, OS X, Linux, Android, iOS, navegador web	Propietaria	$0.30 por hora con Amazon Web Services
Nvidia GRID y GameStream	Activo	Activo	No	**Servidor:** Windows 7 o superior, OS X, Linux **Cliente:** Nvidia Shield	Propietaria	Freeware

10.3.9. ARQUITECTURA DE JUEGO EN LA NUBE

Cai, et al. [33] muestran la estructura modular de un juego en línea convencional, ésta se explica a continuación para tener claridad de las arquitecturas que se crean para optimizar este diseño. Consta de cuatro módulos principales (Ilustración 3: Un sistema modular de cloud gaming [33]):

Módulo de entrada (Input module): Recibe la información de control del jugador, por ejemplo las pulsaciones de un teclado.

Módulo de la lógica del juego (Game logic module): es el encargado de la manipulación de contenidos del juego, tiene componentes que se invocan entre sí e interactúan con la interfaz de red, motor de renderizado, y la interfaz de E / S para facilitar el procesamiento del juego.

Módulo de red (Networking module): Intercambia información mediante el servidor del juego, por ejemplo cómo los avatares interactúan con los demás.

Módulo de presentación (Rendering module): Renderiza el videojuego y lo presenta al jugador.

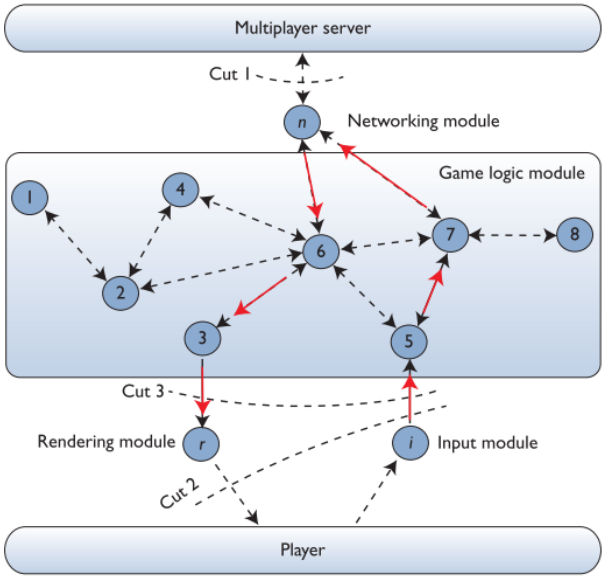


Ilustración 3: Un sistema modular de cloud gaming [33]

Las flechas rojas en la Ilustración 3: Un sistema modular de cloud gaming [33] demuestran una interacción entre el jugador y el sistema de juego durante una sesión de juego. El módulo de entrada transmite las instrucciones del jugador al componente 5, después la información procesada se entrega al componente 7 que invoca el módulo de red para llevar a cabo el intercambio de información. Después del procesamiento sucesivo por los componentes 6 y 3, el módulo de representación genera y transmite el vídeo a la pantalla de juego del jugador.

Así que, ¿cuál es la esencia de un juego basado en la nube desde esta perspectiva? Los tres cortes gráficos de la Figura 1 ilustran la respuesta. El corte 1 demuestra que todos los módulos están implementados en el terminal (por ejemplo un computador), mientras que el módulo de red es la

interfaz entre los clientes del juego y el servidor de juego en línea. En este caso, la nube se utiliza sólo como un servidor de intercambio de información, que es el diseño tradicional de los juegos en línea. El corte 2 va a otro nivel: el terminal contiene sólo el módulo de entrada, mientras que los hosts de la nube tienen todos los módulos y componentes restantes, haciendo que el vídeo en tiempo real del juego se transmita al jugador a través de Internet, este modelo se denomina Remote Rendering GaaS (Renderización a distancia) o RR-GaaS [34].

El corte 3 ilustra otra idea de diseño: la entrada y módulos de representación se ejecutan en el terminal, mientras que los otros módulos se ejecutan en la nube, este enfoque se denomina Local Rendering GaaS (Renderización local) o LR-GaaS [33, 35]. De acuerdo con estos cortes, podemos ver que la esencia del juego en la nube es aprovechar los recursos de la nube para ejecutar varios módulos de juegos, lo que reduce la carga de trabajo y aumenta la eficiencia del terminal.

A continuación se hará una descripción de las arquitecturas anteriores: RR-GaaS, LR-GaaS y una nueva propuesta denominada CRA-GaaS:

10.3.9.1. RR-GaaS

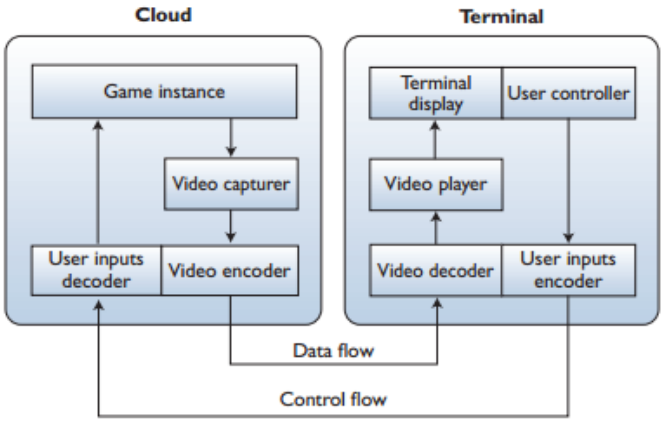


Ilustración 4: Arquitectura del modelo RR-GaaS [33]

RR-GaaS es el modelo de servicio para juego en la nube más maduro . Empresas como OnLive, Gaikai, y G-Cluster han proporcionado GaaS comercializados al público, siguiendo un modelo de negocio donde se ejecuta un código fuente en una plataforma que posee juegos, ésta es operadada por un proveedor de servicios cloud, y luego se entregan los fotogramas de video a los dispositivos de los usuarios a través de algún software ligero.

La Figura 2 muestra la arquitectura de este modelo. La nube virtualiza un entorno de ejecución e inicia una instancia de juego (Game instance) una vez que recibe una solicitud de conexión de un jugador, la instancia es representada como un flujo de datos, incorpora el vídeo del juego en tiempo real en la nube, y registra la captura y procesamiento del vídeo cuadro por cuadro. A continuación, el servidor de juego en la nube transmite las tramas de vídeo al terminal del jugador a través de Internet después de pasar en un codificador de vídeo. Estas tramas codificadas se reconstruyen en el decodificador de video del terminal del cliente, y el reproductor de video los muestra en la pantalla. A la inversa, como lo ilustra el flujo de control, el controlador registra las entradas del

usuario y la terminal los transmite al servidor de la nube después de una codificación previa, el servidor de la nube recibe estas señales codificadas y los decodifica en las entradas de control de las instancias del juego, para que haya una interacción del jugador con el juego alojado en la nube.

10.3.9.2. LR-GaaS

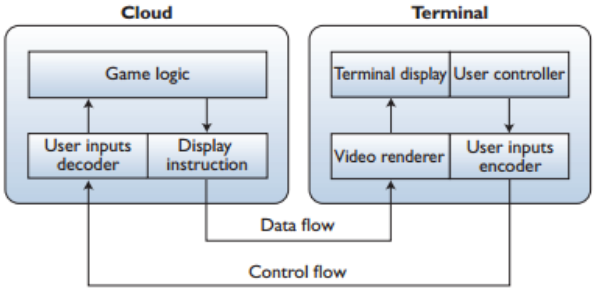


Ilustración 5: Arquitectura del modelo LR-GaaS [33]

En este modelo se tienen mejoras de rendimiento de hardware en los terminales que accederán al juego incluyendo dispositivos móviles, se pueden realizar reproducciones complejas de escenas de juego, dado que el módulo de representación está implementado en el terminal del juego para eliminar la carga de la transmisión de video en tiempo real en la red,

En la Ilustración 5: Arquitectura del modelo LR-GaaS [33] se puede observar la arquitectura para LR-GaaS [36]. En el flujo de datos, el videojuego no se renderiza en el servidor de la nube, en su lugar la lógica del juego genera un conjunto de instrucciones de la pantalla para representar los gráficos del juego y los envía a la terminal a través de Internet. El terminal interpreta la instrucción de la pantalla con un conjunto de instrucciones designadas y renderiza el video del juego localmente en el terminal. El flujo de control inverso es similar al del modelo RR-GaaS.

El beneficio más destacado para el sistema LR-GaaS es que el servidor de la nube ya no tiene que transmitir tramas de vídeo juegos en tiempo real a los terminales por medio de Internet, lo que reduce significativamente la carga de trabajo de la red. Este modelo se ha aplicado principalmente en juegos de navegador [37], ya que el navegador es un cliente ligero que se encuentra en la mayoría de computadoras y dispositivos móviles. Se usan principalmente plugins, librerías, scripts o motores de juego para aumentar la capacidad de renderización de los navegadores, por ejemplo Adobe Flash, Unity Web Player, Java Applet, Akihabara, ammo.js, entre otros.

10.3.9.3. CRA-GaaS

La diferencia intrínseca entre RR-GaaS y LR-GaaS es el entorno de ejecución del módulo de renderizado, como los cortes 2 y 3 de la Figura 1 indican. Por lo anterior se creó otro modelo llamado Cognitive resource allocation GaaS (Asignación de recursos cognitiva GaaS) o CRA-GaaS, el cual ofrece una mayor flexibilidad al adaptar el servidor de juego en la nube para diversas circunstancias y mejorar así la estabilidad de la red [38].

Este modelo tiene un conjunto de componentes interdependientes que se ejecutan en el servidor de la nube o en el terminal del cliente, según lo determinado por las condiciones actuales del terminal y su conexión a Internet. En otras palabras, tiene capacidades cognitivas que permiten a la plataforma de juego en la nube seleccionar combinaciones de componentes óptimos, de acuerdo con el contexto del sistema o de factores externos, para así proporcionar un GaaS de alta eficiencia. La Figura 4 muestra la arquitectura de este modelo (CRA-GaAs):

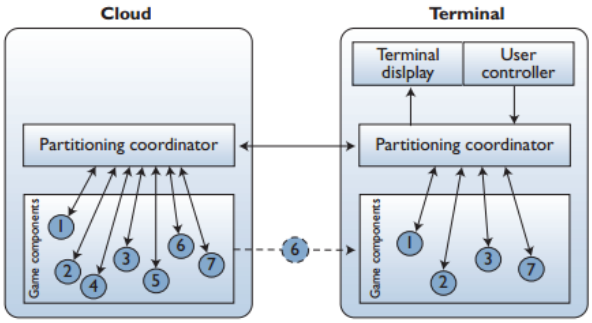


Ilustración 6: Arquitectura del modelo CRA-GaaS [33]

Los programas o juegos son modularizados como componentes, que pueden migrar de la nube al dispositivo del usuario durante una sesión de juego y concatenarse dinámicamente entre sí para formar un juego completo. Es decir, el terminal puede traer y ejecutar un conjunto de componentes redundantes del juego desde la nube para reducir así la carga de los servidores.

11. METODOLOGÍA

La metodología que se usará en el proyecto es la siguiente:

I. **Realizar una búsqueda bibliográfica para seleccionar una plataforma que provea el servicio de Juego en la Nube:** En este apartado se analizarán diferentes plataformas de Juego en la Nube y se seleccionará la más adecuada para la infraestructura que proveerá el servicio.
II. **Estudio de la plataforma GENI:** Se estudiará GENI para saber cómo usarla haciendo algunos de sus laboratorios, y pruebas de la implementación de la plataforma seleccionada de Juego en la Nube.
III. **Seleccionar el protocolo y controlador que se van a configurar en la SDN para dar soporte a la plataforma de Juego en la Nube:** Se designará el protocolo y controlador SDN que se empleará en el sistema, por lo cual se deberán analizar factores como la capa, controlador, lenguaje o las necesidades que tenga la plataforma (como el retardo, consumo eléctrico, sesiones, entre otros).
IV. **Diseñar y configurar una infraestructura SDN en GENI para proveer el servicio de Juego en la Nube:** Se implementará la plataforma de Juego en la Nube en una SDN configurada en GENI, la cual debe estar funcionando en esta etapa.
V. **Evaluar el desempeño de la implementación de la plataforma:** Se probarán técnicas en el controlador de la red definida por software para mejorar el desempeño de la plataforma de Juego en la Nube, las cuales serán comparadas con la red original y sus resultados serán documentados.

12. PROCEDIMIENTO

12.1. SELECCIÓN DE PLATAFORMA DE CLOUD GAMING

Como se puede ver en la **Tabla 1:** Comparación de plataformas de Cloud Gaming), existen pocas plataformas de juego en la nube y todas son privativas a excepción de "GamingAnywhere" la cual es open-source. Pero para la determinación de la plataforma que proveerá el servicio, no es un factor decisivo que el software sea privativo o no, ya que existen plataformas freeware las cuales se pueden instalar y usar sin ningún problema.

"Steam In-Home Streaming", es una funcionalidad del cliente de Steam que permite hacer streaming de videojuegos, pero aunque Steam es multiplataforma tiene un problema en donde el servidor que "transmitirá" el juego debe tener un sistema operativo Windows y además sólo se puede transmitir un juego a la vez. Por lo anterior no sería eficiente implementar un sistema de Cloud Gaming a un público de usuarios con esta plataforma, ya que sólo un usuario podría acceder a la vez en el servidor y éste último sólo podría tener Windows, creando así limitaciones en el control de la máquina. Cabe destacar que para un uso personal podría utilizarse, como lo hizo [39] implementando un servicio de cloud gaming en un servidor de EC2 de Amazon.

"Remote Play y Playstation Now", son plataformas de Sony especializadas en consolas y dispositivos de la marca, además que requieren de una suscripción por lo cual quedan descartadas, al igual que "Ubitus GameCloud" y "Cloud Gaming eXtreme".

La tecnología GRID de Nvidia y su servicio GameStream, permite el streaming de videojuegos pero con tres limitantes: El cliente sólo puede acceder desde una Nvidia Shield, el servidor deberá tener una tarjeta gráfica Nvidia GRID con soporte de esta tecnología y para usar el servicio de GameStream se tendría que pagar una suscripción.

Por todo lo anterior se seleccionó como plataforma de juego en la nube "GamingAnywhere",ya que es open-source, está actualizada, tiene un foro de soporte, es configurable, se pueden iniciar varios juegos a la vez desde diferentes clientes y es multiplataforma.

[40]hicieron una investigación y comparación de distintos sistemas de Cloud Gaming mostrando las bondades de GamingAnywhere, concluyendo los siguientes aspectos de ella:

1. Es extensible, portable y tiene capacidad de reconfiguración debido a que se tiene acceso a su código fuente, con posibilidad de instalarse en un servidor Windows, Linux y OS X, y accederse desde un cliente de los sistemas operativos anteriores incluyendo iOS y Android.

2. GamingAnywhere produce un retardo de procesamiento de cada fotograma en 34 ms, lo que es 3+ y 10+ veces más corto que las plataformas de juego en la nube: OnLive y StreamMyGame, respectivamente. Por lo cual tiene mejoras de rendimiento con menos tráfico de red. Lo anterior es importante ya que hay que tener en cuenta que el usuario

quiere una alta calidad de imagen como 720p (1280x720) a 50 fps (fotrogramas por segundo), dando inherentemente una mayor tasa de bits que hacen que los sistemas de juego en la nube sean vulnerables a una mayor latencia de la red y por lo tanto un retardo en repuesta. Un retardo se traduce en una experiencia del usuario degradada, ya que incuso una tan baja de 100 ms en un juego en línea de primera persona, haría que otros jugadores del mismo juego tomaran ventaja del "lag", disminuyendo así la calidad del servicio.

3. En un enfoque simple para soportar juegos en la nube se usan clientes ligeros de streaming de escritorio genéricos, como LogMeIn, TeamViewer, y UltraVNC. Sin embargo en la investigación revelaron que los clientes genéricos alcanzaban velocidades bajas de fotogramas en un promedio de 13.5 fps, por lo cual los juegos eran relativamente lentos. Para lograr una mejor experiencia del usuario se requiere de un cliente ligero diseñado específicamente para juegos en la nube, por ejemplo Gaikai, Onlive, StreamMyGame y otros que aparecen en (**Tabla 1:** Comparación de plataformas de Cloud Gaming). Pero estos clientes especializados de juego en la nube sufren también de retardos, aunque menores en comparación con los clientes genéricos.

   La plataforma de streaming con mejor desempeño en la investigación fue GamingAnywhere, ya que tiene componentes de transmisión de video que pueden ser reemplazados por otros algoritmos, estándares o protocolos debido a la naturaleza de su código abierto. Por defecto emplea x264 un codificador H.264/AVC altamente optimizado para codificar la captura de video, pero incluye diferentes codificadores como H.264 MVC que pueden ser conectados sin cambios significativos.

   Cabe resaltar que GamingAnywhere ha sido diseñado para ser eficiente, como puede verse por ejemplo en su minimización de tiempo y espacio mediante el uso de buffers circulares compartidos para reducir el número de operaciones de copia de la memoria. Estas optimizaciones permiten que la plataforma proporcione una experiencia de juego de alta calidad con corto retardo de respuesta. En el estudio usaron un servidor con procesador Intel i7 ofreciendo videos de 720p y 1080p en tiempo real en $\geq$ 35 fps, que es equivalente a menos de 28,6 ms de tiempo de procesamiento para cada fotograma de vídeo, con una calidad de vídeo alta. GamingAnywhere logra una calidad de vídeo de 3 dB y 19 dB mayor que la de OnLive y StreamMyGame, en términos de promedio de la relación señal a ruido de pico (PSNR).

4. En el estudio se hizo una comparación de GamingAnywhere, Onlive, y StreamMyGame, donde corrieron tres juegos: LEGO Batman: The Videogame (Batman), F.E.A.R. 2: Project Origin (FEAR) y Warhammer 40,000: Dawn of War II (DOW), usando la topología de la (**Ilustración 7**: Topología de red usada en experimento de GA**)**. Evaluando tres tipos de retardo:

   a. **Processing delay - Retardo de procesamiento (PD):** Es el tiempo requerido para que el servidor reciba y procese las órdenes de un jugador, y para codificar o transmitir la trama correspondiente a ese cliente.
   b. **Playout delay - Retardo de reproducción (OD):** Es el tiempo requerido para que el cliente reciba, decodifique y muestre una trama en la pantalla.

c. **Network delay – Retardo de red (ND):** Es el tiempo requerido para una ronda de intercambios de datos entre el servidor y el cliente. ND también se conoce como el tiempo de ida y vuelta (RTT).

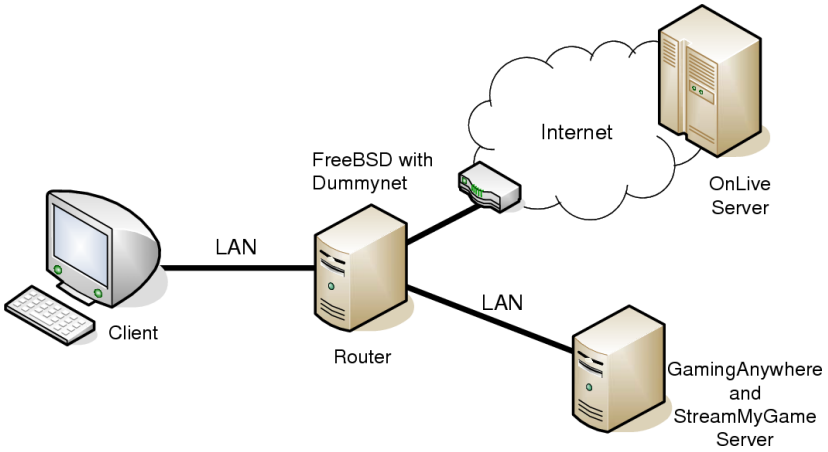


Ilustración 7: Topología de red usada en experimento de GA

.

Teniendo como resultados:

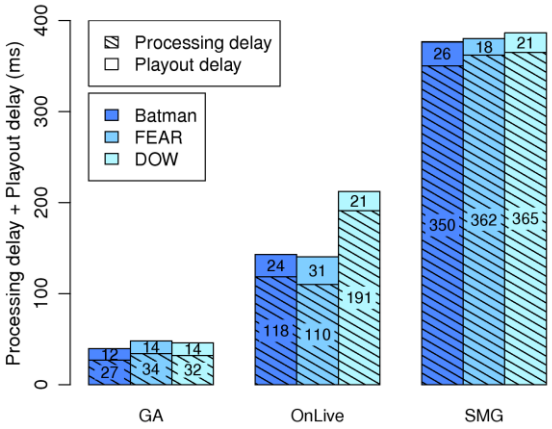


Ilustración 8: Respuesta al retardo de GamingAnywhere, OnLive y StreamMyGame

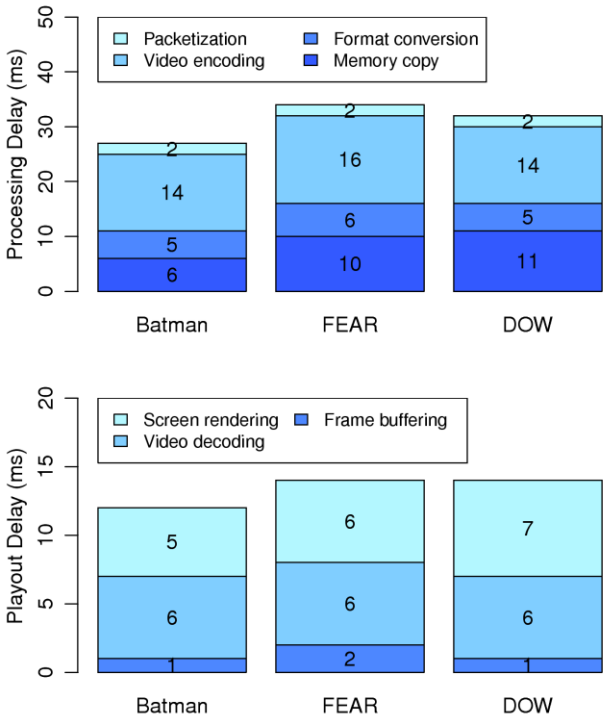


Ilustración 9: Descomposición del retardo en GamingAnywhere

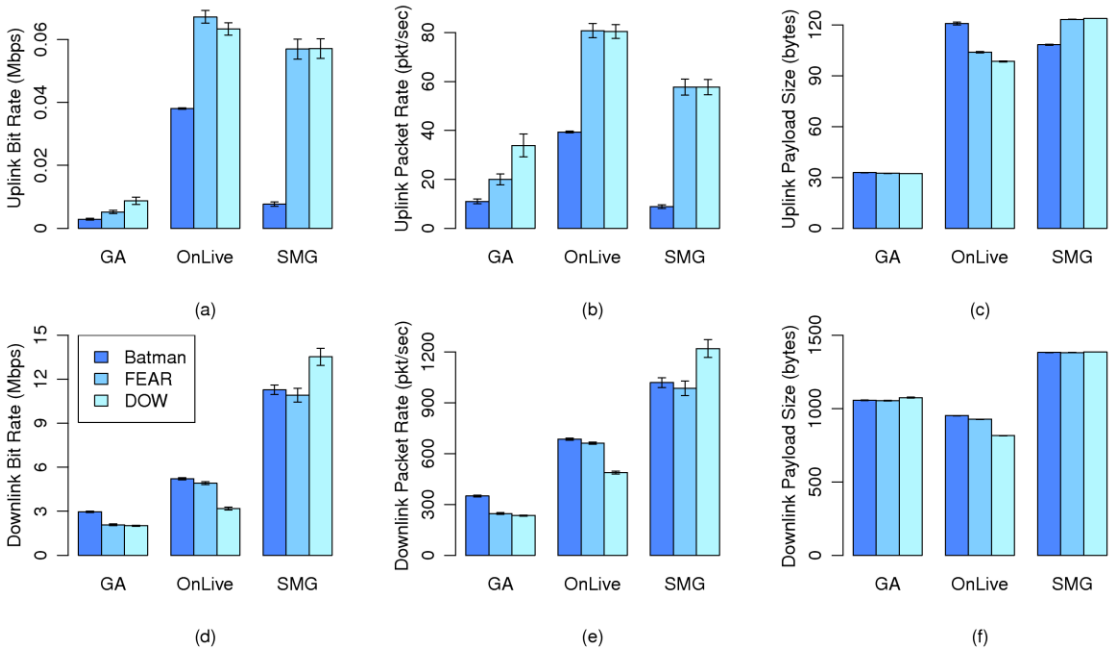


Ilustración 10: Cargas de red consideradas en sistemas de juego en la nube

Por los resultados anteriores del estudio de [40] y al comparar las diferentes plataformas, se decidió como plataforma de juego en la nube GamingAnywhere.

12.2. SELECCIÓN DE PROTOCOLO Y CONTROLADOR DE SDN

Como se ha visto anteriormente, SDN es una arquitectura de red cuyo dinamismo, manejabilidad, rentabilidad y adaptabilidad permiten que sea adecuada para la naturaleza y alto consumo de ancho de banda de las aplicaciones modernas. Al separar el control de la red de las funciones de reenvío con una bien definida API entre ambos, permite una programabilidad del control de red y la abstracción de la infraestructura. **(Ilustración 11**: Arquitectura simplificada de SDN utilizando el protocolo Openflow**)**.

Para implementar una SDN en un sistema de juego en la nube, es importante seleccionar un protocolo de este tipo de arquitectura que se adapte a las necesidades de la red y de la plataforma, ya que aunque OpenFlow es altamente divulgado no es el único estándar para un protocolo de comunicaciones que permita la interacción entre el plano de control y de datos, por lo cual deben compararse protocolos en primer lugar para esta tarea.

Hay que tener en cuenta que un controlador SDN toma el papel del "cerebro" de dicha red, es decir el punto de control estratégico que retransmite información a los switches/routers de debajo (a través del southbound API) y a las aplicaciones y la lógica de negocio encima (a través del northbound API). Un controlador SDN se basa típicamente en un conjunto de módulos pluggable (que se pueden conectar y desconectar libre y fácilmente), que realizan diversas tareas de red, como realizar un inventario de todos los aparatos de red disponibles en ésta, saber sus capacidades, agrupar estadísticas de red, mejorar y brindar seguridad a la red, etc. Actualmente se pueden asociar dominios de controladores SDN usando interfaces de aplicaciones comunes como los protocolos de OpenFlow y OVSDB (Open Virtual Switch DataBase).

Existen múltiples controladores SDN OpenSource, algunos que se han desarrollado por la comunidad, y otros que han sido desarrollados por empresas que han liberado el código. Estos controladores se diferencian por su lenguaje de programación, plataforma y características, pero en esencia realizan las mismas funciones: hablar con el protocolo (por ejemplo OpenFlow) para comunicarse con los dispositivos de red.

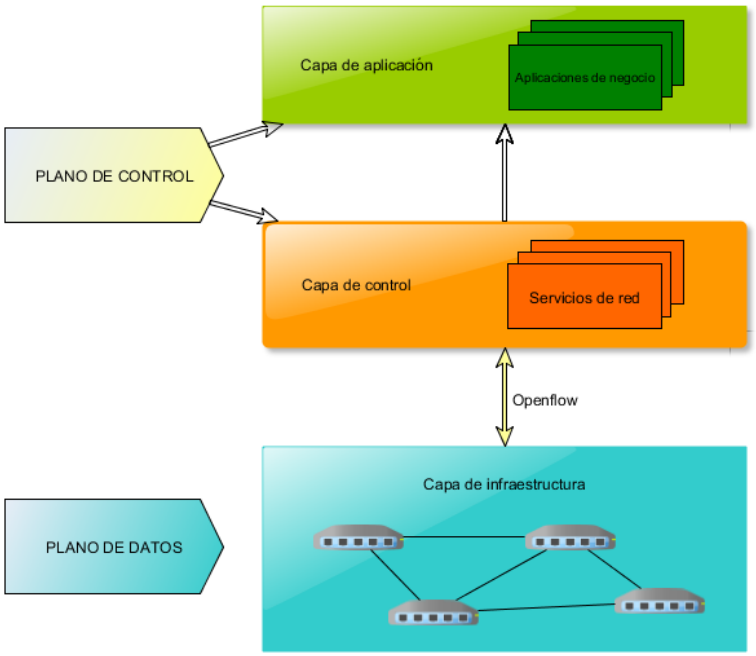


Ilustración 11: Arquitectura simplificada de SDN utilizando el protocolo Openflow

Por lo anterior es importante una buena selección del protocolo y controlador que se va a usar en el sistema de juego en la nube, como se mostrará a continuación:

.

12.2.1. PROTOCOLO DE SDN

Open Networking Foundation define a OpenFlow como la primera interfaz de comunicaciones estándar entre las capas de control y de reenvío de una arquitectura SDN, pero ésta puede no seguir siendo el protocolo predominante, debido a que aunque la mayoría de los proveedores de red ya han desarrollado equipos basados en OpenFlow, también están diseñando arquitecturas SDN que utilizan métodos de comunicación alternativos, incluyendo protocolos de red existentes, tales como MPLS y NETCONF. A continuación se hace una comparación de protocolos SDN:

1. **Border Gateway Protocol (BGP):** Es un protocolo utilizado para intercambiar información de enrutamiento entre hosts de gateways en una red de sistemas autónomos, este protocolo se utiliza a menudo entre hosts de gateways en internet y también se considera un protocolo de gateway exterior estandarizado. BGP a menudo también se clasifica como un protocolo de ruta vectorial o un protocolo de vector de distancia. En la red cada host de gateway tiene típicamente su propio router. La tabla de enrutamiento contiene una lista de routers conocidos, las direcciones que pueden alcanzar, y una métrica de costo asociado con la ruta a cada router permitiendo que la mejor ruta disponible sea elegida.

   Los proveedores están buscando utilizar BGP en redes definidas por software híbridas. Algunos argumentan que el protocolo sothbound en una arquitectura SDN es menos

importante que la agilidad operativa y la capacidad de programación que ofrece SDN, con o sin OpenFlow. Como resultado, los proveedores identificaron a BGP como un protocolo con potencial de permitir la programabilidad de red prometida por SDN.

2. **NETCONF:** Es un protocolo de gestión de red Internet Engineering Task Force (IETF). Proporciona a un administrador o ingeniero de red una forma segura de configurar un firewall, switch, router o cualquier otro dispositivo de red. Se basa en la llamada de procedimiento remoto (RPC) y fue diseñado para resolver los problemas que existen con los protocolos Protocolo de Gestión de Red Simple y la Interfaz de Comandos en Línea, ya que se refieren a la configuración de dispositivos de red.

   La Fundación Open Networking (ONF) abrazó recientemente NETCONF y lo hizo obligatorio para la configuración de dispositivos compatibles con OpenFlow. La especificación, llamada OF-CONFIG, requiere que los dispositivos que la soportan deben implementar el protocolo NETCONF como el transporte.

3. **Protocolo de Presencia y Mensajería Extensible (XMPP)**: Es un protocolo que se basa en el Extensible Markup Language. Su uso previsto es para mensajería instantánea y la detección de la presencia en línea. El protocolo funciona entre o en los servidores y facilita la operación en tiempo casi real. XMPP ha surgido recientemente como un protocolo SDN alternativo a OpenFlow en redes SDN híbridas y puede ser utilizado por el controlador para distribuir información tanto de plano de control como de gestión de plano a los puntos finales de servidor. Gestiona la información en todos los niveles de abstracción, hasta llegar al flujo.

4. **Protocolo de Gestión de Base de Datos Open vSwitch (OVSDB)**: Es un protocolo de configuración de OpenFlow que está destinado a administrar las implementaciones Open vSwitch. Open vSwitch es un conmutador virtual que permite la automatización de la red y el soporte de interfaces y protocolos de administración estándar, tales como NetFlow. El protocolo también soporta la distribución a través de múltiples servidores físicos.

   En una implementación Open vSwitch, un grupo de control contiene administradores y controladores que utilizan el protocolo OVSDB para suministrar información de configuración al servidor de base de datos de switch. Los controladores usan OpenFlow para especificar los detalles de los flujos de paquetes a través del conmutador. Cada administrador y controlador pueden dirigir varios switches, y cada switch puede recibir directivas de varios gestores y controladores.

5. **Perfil de Transporte MPLS (MPLS-TP)**: Es el perfil de transporte para la conmutación de etiquetas multiprotocolo. MPLS-TP está diseñado para ser utilizado como una tecnología de capa de red en redes de transporte. Las extensiones del protocolo a MPLS están siendo diseñados por el IETF basado en los requerimientos proporcionados por los proveedores de servicios. El protocolo será una aplicación de conmutación de paquetes orientado a la conexión (CO-PS) que ofrece una implementación MPLS dedicada a eliminar características que no son relevantes a las aplicaciones CO-PS y a agregar dispositivos que proporcionan soporte a la funcionalidad de transporte crítico.

La Fundación Open Networking propuso cambios a MPLS que incluyen el uso de los datos de plano estándar MPLS con un plano de control más simple basado en SDN y OpenFlow. Al contar con un plano de control simplificado que se desacopla del plano de datos, la ONF argumenta que es capaz de optimizar los servicios globalmente, hacer los servicios más dinámicos y crear nuevos servicios al programar aplicaciones de red sobre el controlador SDN.

6. **OpenFlow:** Es un protocolo que permite a un servidor decirle a los conmutadores de red adónde enviar paquetes. En una red convencional, cada conmutador tiene software propietario que le dice qué hacer. Con OpenFlow se centralizan las decisiones de migración de paquetes, de modo que la red se puede programar independiente de los conmutadores individuales y equipo del centro de datos.

   En un conmutador convencional, la transferencia de paquetes (la trayectoria de datos) y el enrutamiento de alto nivel (la trayectoria de control) suceden en el mismo dispositivo. Un conmutador OpenFlow separa la trayectoria de datos de la trayectoria de control. La porción de trayectoria de datos reside en el propio conmutador, un controlador separado toma las decisiones de enrutamiento de alto nivel, el conmutador y el controlador se comunican por medio de este protocolo. En la actualidad se ha aumentado la preferencia por OpenFlow en aplicaciones como movilidad de máquina virtual (VM), redes críticas y redes móviles basadas en IP de nueva generación.

Ya que para este caso, en un sistema de juego en la nube no se realizará una comunicación entre sistemas autónomos, ni mensajería o gestión de bases de datos. Y se quiere trabajar con un protocolo que sea implementado fácilmente, que sea libre, contenga buena documentación y ofrezca varios controladores para elegir… Se eligió OpenFlow. A continuación se explicará más afondo este protocolo y algunas de sus características principales:

12.2.2. OPENFLOW

Con el protocolo OpenFlow, una red puede ser gestionada como un todo, no como un número de dispositivos que gestionar individualmente, es el propio servidor el que dice a los switches dónde deben enviar los paquetes. Con esta tecnología, las decisiones que impliquen el movimiento de paquetes están centralizadas, por lo que la red puede ser programada independientemente de los switches. En un switch convencional, el reenvío de paquetes Datapath y el encaminamiento de alto nivel (control path) se realizan en el mismo dispositivo, sin embargo en los switches OpenFlow ambos se separan. Con OpenFlow, una parte del datapath reside en el mismo switch, pero es un controlador el que realiza las decisiones de encaminamiento de alto nivel. Ambos elementos se comunican por medio del protocolo OpenFlow.

SWITCH OPENFLOW

La mayoría de los switches Ethernet contienen tablas de flujos (Flow-Tables), que trabajan para implementar firewalls, NAT, QoS y recolectar estadísticas. Aunque las Tablas de Flujos que maneja cada fabricante son propias, se han aprovechado características observadas y comunes a todas ellas. Precisamente eso es lo que ofrece OpenFlow, un protocolo abierto para poder programar las tablas de flujo en diferentes switches y routers. Los administradores de la red solo necesitarían dividir el tráfico entre producción y el dedicado a la investigación. De esta manera, se conseguiría experimentar con nuevos protocolos, nuevos modelos de seguridad, esquemas de direccionamiento, incluso alternativas para IP y en definitiva una innovación mayor. Visto de esta manera OpenFlow podría ser una generalización de las VLAN´s. El tráfico de producción no se vería afectado ya que está aislado y se procesaría de la misma manera que se ha estado realizando hasta hoy día. Las acciones que pueden soportar los switches OpenFlow son extensibles, si bien es necesario tener unas características mínimas y comunes a todos ellos.

PARTES DE UN SWITCH OPENFLOW

Un switch OpenFlow consiste en al menos tres partes:

- **Tabla de flujos:** con una acción asociada a cada entrada de la tabla, indicando al switch cómo debe procesar ese flujo
- Canal seguro que conecte el switch a un proceso de control remoto (controlador), permitiendo comandos y paquetes se puedan enviar entre el switch y el controlador usando el protocolo OpenFlow
- **Controlador:** Un controlador añade y elimina entradas en la tabla de flujos para poder permitir los experimentos.

Un controlador estático podría ser una simple aplicación funcionando en un PC que añadirá estáticamente flujos y que puede interconectarse a su vez a una serie de ordenadores con funciones de analizadores de tráfico. Existen también controladores más sofisticados que dinámicamente añaden/eliminan flujos. De esta manera, un Administrador de red será capaz de gestionar todos los flujos y cómo serán procesados. Un controlador de estas características podría ser soportado por varios investigadores, cada uno con diferentes cuentas y permisos, permitiendo que todos ellos puedan realizar múltiples experimentos con diferentes conjuntos de flujos. El protocolo del controlador del switch trabaja sobra la capa TLS (Transport Layer Security) o conexiones TCP sin seguridad.

TIPOS DE SWITCHES

Existen dos tipos de switches: aquellos switches OpenFlow-only (solo OpenFlow) que no son compatibles con las capas 2 y 3 convencionales y los híbridos.

- **Switches OpenFlow-only** Un switch que soporta OpenFlow, contiene múltiples tablas de flujos, y cada flujo contiene múltiples entradas de flujo. El procesado define cómo los paquetes interactuarán con estas tablas de flujos. Requiere que tenga al menos una tabla de flujo y opcionalmente más. Cuantas menos entradas más simplificado será el procesado. La figura muestra un ejemplo de un switch OpenFlow:

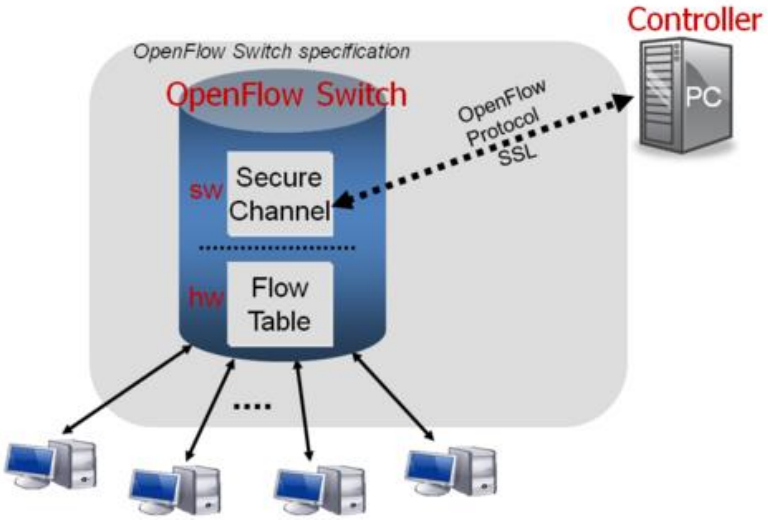


Ilustración 12: Switch OpenFlow

En este contexto los flujos están ampliamente definidos, solo están limitados por la capacidades de una determinada implementación de la Tabla de Flujo. Por ejemplo un flujo puede ser una conexión TCP, o todos los paquetes de una determinada dirección MAC o IP, todos los paquetes con la misma etiqueta VLAN, o todos los paquetes de un mismo puerto del switch.

- **Switches OpenFlow-híbridos:** Switches que soportan tanto la operación OpenFlow como el Switching Ethernet convencional. Operaciones habituales pueden ser: Switching Ethernet de Capa 2, aislamiento de tráfico con VLAN, nivel 3 o de routing (IPV4, IPV6), listas de acceso o QoS. Estos switches deberán proveer un mecanismo de clasificación fuera de OpenFlow que enrute el tráfico o bien ser procesado por OpenFlow. Por ejemplo, un switch deberá usar una etiqueta VLAN o puerto de entrada para decidir si se procesa el paquete de una manera u otra o debe redirigirlos hacia el procesador OpenFlow. Este mecanismo de clasificación sale fuera del ámbito de la especificación. El protocolo OpenFlow permite que el switch sea controlado por dos o más controladores para incrementar el rendimiento y la robustez del sistema. A continuación se muestra una posible configuración de este tipo de switches, en el que todas las tablas de flujos se gestionan por el mismo controlador.

FUNCIONAMIENTO DE OPENFLOW

Los switches tradicionales usan STP, SPB, TRILL para determinar cómo se reenvían los paquetes. OpenFlow traslada esta decisión de reenvío de los switches a los controladores, típicamente un servidor o una estación de trabajo. Una aplicación de gestión se ejecutará en las interfaces del controlador que une todos los switches en la red, facilitando la configuración de caminos de reenvío que utilizarán todo el ancho de banda disponible. La especificación define el protocolo entre el controlador y los switches y un conjunto de operaciones que se pueden realizar entre ellos. Las instrucciones de reenvío se basan en el flujo, que consiste en que todos los paquetes comparten una serie de características comunes. Existen infinidad de parámetros que pueden especificarse para definir un flujo. Entre los posibles criterios podemos incluir los puertos por donde se reciben

los paquetes cuando llegan, el puerto Ethernet de origen, la etiqueta VLAN, el destino Ethernet o el puerto IP y otro número de características de los paquetes. Un nuevo flujo se debe crear cuando un paquete que llega no encuentra ninguna coincidencia con ninguna entrada de la tabla. El switch debería tener configurado un descartado de paquetes para el flujo que no haya sido definido, pero en la mayoría de los casos, el paquete será enviado al controlador. El controlador entonces define un nuevo flujo para ese paquete y crea una o más entradas para la tabla. Éste envía la entrada o entradas al switch para que sean añadidas a las tablas de flujo. Finalmente, el paquete se envía de vuelta al switch para ser procesado con las nuevas entradas creadas.

FLUJOS

Cada flujo de entrada tiene asociada una acción simple. Las tres básicas son:

- Reenvío de flujo de paquetes a un puerto dado (o puertos). Esto permite que los paquetes ser encaminados a través de la red. En la mayoría de los switches sucede a la velocidad de la línea.
- Encapsulación y reenvío este flujo de paquetes al controlador. El paquete será enviado al Canal Seguro, donde se encapsula y se envía al controlador. Típicamente se usa para el primer paquete en un nuevo flujo, para que el controlador pueda decidir si el flujo debe ser añadido a la tabla de flujos. También se puede usar para reenviar todos los paquetes al controlador para que sean procesados.
- Descartar este flujo de paquetes. Puede ser usado por seguridad , para parar ataques de denegación de servicio o reducir el falso tráfico de descubrimiento broadcast desde los hosts finales.

BENEFICIOS

OpenFlow permite que las redes evolucionen dando a un controlador que funciona mediante software y lógicamente centralizado, el poder de modificar el comportamiento de los dispositivos de red a través de un conjunto de instrucciones de reenvío bien definidas. Modificando en cierto modo la red, los beneficios de OpenFlow-basado en SDN genera los siguientes beneficios:

- Crea flexibilidad en como la red se usa, se opera en ella y se vende. El software que lo controla puede ser escrito por empresas y proveedores de servicio usando un entorno de software común.
- Los operadores de red pueden implementar características que quieran en el software que controlan, en vez de tener que esperar que los fabricantes los ponga en marcha en sus propios productos.
- Reduce el gasto de operación, lo que conlleva menos errores y menor inactividad (frente a fallos) ya que permite la configuración automatizada de la red y reduce la configuración manual.
- Habilita la virtualización de la red y por lo tanto la integración de la Red con la Informática y el almacenamiento. Permite que la operación de las TIC sea controlada de manera más brillante con un solo punto de vista y con las mismas herramientas.
- Se puede integrar fácilmente con la informática para los recursos de gestión y mantenimiento OAM
- Reúne requisitos beneficiosos para las empresas

- Al tratarse de una manera estándar de transmitir la información de las Tablas de Flujos a los dispositivos de red, acoge a mercados abierto y diversos.

Debido a todos estos beneficios que se han creado, OpenFlow ha sido usado en ámbitos tan diversos como los centros de datos a gran escala, los centros de datos de empresas, proveedores públicos y privados de servicios en la red, redes de telecomunicaciones, redes de conmutación de circuitos, redes ópticas. También se está utilizando para variedad de servicios de virtualizacion de redes, seguridad, control de acceso para balanceo de carga, Ingeniería de Tráfico y gestión de la energía.

12.2.3. CONTROLADOR DE SDN

En las SDN, es el controlador central el que dicta el comportamiento general de la red a partir de los requerimientos de las aplicaciones. A continuación se hará una comparación entre algunos controladores de código abierto y comerciales existentes.

12.2.3.1. CONTROLADORES DE CÓDIGO ABIERTO

En la **Tabla 2**: Algunas características de controladores SDN se muestran algunas características de los controladores: Beacon, Floodlight, NOX, POX, Ryu, Trema y OpenDayLight (ODL).

	Beacon	Floodlight	NOX	POX	Trema	Ryu	ODL
Soporte OpenFlow	OF v1.0	OF v1.0	OF v1.0	OF v1.0	OF v1.3	OF v1.0, v1.2, v1.3 y extensiones Nicira	OF v1.0
Virtualización	Mininet y Open vSwitch	Mininet y Open vSwitch	Mininet y Open vSwitch	Mininet y Open vSwitch	Construcción de una herramienta virtual de simulación	Mininet y Open vSwitch	Mininet y Open vSwitch
Lenguaje de desarrollo	Java	Java	C++	Python	Rudy/C	Python	Java
Provee REST API	No	Si	No	No	Si (Básica)	Si (Básica)	Si
Interfaz Gráfica	Web	Web	Python+, QT4	Python+, QT4, Web	No	Web	Web
Soporte de plataformas	Linux, Mac OS, Windows y Android para moviles	Linux, Mac OS, Windows	Linux	Linux, Mac OS, Windows	Linux	Linux	Linux, Mac OS, Windows
Soporte de OpenStack	No	Si	No	No	Si	Si	Si
Multiprocesos	Si	Si	Si	No	Si	No	Si
Código Abierto	Si	Si	Si	Si	Si	Si	Si
Tiempo en el mercado	4 años	2 años	6 años	1 años	2 años	1 años	5 meses
Documentación	Buena	Buena	Media	Pobre	Media	Media	Media

Tabla 2: Algunas características de controladores SDN

Beacon: Es un software para la programación de reglas de flujo en controladores SDN escrito en Java, multiplataforma y modular. Posee las siguientes características:

- **Estable:** Este software se considera estable ya que ha sido probado en varios proyectos de investigación y en Data Centers sin sufrir ninguna interrupción en sus servicios por largos periodos de tiempo.
- **Multiplataforma:** Gracias a las características de Java de poder correr sobre cualquier dispositivo debido a que se ejecuta sobre una máquina virtual propia de Java denominada JVM (Java Virtual Machine).
- **Código Abierto:** El software se encuentra en el mercado bajo la licencia GPLv2 y la licencia de la Universidad de Stanford FOSS2 v1.0.
- Rápido funcionamiento: debido a la utilización de múltiples hilos de ejecución. https://openflow.stanford.edu/display/Beacon/Home

Floodlight: Es un software controlador SDN también escrito en Java, que corre sobre una máquina virtual de Java JVM. El software se caracteriza por:

- Trabajar con OpenFlow tanto en conmutadores físicos como virtuales.
- Es un controlador basado en módulos que permiten obtener ciertas funcionalidades como ruteo, conmutación, etc. El módulo de conmutación se denomina Learning Switch.
- Se puede instalar en Ubuntu 10.04 o superior o MacOS 10.6 o superior. •
- Para su funcionamiento se requiere la Java Developement Kit (JDK) y ANT de Apache, que es una herramienta usada en programación para realizar tareas repetitivas en Java.
- Permite tener una isla OpenFlow, es decir una agrupación de dispositivos que entienden este protocolo, conectado a una red tradicional que no entiende OpenFlow. http://www.projectfloodlight.org/floodlight/

NOX: Fue fabricado por Niciara Networks y desarrollado a la par que el protocolo OpenFlow. Se trata del primer controlador SDN OpenSource. A continuación se muestran las principales características de NOX:

- Incluye una API en C++.
- Posee una entrada y salida rápida y asincrónica, es decir que al software controlador puede ingresar y salir un gran flujo de datos en cualquier instante de tiempo.
- Está diseñado para ser instalado y configurado sobre sistemas operativos Linux, sobre todo Ubuntu en sus versiones 11.0 y 12.04, aunque también puede implementarse fácilmente sobre Debian, Red Hat o CentOS.

Actualmente se encuentran disponibles dos versiones de NOX:

- **NOX Classic:** Es una versión antigua que ofrece soporte para Python y C++. Sin embargo, la versión fue descontinuada y ya no existe desarrollo en esta línea. No se recomienda implementar esta versión, aunque cuenta con un rango de módulos más amplio que la nueva versión. La versión activa de NOX se llama “verity”.
- **NOX:** Es la nueva versión de este software. Ofrece soporte exclusivamente para C++. Sin embargo, ofrece un rango de aplicaciones reducido en comparación a la versión clásica. Su funcionamiento es más rápido y cuenta con un código más depurado. Esta versión se encuentra en continuo desarrollo.

  POX es una herramienta muy similar a NOX, pero sólo usa el lenguaje Python para programar las reglas que le indican al controlador qué acciones llevar a cabo, mientras que NOX puede usar tanto Python como C++. Otra iniciativa de controlador SDN basado en NOX de la Universidad Tsukuba es Jaxon. El objeto de este controlador es servir de puente entre aplicaciones desarrolladas en Java y el controlador NOX

  http://www.noxrepo.org/

POX: Presenta las siguientes características:

- Posee una interfaz OpenFlow denominada “Pythonic” por encontrarse en el ambiente de desarrollo de Python.

- Puede correr en cualquier plataforma: Linux, Windows, Mac OS o cualquier otra, ya que se puede combinar con “PyPy”, que es un entorno de ejecución de programas escritos en Python. Este entorno de ejecución tiene implementaciones básicas de comandos de bajo nivel, pero también puede ejecutar comandos de alto nivel.
- Soporta la misma GUI (Interfaz Gráfica de Usuario) que NOX y tiene la misma visualización.
- Tiene un mejor rendimiento que los programas NOX escritos en Python, por lo que ha reemplazado a la versión descontinuada NOX Classic. Si se desea programar el controlador en Python, la herramienta más efectiva y de mayor rendimiento es POX.
- Posee ciertos componentes que pueden ser reutilizables para la selección del mejor camino, descubrimiento de topologías, etc.

Trema: Es un framework para programar un controlador SDN, a gusto del programador, en los lenguajes Ruby y C. Este entorno de programación genera un fichero de configuración que después puede ser ejecutado por Trema para comportarse como un controlador SDN. http://trema.github.io/trema/

FlowER: Es un controlador SDN OpenFlow escrito en lenguaje de programación Erlang. Es un controlador menos conocido que nació en 2012 basado en un lenguaje de programación concurrente y un sistema de ejecución que incluye una máquina virtual y biblioteca. 3.2.6.

NodeFlow :Es el controlador SDN escrito en JavaScript creado inicialmente por Cisco. Este controlador proveía una librería asíncrona sobre JavaScript para programar las redes SDN. La idea de usar JavaScript era descargar de CPU al controlador. Este controlador ha quedado descontinuado por la decisión de Cisco de abordar las redes SDN desde un punto de vista no OpenFlow. 3.2.7.

Maestro: Se define como un sistema operativo de orquestación de aplicaciones de control de red. En realidad, es una plataforma de controlador SDN OpenFlow. Ha sido diseñado en Java aunque requiere ANT. Es multi-thread, es decir explota la capacidad de CPU de los procesadores multi-core y se distribuye bajo licencia GNU.
https://code.google.com/p/maestro-platform/

OpenDayLight: Es un proyecto de Código Abierto (Open Source) el cual tiene como objetivo acelerar y acrecentar la difusión de la innovación en el diseño e implementación de un estándar abierto y transparente de Redes Definidas por Software (RDS), es decir, Software-Defined Networking (SDN). Actualmente el proyecto tiene el apoyo de grandes y reconocidas compañías como: Networks, Brocade, Cisco, Citrix, Ericsson, IBM, Juniper Networks, Microsoft, NEC, RedHat,, VMware

OpenDayLight pretende emular lo que Hadoop ha hecho en el mercado de BigData y lo que OpenStack hace en el Cloud Computing. En otras palabras, convertirse en una plataforma abierta que utilicen todas las empresas, evitando que las aplicaciones

(programas) privativas (propietarios) restrinjan el crecimiento del mercado y al mismo tiempo se reduzcan los costes de desarrollo.

OpenDayLight pretende convertirse en una plataforma SDN común y abierta que incorpore áreas como la plataforma del controlador o los protocolos de las aplicaciones de red, las interfaces de usuario, los switches virtuales o las interfaces físicos del dispositivo.

La principal ventaja de esta propuesta es que elimina las barreras de adopción, ya que algunas organizaciones no quieren comprometerse con un fabricante especifico que termine bloqueando su futuro. Al ser una plataforma común, las empresas podrán optar por tecnologías de fabricantes diversos que serán interoperables.

En concreto Big Switch Networks, Brocade, Cisco, Citrix, Ericsson, IBM, Juniper Networks, Microsoft, NEC, RedHat y VMware son los miembros Platinum y Gold (fundadores principales) del proyecto, los cuales donarán software y recursos de ingeniería para este proyecto Open Source y ayudarán a definir la plataforma SDN abierta, según han informado oficialmente.

En definitivas cuentas OpenDayLight es un controlador de OpenFlow, el cual a su vez es un protocolo que permite a un servidor decirle a los conmutadores de red adónde enviar paquetes. En cada red tradicional, cada conmutador tiene software privativo (propietario) que le especifica las reglas por cumplir (qué hacer). Con OpenFlow se centralizan las decisiones de migración de paquetes, de modo que la red se puede programar independiente de los conmutadores individuales y equipo del centro de datos.

Las características principales de OpenDayLight son:

- Tiene una API abierta.
- Emplea un enfoque basado en modelos para describir la red, las funciones a desempeñar en ella, sus estados y condiciones resultantes.
- **Microservicios:** Al compartir estructuras de datos Yang en una infraestructura de almacenamiento de datos común y de mensajería, OpenDaylight permite la implementación de servicios pequeños que se combinan para resolver problemas más complejos. Cualquier aplicación o función puede ser agrupada en un servicio que está a continuación de otro y así sucesivamente hasta cargarse en el controlador. Los servicios pueden ser configurados y encadenados juntos en cualquier número de formas para que coincida con las necesidades dentro de la red.
- Al basarse en microservicios se pueden instalar solamente los protocolos y servicios que se requieran
- Soporte para un amplio conjunto de protocolos SDN, tanto tradicionales como emergentes, mejorando la capacidad de programación de las redes modernas y la resolución de una amplia gama de necesidades de los usuarios. Por ejemplo, la plataforma soporta OpenFlow y extensiones OpenFlow como tabla de tipos de patrones (TTP), así como los protocolos tradicionales, incluyendo

NETCONF, BGP / PCEP y CAPWAP. Además, las interfaces de EAD con OpenStack y Open vSwitch a través de OVSDB.

- La comunidad de OpenDayLight aporta información o mejoras, que posteriormente pueden quedar en una reléase dependiendo de sus características.
- Tiene mayor seguridad y apoyo por proveedores, fabricantes y empresas importantes en el mundo de TI.

12.2.3.2. CONTROLADORES COMERCIALES

EL CONTROLADOR DE INFRAESTRUCTURA DE POLÍTICAS DE APLICACIONES (APIC) DE CISCO: Se considera un sistema distribuido implementado como un clúster de controladores. Dentro de la Infraestructura Centrada en Aplicaciones (ACI) de Cisco, APIC actúa como un único punto de control. Proporciona una API central, un repositorio central de datos globales y un repositorio de datos de políticas. El controlador puede aplicar automáticamente políticas y funciones de red centradas en las aplicaciones con el aprovisionamiento declarativo basado en modelos de datos.

El objetivo principal de APIC es proveer de autoridad política y de mecanismos de resolución de políticas a los dispositivos Cisco ACI para optimizar el desempeño de las aplicaciones y la eficiencia de la red. Se dice que la automatización se produce como consecuencia directa de la resolución de políticas y de la prestación de sus efectos sobre el tejido ACI.

Hay nodos de switch spine y leaf ACI con los cuales se comunica APIC. De esta manera, es capaz de distribuir las políticas y entregar varias funciones administrativas. Al no tener el controlador directamente involucrado en el envío de plano de datos, un cluster no perderá ninguna funcionalidad de centro de datos si hay una desconexión de los componentes de APIC.

EL CONTROLADOR SDN VIRTUAL APPLICATION NETWORKS (VAN) DE HP: Controla las decisiones de política y reenvío en una red SDN que ejecuta switches habilitados para OpenFlow en el centro de datos o en la infraestructura del campus. HP también está trabajando con Verizon e Intel para desarrollar una app utilizada para proporcionar ancho de banda WAN utilizando el controlador VAN.

El controlador también permite el control centralizado y la automatización. Dentro de un entorno HP SDN, el controlador VAN ofrece integración entre la red y el sistema de negocios. Utiliza interfaces programables que permiten la orquestación de aplicaciones y la automatización de las funciones de red. El controlador también proporciona un control de la red, incluyendo funciones tales como la detección de la topología de red.

El controlador VAN también puede agruparse, lo que permite a un controlador hacerse cargo de las funciones de otro si uno falla. En lo que respecta a la seguridad, el controlador

utiliza métodos de autenticación y autorización. A su vez, las aplicaciones SDN pueden interactuar con el controlador, mientras que las aplicaciones no autorizadas no pueden tener acceso a la red. Las conexiones hacia el sur entre los switches OpenFlow y el controlador de HP también están aseguradas y encriptadas.

EL CONTROLADOR PROGRAMMABLE FLOW PF6800 DE NEC: Está en el centro del tejido de red basado en OpenFlow Programmable Flow de NEC. Proporciona un punto de control para las redes físicas y de gestión para las redes virtuales y físicas. El controlador se considera programable, así como estandarizado. Se integra tanto con OpenStack, como con Microsoft System Center Virtual Machine Manager para una gestión de red y orquestación añadidos. El controlador también incluye la tecnología de red de inquilino virtual de NEC, que permite redes aisladas, con varios inquilinos.

EL CONTROLADOR DE SERVICIOS VIRTUALIZADOS (VSC) DE NUAGE NETWORKS: Permite la visión completa de una red por inquilino y las topologías de servicio, mientras externaliza plantillas de servicios de la red definidos a través del Directorio de Servicios Virtualizados de Nuage Networks. El directorio de servicios es un motor de políticas que utiliza el análisis y las reglas de red para autorizar permisos basados en roles. El VSC envía mensajes usando esas reglas a la plataforma de conmutación y ruteo virtual de Nuage. La plataforma detecta ya sea la creación o supresión de una máquina virtual y luego le pregunta al controlador SDN si hay una política instalada para ese inquilino. Si existe una regla, la conectividad de red se establece inmediatamente.

EL CONTROLADOR NSX DE VMWARE: Se considera un sistema de gestión de estado distribuido que controla las redes virtuales y superpone los túneles de transporte. Es el punto de control central para todos los switches lógicos dentro de una red. El controlador mantiene la información de las máquinas virtuales, hosts, switches lógicos y VXLANs, mientras usa APIs en dirección norte para hablar con las aplicaciones.

Cuando se trabaja con el controlador, las aplicaciones comunican lo que necesitan, y el controlador programa todos los switches virtuales bajo el control NSX en dirección sur para cumplir con esos requisitos. El controlador podría ejecutarse de dos maneras dentro de NSX: ya sea como un clúster apartado de máquinas virtuales en un entorno vSphere, o en los aparatos físicos para aquellos con hipervisores mixtos.

Anteriormente se mostraron los controladores de código abierto y comerciales disponibles con sus características principales, pero para saber qué controlador conviene más para cualquier tipo de sistema, existen herramientas para la evaluación de controladores SDN, entre las cuales se destacan dos: Cbench y Hcprobe.

Cbench (Controller Benchmarker) es una herramienta para monitorear controladores OpenFlow a través de la generación de eventos. Cbench emula un número configurable de switches OpenFlow los cuales se comunican con un controlador OpenFlow para medir diferentes aspectos del

rendimiento y latencia del mismo. Su funcionamiento básico consiste en que cada switch emulado envía un número configurable de mensajes de nuevos flujos (mensajes OpenFlow packet-in) al controlador OpenFlow, espera por las respuestas apropiadas de configuración de flujos (mensajes OpenFlow packet-out o mensajes OpenFlow de modificación de flujos flow-mod) y registra estadísticas de la diferencia de tiempo entre las solicitudes y las respuestas, así como otras métricas de desempeño.

Hcprobe, escrita en Haskell, permite crear con facilidad escenarios para pruebas de control SDN. Es capaz de simular un gran número de switch y host conectados a un controlador. Empleando Hcprobe se pueden analizar varios índices de operación del controlador de forma flexible. Con esta herramienta se pueden especificar patrones para generar mensajes OpenFlow (incluidos los malformados) y establecer perfiles de tráfico, entre otros. Sus características principales son:

- Generación de paquetes OpenFlow y de tablas de red
- Implementación en un lenguaje de alto nivel, lo que hace que sea más fácil de extender
- Existencia de una API para el diseño de pruebas personalizadas
- Un lenguaje específico de dominio embebido (EDSL) para la creación de pruebas.
- Proporciona un framework para la creación de varios casos de uso para estudiar el comportamiento de los controladores OpenFlow a través del procesamiento de diferentes tipos de mensajes. Con esta herramienta se puede generar todo tipo de mensajes OpenFlow switch-controlador y reproducir diferentes escenarios de comunicación entre ellos.

Open Networking Foundation en convenio con Standford hicieron un estudio previo comparando el desempeño de los controladores de código abierto anteriormente mencionados, con los siguientes resultados:

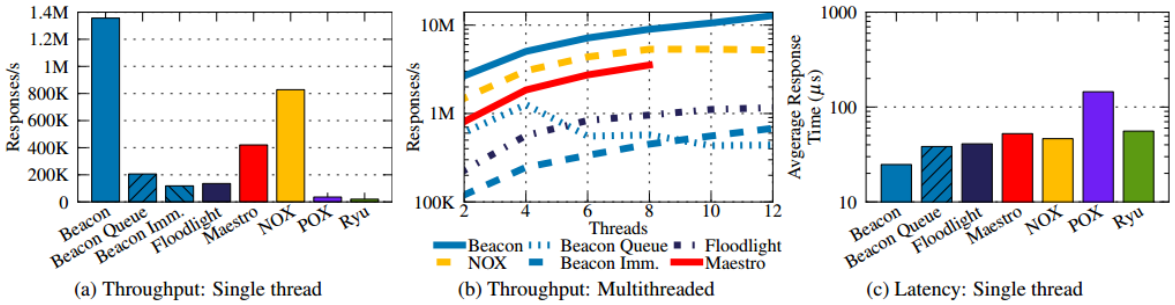


Ilustración 13: Comparación de controladores

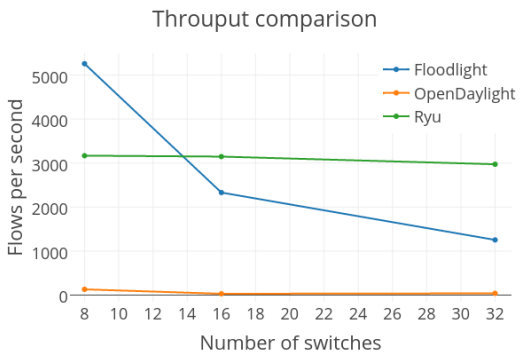


Ilustración 14: Comparación de controladores 2

Como se puede apreciar en las gráficas los controladores con mejor rendimiento y estabilidad son Beacon, Ryu, Maestro, POX, FloodLight y OpenDayLight. Pero los primeros tres no tienen soporte y no están actualmente en desarrollo, POX aunque no está actualizado su lenguaje proporciona una gran flexibilidad para poder incorporar algoritmos al controlador.

Debido a que POX no tiene la suficiente documentación, y a que OpenDayLight tiene más seguridad, con la posibilidad de implementar microservicios, junto con una documentación más extensa gracias a la colaboración de todas las compañías que han aportado en su desarrollo, además de que posee muchas posibilidades de configuración (por ejemplo OpenStack)… Se elegirá para el sistema de Cloud Gaming: OpenDayLight.

12.3. IMPLEMENTACIÓN DE GAMINGANYWHERE EN GENI

Por las características detalladas en el punto **10.3.8**, se optó por usar GENI, además que gracias a este laboratorio de redes se puede tener la seguridad de que los experimentos instanciados, tendrán recursos reales y los resultados serán más cercanos a la realidad.

En GENI se diseñó la siguiente red, la cual fue usada como escenario para implementar la plataforma de juego en la nube GamingAnywhere en una SDN con OpenDayLight como controlador:

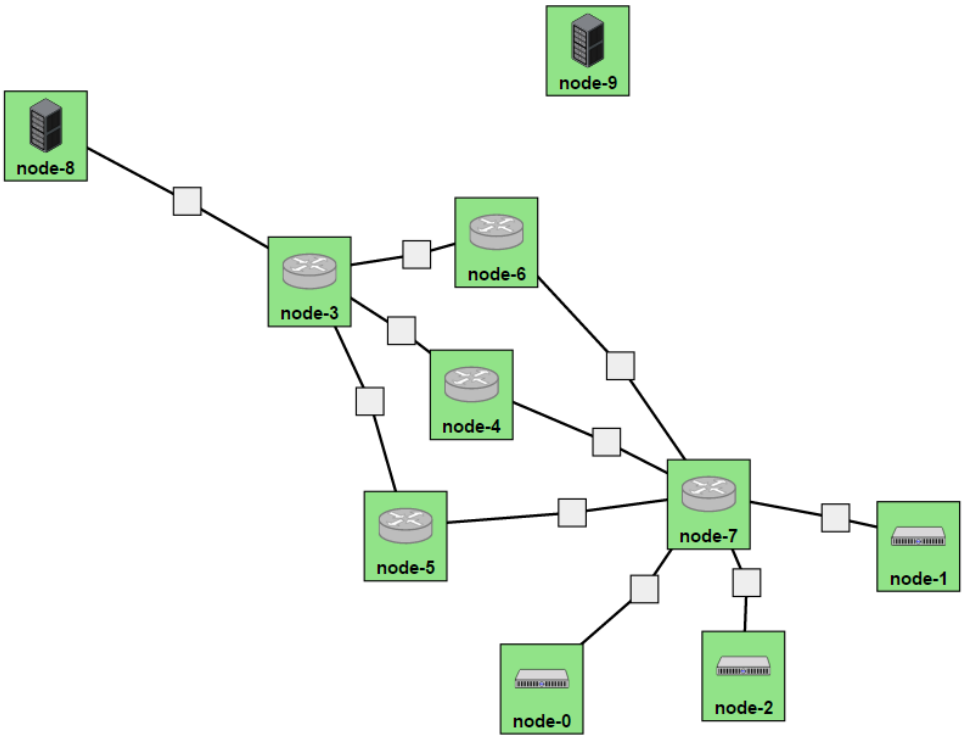


Ilustración 15: Diseño de la red usada para el sistema de Cloud Gaming

Dónde:

- Los nodos 3, 4, 5, 6 y 7 son switches OpenFlow.
- Los nodos 0,1 y 2 son hosts,
- El nodo 8 es un servidor
- El nodo 9 actúa como controlador con OpenDayLight previamente instalado.

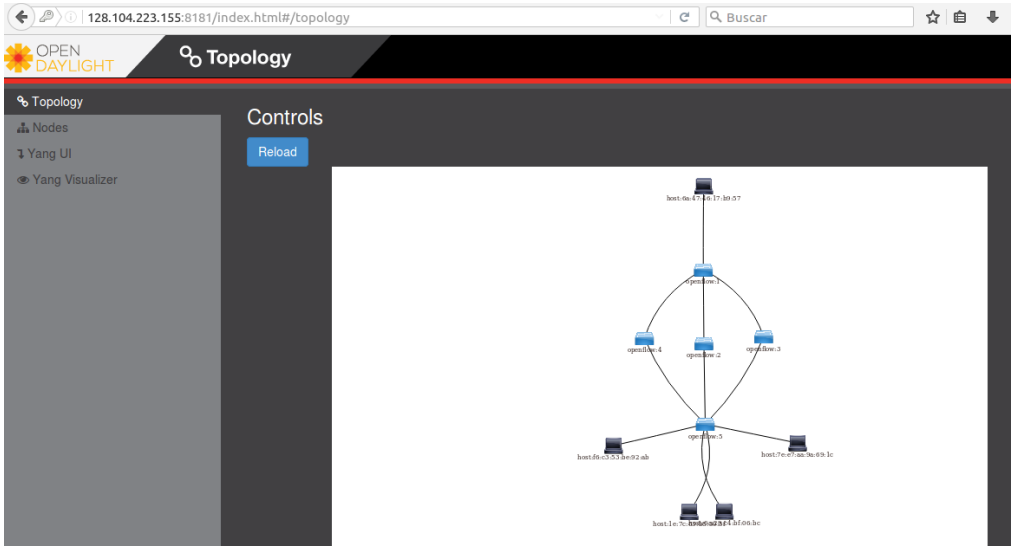
Para implementar GamingAnywhere en la red anterior configurada en GENI, se siguieron los siguientes pasos:

- Probar la red con Mininet para saber sí puede funcionar correctamente (ver final del **Anexo 2**).
- Instalar GamingAnywhere en el nodo 8 y algún host (**ver Anexo 1**).
- Instalar OpenDayLight en el nodo 9 (**ver Anexo 2**).
- Configurar los switches OpenFlow 3, 4, 5, 6 y 7 para que se comuniquen con el controlador del nodo 9 (**ver Anexo 3**)
- Crear una conexión una conexión entre el nodo 8 y algún host que tenga la plataforma instalada previamente con los pasos descritos en el **Anexo 1.**

13. RESULTADOS

Antes de la implementación del sistema de juego en la nube se publicó un artículo en la revista Entre Ciencia e Ingeniería, ISSN 1909-8367 Año 10 No. 20 - Segundo Semestre de 2016, página 82 – 9, titulado “Juego en la nube: un estado del arte”

Así mismo se implementó la plataforma de juego en la nube en GENI con la ayuda de CloudLab (una extensión de GENI), obteniendo conectividad con el controlador, como se muestra a continuación:



Finalmente se transmitió el juego Minecraft con 34 FPS desde el servidor, y en el cliente se pudo visualizar con 32 FPS y un retardo de 126 ms.

14. CONCLUSIONES

Se puede implementar una plataforma de juego en la nube en una red definida por software con el controlador OpenDayLight en GENI.

Así mismo la experiencia del usuario en cuanto a frames por segundo es positiva ya que el cliente puede acceder al juego fluidamente, pero con un retardo superior a 120 ms, lo cual es negativo a la hora de acceder a juegos multijugador.

El lograr una buena experiencia para el usuario sin inversión excesiva de hardware y retardos, son problemas difíciles, especialmente este último ya que si el usuario tiene un retraso de los comandos u órdenes con las cuales interactuará con el juego, podría provocar que éste no fuera jugable, comprometiendo la calidad del servicio.

Pese a lo anterior el juego en la nube abre nuevos campos de investigación y ofrece un modelo de servicio en el cual los proveedores de la nube podrán ofrecer aplicativos para cualquier tipo de dispositivo con conexión a Internet, de modo que la implementación de esta tecnología serviría para ofrecer no solamente videojuegos sino programas de gran uso de unidades de procesamiento gráfico, como los de edición de video, desarrollo de videojuegos, entre otros, creando de esta manera modelos de negocio dónde no se vende solamente software sino una forma de ejecutarlo por medio de la nube.

GaaS se considera la próxima generación de la industria del juego, en un futuro los jugadores serán capaces de acceder a contenido de juegos de forma globalizada a través de dispositivos con capacidades diferentes, mitigando de esta forma la necesidad de equipos de alto procesamiento gráfico o de CPU.

15. BIBLIOGRAFÍA

[1] E. Piehl. (2016, May). *G-Cluster Global*. Available: http://www.gcluster.com/eng/

[2] T. Lynas. 2005, April). Club iT and Alcatel Bring G-cluster Gaming-on-Demand to Cyprus Telecommunications Authority Interactive TV Service as Part of Alcatel's Open Media Platform. Available: http://www.businesswire.com/news/home/20050415005167/en/Club-Alcatel-Bring-G-cluster-Gaming-on-Demand-Cyprus-Telecommunications

[3] M. Maal. (2012, January). *G-cluster Global receives the support of the investment funds of SFR* Available: http://www.gcluster.com/eng/pdf/20120119_001_E.pdf

[4] GAIKAI. (2016). *The Journey Continues*. Available: https://www.gaikai.com/#!/history

[5] K.-T. Chen, Y.-C. Chang, P.-H. Tseng, C.-Y. Huang, and C.-L. Lei, "Measuring the latency of cloud gaming systems," presented at the Proceedings of the 19th ACM international conference on Multimedia, Scottsdale, Arizona, USA, 2011.

[6] C.-Y. Huang, C.-H. Hsu, Y.-C. Chang, and K.-T. Chen, "GamingAnywhere: an open cloud gaming system," presented at the Proceedings of the 4th ACM Multimedia Systems Conference, Oslo, Norway, 2013.

[7] A. Khalifa, "Tele-Rehabilitation Games on the Cloud: A Survey and a Vision," *American Journal of Computer Science and Engineering Survey (AJCSES),* vol. 3, pp. 143-151, 2015.

[8] H. J. Hong, D. Y. Chen, C. Y. Huang, K. T. Chen, and C. H. Hsu, "Placing Virtual Machines to Optimize Cloud Gaming Experience," *IEEE Transactions on Cloud Computing,* vol. 3, pp. 42-53, 2015.

[9] K. T. Chen, Y. C. Chang, H. J. Hsu, D. Y. Chen, C. Y. Huang, and C. H. Hsu, "On the Quality of Service of Cloud Gaming Systems," *IEEE Transactions on Multimedia,* vol. 16, pp. 480-495, 2014.

[10] H. Guan, J. Yao, Z. Qi, and R. Wang, "Energy-Efficient SLA Guarantees for Virtualized GPU in Cloud Gaming," *IEEE Transactions on Parallel and Distributed Systems,* vol. 26, pp. 2434-2443, 2015.

[11] S. P. Chuah, C. Yuen, and N. M. Cheung, "Cloud gaming: a green solution to massive multiplayer online games," *IEEE Wireless Communications,* vol. 21, pp. 78-87, 2014.

[12] Y. Li, X. Tang, and W. Cai, "Play Request Dispatching for Efficient Virtual Machine Usage in Cloud Gaming," *IEEE Transactions on Circuits and Systems for Video Technology,* vol. 25, pp. 2052-2063, 2015.

[13] Q. Hou, C. Qiu, K. Mu, Q. Qi, and Y. Lu, "A Cloud Gaming System Based on NVIDIA GRID GPU," in *Distributed Computing and Applications to Business, Engineering and Science (DCABES), 2014 13th International Symposium on*, 2014, pp. 73-77.

[14] M. Amiri, H. A. Osman, S. Shirmohammadi, and M. Abdallah, "An SDN Controller for Delay and Jitter Reduction in Cloud Gaming," presented at the Proceedings of the 23rd ACM international conference on Multimedia, Brisbane, Australia, 2015.

[15] H. A. Osman, M. Amiri, S. Shirmohammadi, and M. Abdallah, "SDN-based game-aware network management for cloud gaming," in *Network and Systems Support for Games (NetGames), 2015 International Workshop on*, 2015, pp. 1-6.

[16] D. Kreutz, F. M. V. Ramos, P. E. Ver, x00Ed, ssimo, C. E. Rothenberg*, et al.*, "Software-Defined Networking: A Comprehensive Survey," *Proceedings of the IEEE,* vol. 103, pp. 14-76, 2015.

[17] P. Ligia Rodrigues, A. A. Shinoda, C. M. Schweitzer, and R. L. S. d. Oliveira, "Simulation in an SDN network scenario using the POX Controller," in *Communications and Computing (COLCOM), 2014 IEEE Colombian Conference on*, 2014, pp. 1-6.

[18] K. Teemu, inen, Y. Shan, M. Siekkinen, A. Yl, J.*, et al.*, "Virtual machines vs. containers in cloud gaming systems," in *Network and Systems Support for Games (NetGames), 2015 International Workshop on*, 2015, pp. 1-6.

[19] A. Ribi, x00E, and re, "Emulation of obsolete hardware in open source virtualization software," in *2010 8th IEEE International Conference on Industrial Informatics*, 2010, pp. 354-360.

[20] C. Anderson, "Docker [Software engineering]," *IEEE Software,* vol. 32, pp. 102-c3, 2015.

[21] S. Hykes. (2016). *Docker: Build, Ship, Run*. Available: https://www.docker.com/

[22] S. Shi, C.-H. Hsu, K. Nahrstedt, and R. Campbell, "Using graphics rendering contexts to enhance the real-time video coding for mobile cloud gaming," presented at the Proceedings of the 19th ACM international conference on Multimedia, Scottsdale, Arizona, USA, 2011.

[23] S. Wang and S. Dey, "Rendering Adaptation to Address Communication and Computation Constraints in Cloud Mobile Gaming," in *Global Telecommunications Conference (GLOBECOM 2010), 2010 IEEE*, 2010, pp. 1-6.

[24] C.-Y. Huang, Y.-L. Huang, Y.-H. Chi, K.-T. Chen, and C.-H. Hsu, "To Cloud or Not to Cloud: Measuring the Performance of Mobile Gaming," presented at the Proceedings of the 2nd Workshop on Mobile Gaming, Florence, Italy, 2015.

[25] Y. Lu, Y. Liu, and S. Dey, "Optimizing Cloud Mobile 3D Display Gaming user experience by asymmetric object of interest rendering," in *2015 IEEE International Conference on Communications (ICC)*, 2015, pp. 6842-6848.

[26] E. Mucci García, "Universidad Politécnica de Cataluña," in *El impacto de la Nube en la productividad de la PYME*, ed, 2010.

[27] R. . (2014, IETF. *Software-Defined Networking: A Perspective from within a Service Provider Environment*. Available: https://tools.ietf.org/pdf/rfc7149.pdf

[28] N. Corporation, "Ventajas de GRID para el juego en al nube," ed, 2015.

[29] B. Sinclair. (2009). *David Perry demos Gaikai game streaming*. Available: http://www.gamespot.com/articles/david-perry-demos-gaikai-game-streaming/1100-6212832/

[30] S. I. E. LLC. (2016). *Playstation now - PS Now Suscription for PS Games*. Available: https://www.playstation.com/en-us/explore/playstationnow/

[31] G. Streaming. (2016). *Stream high quality games directly to your TV*. Available: https://www.gamefly.com/#!/stream

[32] R. Gharsallaoui, M. Hamdi, and T. H. Kim, "A Comparative Study on Cloud Gaming Platforms," in *Control and Automation (CA), 2014 7th Conference on*, 2014, pp. 28-32.

[33] W. Cai, M. Chen, and V. C. M. Leung, "Toward Gaming as a Service," *IEEE Internet Computing,* vol. 18, pp. 12-18, 2014.

[34] G. D'Angelo, S. Ferretti, and M. Marzolla, "Cloud for Gaming," *arXiv preprint arXiv:1505.02435,* 2015.

[35] R. Shea, J. Liu, E. C. H. Ngai, and Y. Cui, "Cloud gaming: architecture and performance," *IEEE Network,* vol. 27, pp. 16-21, 2013.

[36] M. Semsarzadeh, A. Yassine, and S. Shirmohammadi, "Video Encoding Acceleration in Cloud Gaming," *IEEE Transactions on Circuits and Systems for Video Technology,* vol. 25, pp. 1975-1987, 2015.

[37] W. Cai, V. C. M. Leung, and M. Chen, "Next Generation Mobile Cloud Gaming," in *Service Oriented System Engineering (SOSE), 2013 IEEE 7th International Symposium on*, 2013, pp. 551-560.

[38] W. Cai, M. Chen, C. Zhou, V. C. M. Leung, and H. C. B. Chan, "Resource management for cognitive cloud gaming," in *2014 IEEE International Conference on Communications (ICC)*, 2014, pp. 3456-3461.

[39] L. Gadea, in *Revised and much faster, run your own high-end cloud gaming service on EC2!*, ed, 2015.

[40] C.-Y. Huang, C.-H. Hsu, and K.-T. Chen. (2014, Enero). GamingAnywhere: An Open Cloud Gaming System. *ACM Transactions on Multimedia Computing, Communications and Applications*. Available: http://mmnet.iis.sinica.edu.tw/pub/huang13_gaming_anywhere.pdf

ANEXOS

INSTALACIÓN DE GAMINGANYWHERE

El primer paso para la instalación de GamingAnywhere es descargar el paquete de la página oficial http://gaminganywhere.org/download.html, que contiene el proyecto, el código fuente, las dependencias, binarios precompilados y configuraciones de videojuegos en los que se ha probado antes la plataforma para facilitar su ejecución.

1. Sí el usuario usa Windows se descarga desde un navegador y posteriormente se descomprime. En caso de ser usuario de plataformas POSIX (Linux y Mac OS X) se descarga con el siguiente comando:

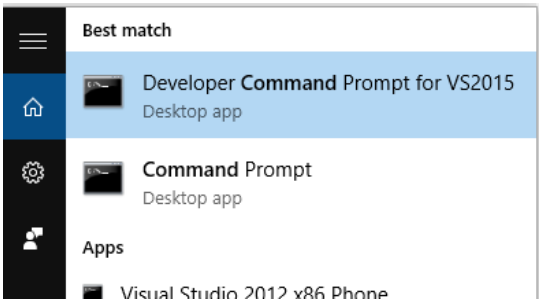
   wget http://gaminganywhere.org/dl/gaminganywhere-0.8.0-all-in-one.tar.bz2

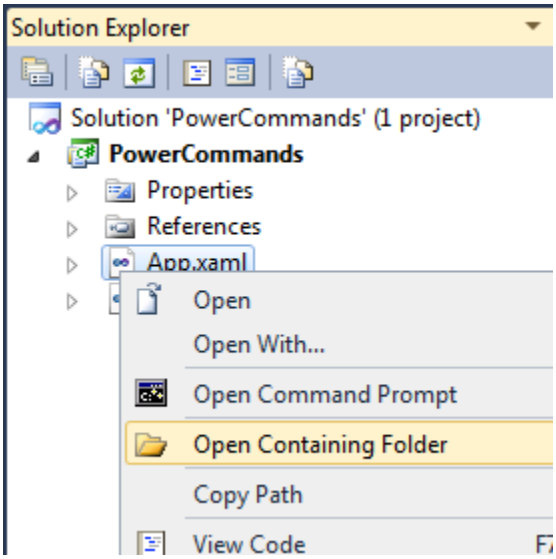
2. Posteriormente se desempaqueta de la siguiente manera:

   tar -xjvf gaminganywhere-0.8.0-all-in-one.tar.bz2

Para instalar la plataforma en Windows se requiere de Visual Studio (C++) 2010 o posterior, DirectX SDK y Microsoft SDK actualizados. Luego de ello se siguen estas instrucciones:

1. Se instalan las dependencias que requiere el programa ejecutando el archivo “install.cmd” que se ubica en la carpeta “deps.pkgs.win32”.
2. Verificar que DirectX SDK está instalado en “C:\Microsoft DirectX SDK”, de lo contrario hay que modificar la ruta en un bloc de notas en los archivos “ga/module/vsource-desktop/NMakefile.d3d” y “ga/server/event-driven/NMakefile”, donde al abrirlos se mostrará la ruta antigua, simplemente se cambiaría a la nueva ubicación de DirectX SDK.
3. Abrir la línea de comandos de Visual C++, aquí se muestran algunos ejemplos:





4. En la línea de comandos ingresar al directorio “ga”, y compilarlo con el comando “nmake /f NMakefile all”
5. Instalar GamingAnywhere estando en el mismo directorio anterior, con el comando “nmake /f NMakefile install”.
6. Sí se generó una carpeta llamada “bin” que contenga como archivos “ga-client” o “ga-server-*” la instalación fue exitosa.

Para instalar GamingAnywhere en plataformas de POSIX (Linux y Mac), se requiere un compilador de C++, junto con las siguientes librería:s libX11, libXext, libXtst, libfreetype6, libgl1-mesa, libglu1-mesa, libpulse, libasound2. Siguiendo las siguientes instrucciones:

1. Instalar todas las librerías requeridas con el siguiente comando (se muestra la sintaxis de Ubuntu):

   sudo apt-get install make cmake patch g++ build-essential libfreetype6-dev libogg-dev libsdl2-dev yasm cmake libX11-dev libXext-dev libXtst-dev libfreetype6-dev libgl1-mesa-dev libglu1-mesa-dev libpulse-dev libasound2-dev pkg-config lib32z1

   sudo apt-get install libswscale-dev libswresample libpostproc-dev libavdevice-dev libavfilter-dev libavcodec-dev libavformat-dev libavutil-dev libsdl2-ttf x264 libav-tools

   En caso de instalar una versión de 32 bits en un sistema operativo de 64 bits (no es necesario hacerlo, pero en caso de que se hiciera):

   sudo apt-get install make cmake patch g++ build-essential libfreetype6-dev libogg-dev:i386 libsdl2-dev:i386 yasm cmake libX11-dev:i386 libXext-dev:i386 libXtst-dev:i386 libgl1-mesa-dev:i386 libglu1-mesa-dev:i386 libpulse-dev:i386 libasound2-dev:i386 pkg-config lib32z1:i386

sudo apt-get install libswscale-dev:i386 libswresample:i386 libpostproc-dev:i386 libavdevice-dev:i386 libavfilter-dev:i386 libavcodec-dev:i386 libavformat-dev:i386 libavutil-dev:i386 libsdl2-ttf:i386 x264:i386 libav-tools:i386

2. Ingresar al directorio donde se tenga desempaquetado GamingAnywhere y verificar sí en el archivo "env-setup" se usarán las rutas por defecto, luego usar el comando "source":

   cd gaminganywhere-0.8.0
   source env-setup

3. Acceder al directorio "deps.src" y armar el paquete con el comando "make", luego ingresar al directorio "ga" y ejecutar el comando "make all":

   cd deps.src/
   make
   cd gaminganywhere-0.8.0/ga
   make all

   En caso de tener errores hasta este punto deberían aparecer las librerías que tienen conflicto con el programa, así que habría que instalarlas, de lo contrario no se siguió bien el paso 1.

4. Ya que estamos en "ga", sólo restaría instalar GamingAnywhere:

   make install

5. Sí se generó una carpeta llamada "bin" que contenga como archivos "ga-client" o "ga-server-*" la instalación fue exitosa.

```
frannier@host1:~/gaminganywhere-0.8.0/bin$ ls
config        ga-hook-gl.so        ga-hook-sdl.so          libga.so
FreeSans.ttf  ga-hook-pulse.so     ga-server-event-driven  mod
ga-client     ga-hook-sdlaudio.so  ga-server-periodic
```

6. Se recomienda modificar los siguientes archivos:

   sudo nano /etc/ssh/ssh_config
   Cambiar a "ForwardX11 yes"

   sudo nano /etc/ssh/sshd_config
   Cambiar a "X11Forwarding yes"

Para la ejecución de la plataforma se necesitan los siguientes prerrequisitos:

- En Linux hay que agregar "deps.posix/lib" en system-wide ld.so, además de modificar el archivo "bin/config/common/server-common.conf" con la resolución de nuestra pantalla y asegurarse de que ella es accesible con la extensión XTEST activa.

- En Windows hay que agregar “deps.win32/bin/*.dll” en el directorio de DLL del sistema o en “bin”.
- Estar en el directorio “bin”

En “bin” se podrá acceder a tres tipos de ejecutables, los cuales son:

- **ga-server-periodic:** Para hacer un streaming del escritorio completo al cliente. Con la siguiente sintaxis de ejecución:

  ga-server-periodic [16]
  e.g., $ ga-server config/server.x264+mp3.conf

  En caso de que querer transmitir solamente una ventana en el archivo de configuración se deben cambiar las siguientes variables:

  find-window-name = [window name], or
  find-window-class = [window class]

- **ga-server-event-driven:** Carga una aplicación específica que esté en el archivo de configuración, por lo cual sirve para una transmisión dedicada con mayor optimización ya que no se cargará todo el escritorio. Con la siguiente sintaxis de ejecución:

  ga-server-event-driven [16]
  e.g., $ ga-server-event-driven config/d3dex.conf

- **ga-client:** Con este ejecutable el cliente se conectará al servidor. Con la siguiente sintaxis (teniendo en cuenta que el puerto que se usa por defecto es 8554):

  ga-client [16] rtsp://server-address:server-port/desktop
  e.g., $ ga-client config/client.rel.conf rtsp://192.168.1.1:8554/desktop

  ```
  C:\Users\Julio\Desktop\gaminganywhere-0.8.0\bin>ga-client.exe config/client.abs.
  conf rtsp://192.168.0.5:8554/desktop
  ```

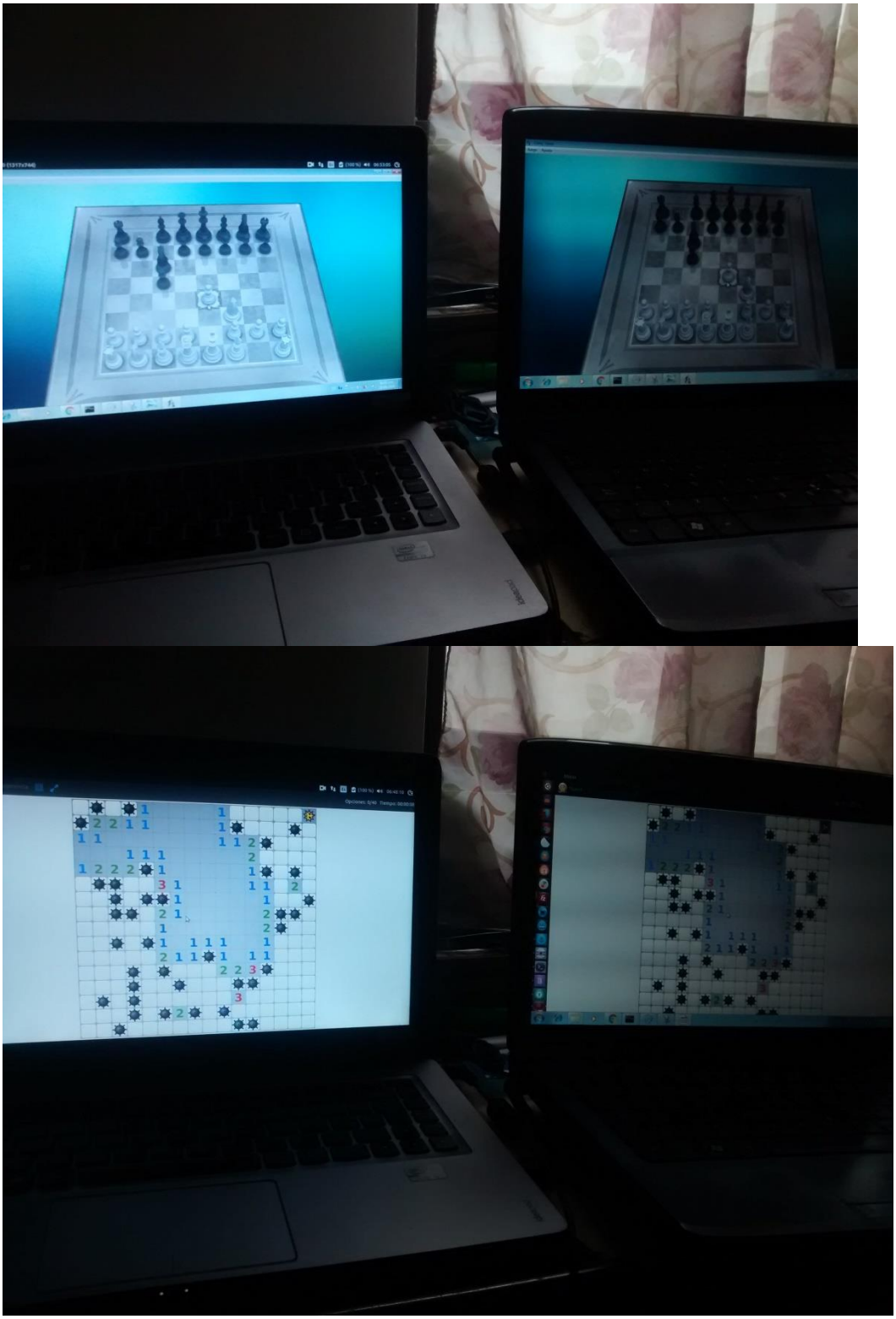
  Existen dos archivos de configuración por defecto corriendo con codificación H.264 y mp3: client.abs.conf (para trabajar con una posición absoluta del mouse) y client.rel.conf (para trabajar con una posición relativa del mouse).

Después de iniciar el ejecutable según la clase de streaming que se desea, debería aparecer algo como esto:

```
Administrador: C:\Windows\System32\cmd.exe - ga-server-periodic.exe config/server.desktop.conf
# [3336] 1461152913.991384 sdl replayer: Replay using SendInput(), screen-size=1
366x768
# [3336] 1461152914.006984 dpipe: 'video-0' initialized, 8 frames, framesize = 1
6384096
# [3336] 1461152914.178584 video-source: video-0 initialized max-curr-out = (256
0x1600)-(1366x768)-(1366x768)
# [3336] 1461152914.178584 Frame converter created: from (1366,768)[30] -> (1366
,768)[0]
# [3336] 1461152914.178584 dpipe: 'filter-0' initialized, 8 frames, framesize =
16384096
# [3336] 1461152914.334584 video encoder: video source #0 from 'filter-0' (1366x
768).
# [3336] 1461152914.381384 vencoder-init: option b = 3000000
# [3336] 1461152914.381384 vencoder-init: option g = 48
# [3336] 1461152914.396984 vencoder-init: option intra-refresh = 1
# [3336] 1461152914.396984 vencoder-init: option me_method = dia
# [3336] 1461152914.396984 vencoder-init: option me_range = 16
# [3336] 1461152914.396984 vencoder-init: option preset = faster
# [3336] 1461152914.412584 vencoder-init: option profile = main
# [3336] 1461152914.412584 vencoder-init: option refs = 1
# [3336] 1461152914.412584 vencoder-init: option slices = 4
# [3336] 1461152914.428184 vencoder-init: option threads = 4
# [3336] 1461152914.428184 vencoder-init: option tune = zerolatency
[libx264 @ 006fa1c0] using cpu capabilities: MMX2 SSE2Fast LZCNT
[libx264 @ 006fa1c0] profile Main, level 3.2
[libx264 @ 006fa1c0] 264 - core 142 r2431 ac76440 - H.264/MPEG-4 AVC codec - Cop
yleft 2003-2014 - http://www.videolan.org/x264.html - options: cabac=1 ref=1 deb
lock=1:0:0 analyse=0x1:0x111 me=dia subme=4 psy=1 psy_rd=1.00:0.00 mixed_ref=0 m
e_range=16 chroma_me=1 trellis=1 8x8dct=0 cqm=0 deadzone=21,11 fast_pskip=1 chro
ma_qp_offset=0 threads=4 lookahead_threads=4 sliced_threads=1 slices=4 nr=0 deci
mate=1 interlaced=0 bluray_compat=0 constrained_intra=0 bframes=0 weightp=1 keyi
nt=48 keyint_min=4 scenecut=40 intra_refresh=1 rc=abr mbtree=0 bitrate=3000 rate
tol=1.0 qcomp=0.60 qpmin=0 qpmax=69 qpstep=4 ip_ratio=1.40 aq=1:1.00
# [3336] 1461152914.568585 video encoder: initialized.
# [3336] 1461152914.584185 WAVEFORMATEX: samplerate=44100, bitspersample=32
# [3336] 1461152914.646585 audio source: setup chunk=4410, samplerate=44100, bps
=16, channels=2
# [3336] 1461152914.689785 audio encoder: encoder_size=4608, frame_size=1152, ds
tlines[0] = 2304
# [3336] 1461152914.689785 audio encoder: on-the-fly audio format conversion ena
bled.
# [3336] 1461152914.689785 audio encoder: convert from 2ch(3)@44100Hz (s16) to 2
ch(3)@44100Hz (s16p).
# [3336] 1461152914.699785 audio encoder: initialized.
# [3336] 1461152914.699785 video source thread started: tid=2192
# [3336] 1461152914.699785 video encoder: ffmpeg-video-encoder registered
# [3336] 1461152914.699785 controller socket: socket address [0.0.0.0:8555]
# [3336] 1461152914.709785 controller server started: tid=1796.
# [3336] 1461152914.709785 audio encoder: ffmpeg-audio-encoder registered
# [3336] 1461152914.709785 audio source thread started: tid=2004
# [3336] 1461152914.719785 WASAPI: estimated trimmed frames = 3228
# [3336] 1461152914.699785 RGB2YUV filter[940]: pipe#0 from 'video-0' to 'filter
-0' (output-resolution=1366x768)
# [3336] 1461152914.739785 encoder: packet queue initialized (3x3145728 bytes)
# [3336] 1461152914.739785 qos-measurement: initialized.
# [3336] 1461152914.749785 (Use port 80 for optional RTSP-over-HTTP tunneling.)
```

```
[libx264 @ 0xa58cc0] profile Main, level 3.2
[libx264 @ 0xa58cc0] 264 - core 148 - H.264/MPEG-4 AVC codec - Copyleft
2003-2015 - http://www.videolan.org/x264.html - options: cabac=1 ref=1 d
eblock=1:0:0 analyse=0x1:0x111 me=dia subme=4 psy=1 psy_rd=1.00:0.00 mix
ed_ref=0 me_range=16 chroma_me=1 trellis=1 8x8dct=0 cqm=0 deadzone=21,11
 fast_pskip=1 chroma_qp_offset=0 threads=4 lookahead_threads=4 sliced_th
reads=1 slices=4 nr=0 decimate=1 interlaced=0 bluray_compat=0 constraine
d_intra=0 bframes=0 weightp=1 keyint=48 keyint_min=4 scenecut=40 intra_r
efresh=1 rc=abr mbtree=0 bitrate=3000 ratetol=1.0 qcomp=0.60 qpmin=0 qpm
ax=69 qpstep=4 ip_ratio=1.40 aq=1:1.00
# [3477] 1461152377.464598 video encoder: initialized.
1461152377.464639
# [3477] 1461152377.477009 ALSA init: unable to retrieve pcm delay
# [3477] 1461152377.477032 audio source: setup chunk=5512, samplerate=44
100, bps=16, channels=2
# [3477] 1461152377.496965 audio encoder: encoder_size=4608, frame_size=
1152, dstlines[0] = 2304
# [3477] 1461152377.497115 audio encoder: on-the-fly audio format conver
sion enabled.
# [3477] 1461152377.497138 audio encoder: convert from 2ch(3)@44100Hz (s
16) to 2ch(3)@44100Hz (s16p).
# [3477] 1461152377.497159 audio encoder: initialized.
# [3477] 1461152377.497312 controller socket: socket address [0.0.0.0:85
55]
# [3477] 1461152377.497368 video encoder: ffmpeg-video-encoder registere
d
# [3477] 1461152377.497419 video source thread started: tid=3490
# [3477] 1461152377.497518 audio encoder: ffmpeg-audio-encoder registere
d
# [3477] 1461152377.497454 controller server started: tid=3489.
# [3477] 1461152377.497591 audio source thread started: tid=3492
# [3477] 1461152377.497683 RGB2YUV filter[3491]: pipe#0 from 'video-0' t
o 'filter-0' (output-resolution=1366x768)
# [3477] 1461152377.497966 encoder: packet queue initialized (3x3145728
bytes)
# [3477] 1461152377.499021 qos-measurement: initialized.
# [3477] 1461152377.499134 (Use port 8000 for optional RTSP-over-HTTP tu
nneling.)
```

Ya que el cliente está escuchando y el servidor está transmitiendo, el cliente tiene acceso al juego como se muestra a continuación:



GUÍA DE INSTALACIÓN DE OPENDAYLIGHT Y MININET

A continuación se mostrará cómo instalar OpenDayLight y cómo integrar este controlador a Mininet. En primer lugar es importante señalar que el hecho de utilizar OpenDayLight y no otros controladores disponibles radica en una investigación previa de otros controladores, además que éste tiene apoyo de grandes empresas relacionadas con la materia como CISCO y las prestaciones que nos ofrece.

Antes de descargar cualquier distribución de OpenDayLight hay que tener en cuenta sus limitaciones y posibilidades. Podemos distinguir entre dos distribuciones del controlador; la distribución base y la Helium:

- La distribución base ofrece lo mínimamente necesario para poder arrancar el controlador. Se trabaja con OSGi (Open Service Gateway initiative), lo cual permite diseñar plataformas compatibles que puedan proporcionar múltiples servicios, en este caso con Java, facilitanto la programación, instalación, ejecución y detención de paquetes de este lenguaje de manera sencilla.

- La distribución Helium está desarrollada con una serie de complementos cuyo objetivo es permitirnos interactuar con el controlador de forma más transparente. Por lo cual se trabaja con Karaf, una plataforma genérica que proporciona funciones y servicios de alto nivel diseñados específicamente para la creación de servidores basados en OSGi (Open Service Gateway initiative). Esta distribución viene con una serie de proyectos incluidos que son fáciles de instalar gracias a Karaf. Para más información acerca de como trabajar con Karaf se puede visitar la siguiente página:

  https://wiki.opendaylight.org/view/Karaf:Step_by_Step_Guide

Ambas son fáciles de instalar ya que basta con descargar un .zip y descomprimirlo en el directorio deseado. Para este proyecto se usó la versión Beryllium-SR3 por su estabilidad, aunque la versión más reciente a la fecha es la Boron (lanzada en Septiembre 21 de 2016).

1. Descarga y descompresión

Antes de empezar se recomienda instalar Java JDK o en su defecto OpenJDK puesto que OpenDayLight está desarrollado en Java.

Primero se descarga la versión elegida de OpenDayLight y se descomprime en un directorio, en este caso se usará la Berylium-SR3, la cual está disponible con el siguiente enlace:

https://nexus.opendaylight.org/content/repositories/public/org/opendaylight/integration/distribution-karaf/0.4.3-Beryllium-SR3/distribution-karaf-0.4.3-Beryllium-SR3.zip

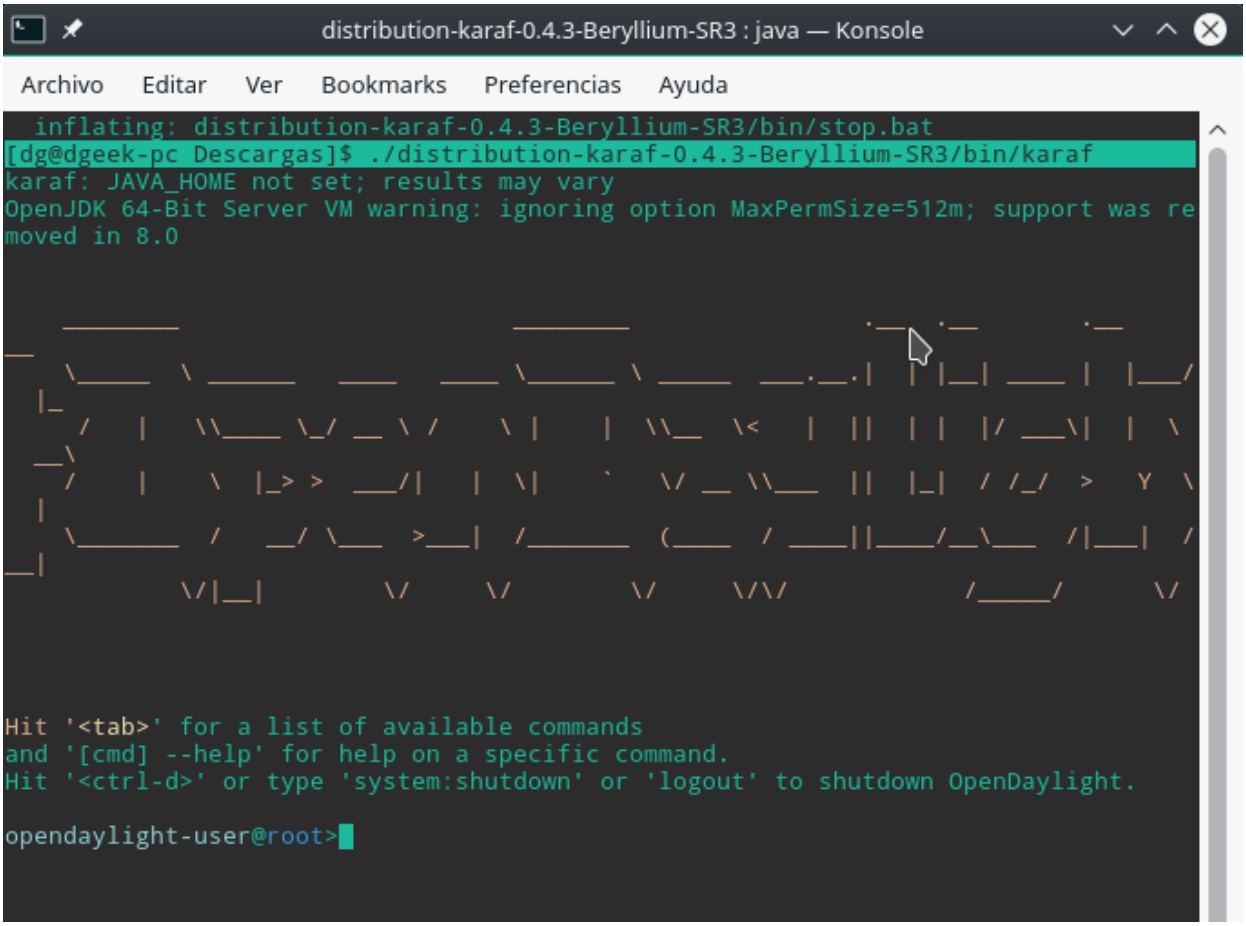
En caso de tener alguna distribución basada en GNU/Linux basta con copiar el link antecedido de la palabra *wget* en la terminal para iniciar su descarga. Para descomprimir el archivo desde una terminal se ingresa el comando *unzip archivo.zip* (sólo sí se descargó la versión en zip) o *tar -xzvf archivo.tar.gz* (si la extensión es .tar.gz):

unzip distribution-karaf-0.4.3-Beryllium-SR3.zip
tar -xzvf distribution-karaf-0.4.3-Beryllium-SR3.tar.gz

2. **Ejecución**

Se ingresa a la carpeta *bin* del archivo descomprimdo y se ejecuta en la terminal el archivo *karaf:*
./distribution-karaf-0.4.3-Beryllium-SR3/bin/karaf

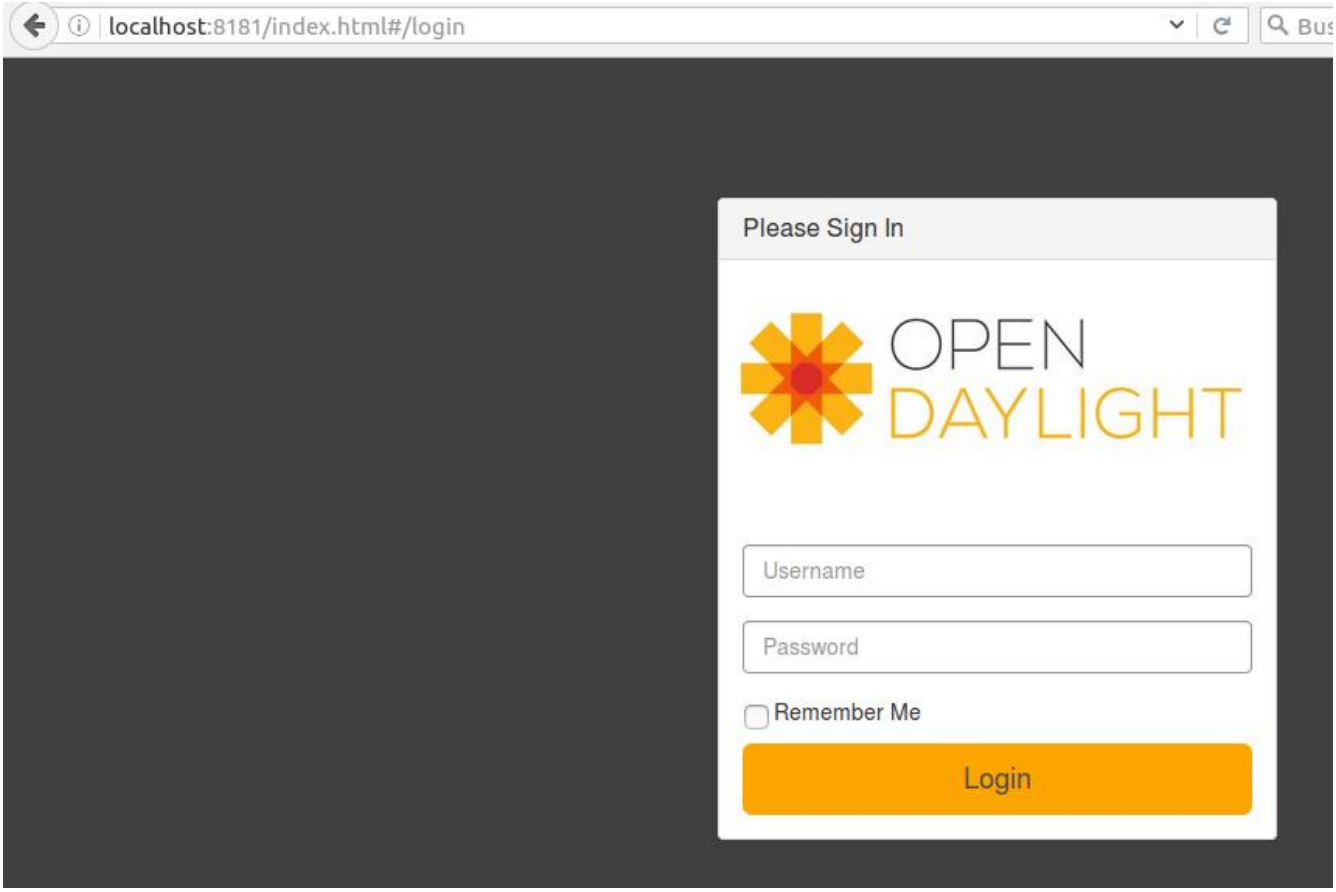
Sí todo salió bien debería iniciarse OpenDayLight de la siguiente forma:



Para instalar componentes en el controlador desde la consola de OpenDayLight hay que escribir *feature:install nombredecomponente*. En este caso se instalará un algoritmo de enrutamiento y una extensión para que la topología pueda ser vista gráficamente:

feature:install odl-restconf odl-l2switch-switch odl-mdsal-apidocs odl-dlux-core

Para comprobar que el controlador se está ejecutando correctamente, se debe ingresar en un navegador a la dirección “localhost:8181/index.html”, y aparecerá un panel de administración como el siguiente:



Para acceder el usuario y contraseña por defecto son “admin”.

3. Instalación de Mininet

Para instalar Mininet basta con hacerlo desde los repositorios o paquetes de la distribución de GNU/Linux. Por ejemplo en el caso de distribuciones basadas en Debian: *sudo apt-get install mininet,*

También puede instalarse desde el código fuente siguiendo las indicaciones del siguiente enlace: http://mininet.org/download/#option-2-native-installation-from-source

4. Integración de OpenDayLight con Mininet

Sí se siguieron los pasos anteriores en este momento el controlador se está ejecutando en una dirección IP, la cual puede comprobarse escribiendo en consola el comando “ifconfig”.

Para comprobar que una red puede reconocer el controlador, se recomienda montarla en Mininet y enlazarla con la IP del controlador.

En este caso se creó un script de la red que iba a ser implementada a futuro en GENI, nombrado “topology2.py” con el siguiente código:

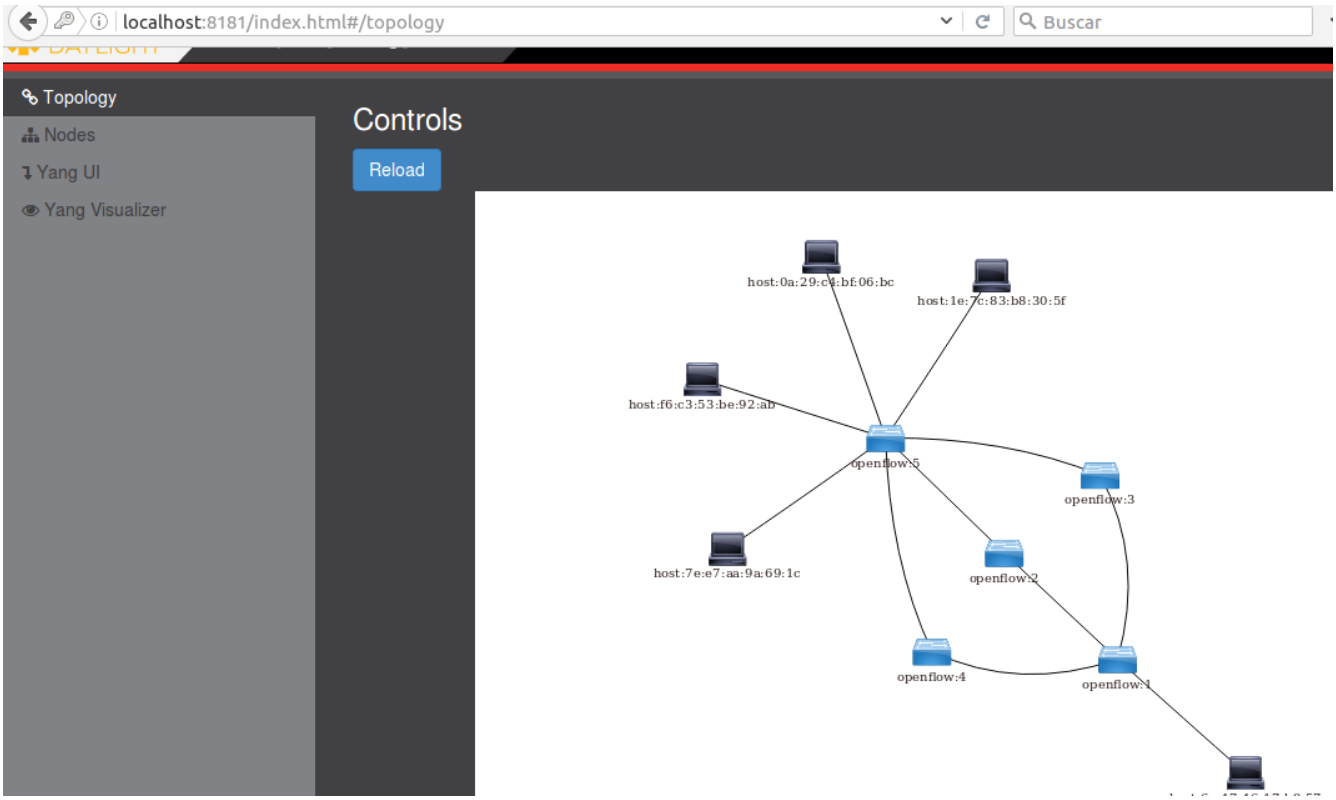
```
#!/usr/bin/python

from mininet.net import Mininet
from mininet.node import Controller, RemoteController, OVSController
from mininet.node import CPULimitedHost, Host, Node
from mininet.node import OVSKernelSwitch, UserSwitch
from mininet.node import IVSSwitch
from mininet.cli import CLI
from mininet.log import setLogLevel, info
from mininet.link import TCLink, Intf
from subprocess import call

def myNetwork():

    net = Mininet( topo=None,
                   build=False,
                   ipBase='10.0.0.0/8')

    info( '*** Adding controller\n' )
    c0=net.addController(name='c0',
                         controller=Controller,
                         protocol='tcp',
                         port=6633)

    info( '*** Add switches\n')
    s5 = net.addSwitch('s5', cls=OVSKernelSwitch)
    s3 = net.addSwitch('s3', cls=OVSKernelSwitch)
    s2 = net.addSwitch('s2', cls=OVSKernelSwitch)
    s4 = net.addSwitch('s4', cls=OVSKernelSwitch)
    s1 = net.addSwitch('s1', cls=OVSKernelSwitch)

    info( '*** Add hosts\n')
    h2 = net.addHost('h2', cls=Host, ip='10.0.0.2', defaultRoute=None)
    h1 = net.addHost('h1', cls=Host, ip='10.0.0.1', defaultRoute=None)
    h5 = net.addHost('h5', cls=Host, ip='10.0.0.5', defaultRoute=None)
    h3 = net.addHost('h3', cls=Host, ip='10.0.0.3', defaultRoute=None)
    h4 = net.addHost('h4', cls=Host, ip='10.0.0.4', defaultRoute=None)

    info( '*** Add links\n')
    net.addLink(h1, s1)
    net.addLink(s1, s3)
    net.addLink(s1, s2)
```

```
    net.addLink(s1, s4)
    net.addLink(s2, s5)
    net.addLink(s4, s5)
    net.addLink(s3, s5)
    net.addLink(s5, h2)
    net.addLink(s5, h3)
    net.addLink(s5, h4)
    net.addLink(s5, h5)

    info( '*** Starting network\n')
    net.build()
    info( '*** Starting controllers\n')
    for controller in net.controllers:
        controller.start()

    info( '*** Starting switches\n')
    net.get('s5').start([])
    net.get('s3').start([])
    net.get('s2').start([])
    net.get('s4').start([])
    net.get('s1').start([])

    info( '*** Post configure switches and hosts\n')

    CLI(net)
    net.stop()

if __name__ == '__main__':
    setLogLevel( 'info' )
    myNetwork()
```

Se ejecuta el script con el comando “sudo python topology2.py” (con el controlador previamente iniciado) y posteriormente se hace un ping entre todos los hosts, como puede verse en esta imagen:

```
dg@u310:~/Descargas$ sudo python topology2.py
*** Adding controller
*** Add switches
*** Add hosts
*** Add links
*** Starting network
*** Configuring hosts
h3 h1 h4 h5 h2
*** Starting controllers
*** Starting switches
*** Post configure switches and hosts
*** Starting CLI:
mininet> pingall
*** Ping: testing ping reachability
h3 -> X X X X
h1 -> X X X X
h4 -> X X X X
h5 -> X X X X
h2 -> X X X h5
*** Results: 95% dropped (1/20 received)
mininet> pingall
```

Al ingresar a la IP del controlador en el puerto 8181 desde un navegador e iniciar sesión debería mostrarse la topología de la red correctamente:



CREACIÓN DE CUENTA DE GERRIT EN OPENDAYLIGHT

Git es un sistema de control de versiones de código abierto distribuido (DVCS) escrito en el lenguaje C y desarrollado originalmente por Linus Torvalds y otros, para gestionar el kernel de Linux. En Git, no hay ninguna copia central de un repositorio, ya que éste se puede clonar, teniendo así una copia total funcional del código fuente con todas las ramas, etiquetas y cambios en un repositorio local.

Gerrit es una herramienta de revisión de código de colaboración basada en una web de código abierto que se integra con Git.,Fue desarrollada en Google por Shawn Pearce. Gerrit proporciona un marco para revisar el código y cada cambio que hacen los desarrolladores o la comunidad en general, antes de que sean aceptados en el código base. Los cambios se pueden subir a Gerrit por cualquier usuario. Sin embargo, los cambios no se realizan en una parte del proyecto hasta que se complete una revisión de código. Gerrit también es una buena herramienta de colaboración para el almacenamiento de las conversaciones que se producen cuando se sube un código.

El código fuente OpenDaylight se encuentra alojado en un repositorio Git, y los desarrolladores deben utilizar Gerrit para poder subir cambios en el código del repositorio OpenDaylight.

Para obtener más información sobre Git, ver http://git-scm.com/ y para Gerrit https://code.google.com/p/gerrit/.

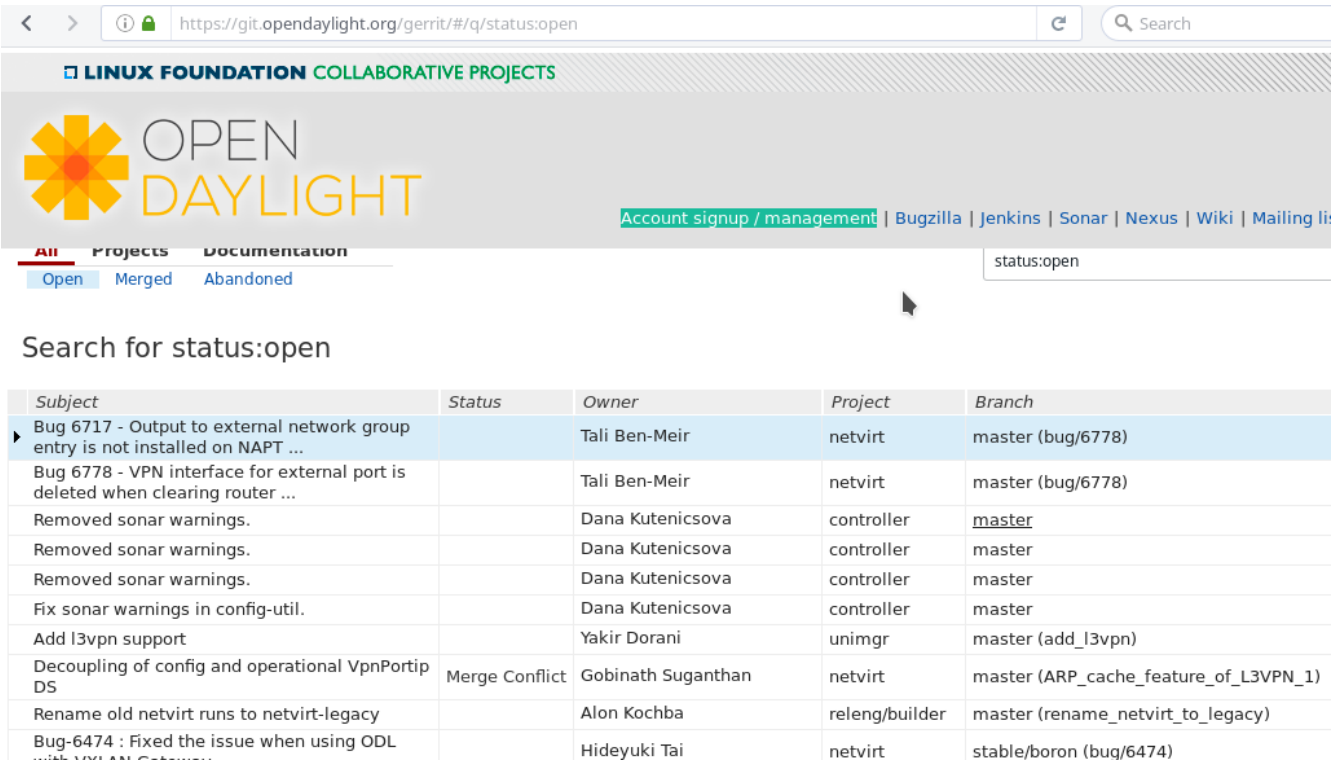
NOTA: Se debe tener una cuenta de Gerrit antes de poder empezar a contribuir al proyecto de OpenDaylight o desarrollar.

A continuación se mostrarán los pasos para poder crear una cuenta en el Gerrit de OpenDayLight:

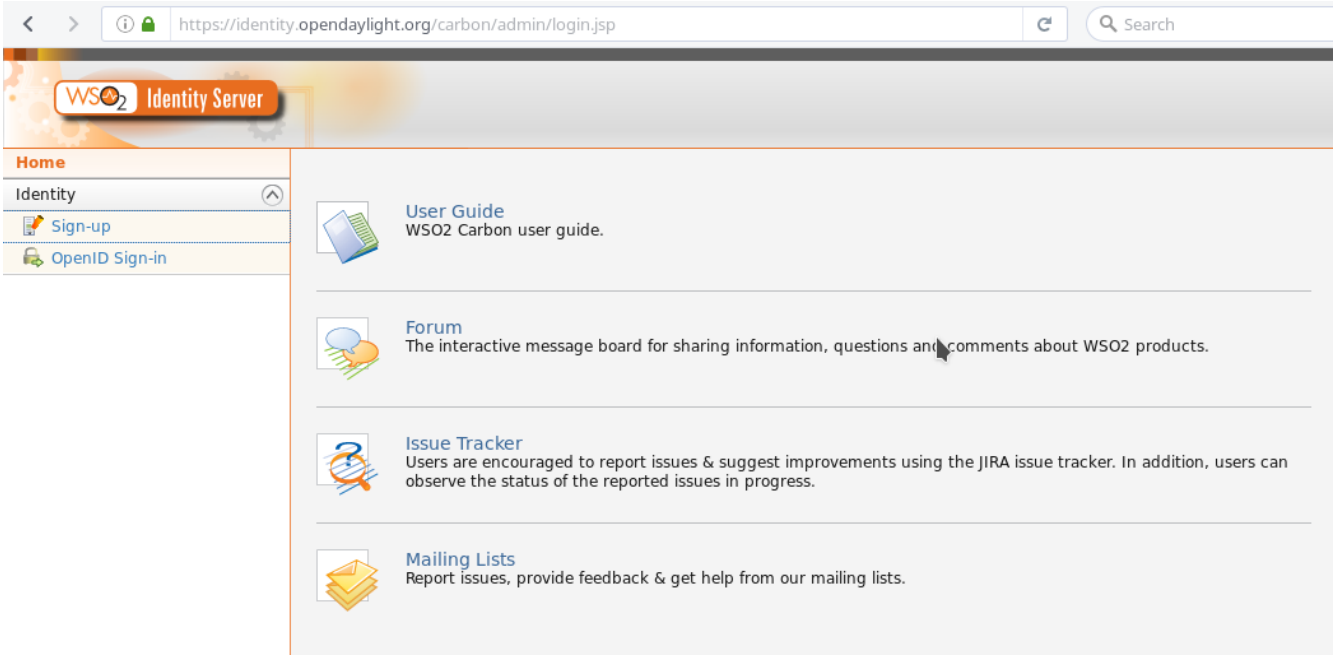
1. Registro

Usando un navegador se ingresa al siguiente enlace: https://git.opendaylight.org/gerrit/. La página web muestra los cambios que han hecho en el código los desarrolladoreyados en tiempo real, pero cada uno de ellos son revisados y comentados para posteriormente unirlos al código fuente de OpenDayLight sí fueron exitosos.

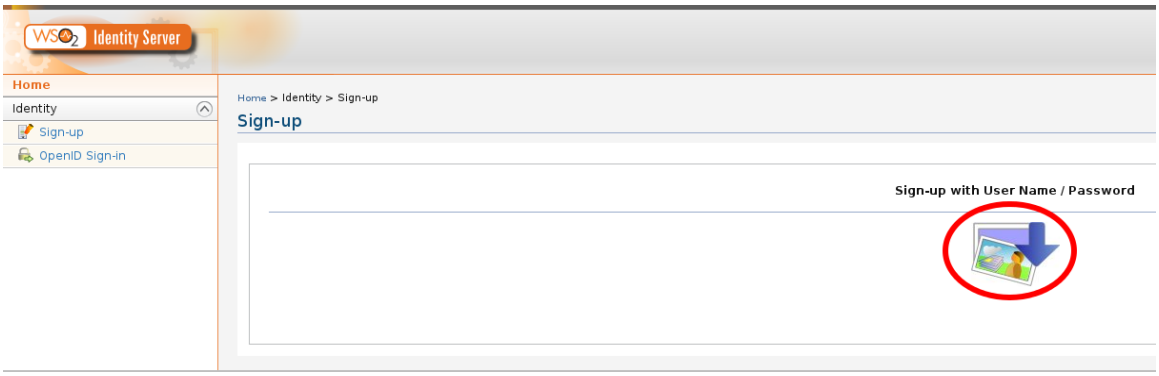
Para crear una cuenta se dará clic en el texto subrayado “Account signup / management”:

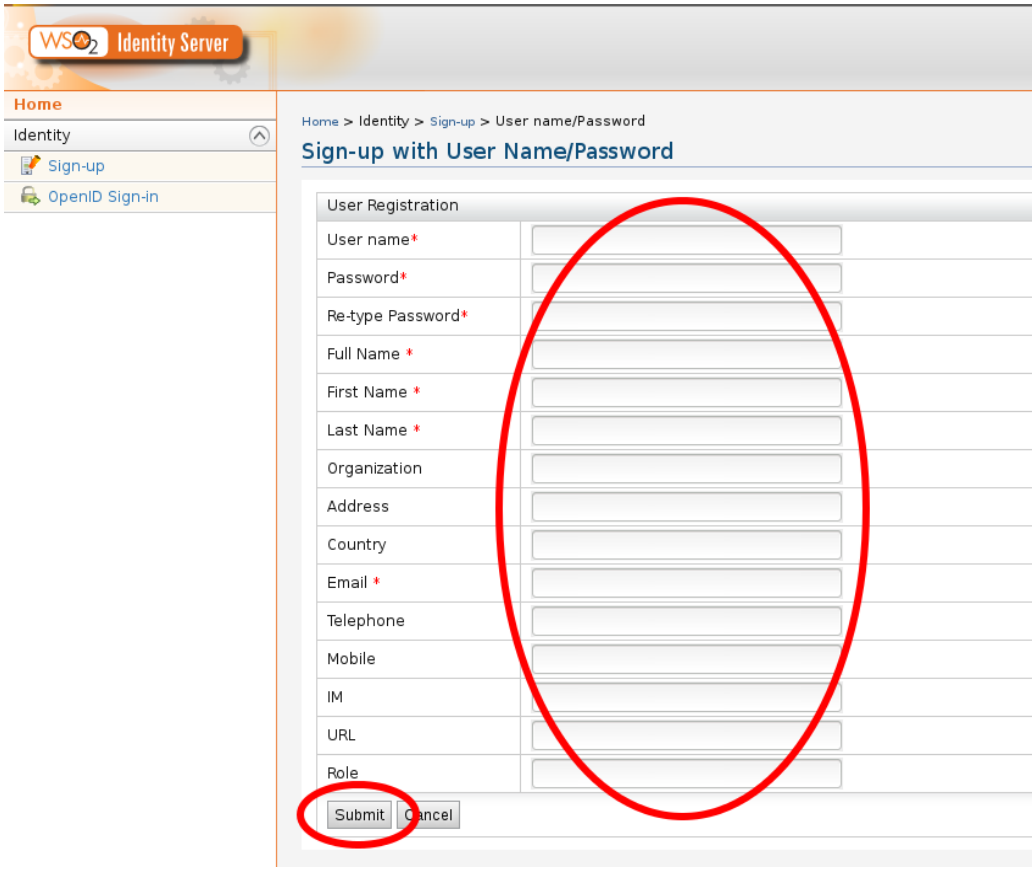


Aparecerá una página web llamada “WS02 Identity Server page”, la cual sirve para crear identificaciones en OpenDaylight:



Se dará clic en la opción “Sign Up” del panel izquierdo, luego en “Sign-up with User Name / Password”, y finalmente se llenarán los datos de la cuenta, como se muestra a continuación:





Sí el proceso de registro fue exitoso se deberá poder iniciar sesión en cualquiera de los siguientes enlaces con los datos de la cuenta:

https://git.opendaylight.org/gerrit/login/
https://identity.opendaylight.org/carbon/admin/login.jsp

2. Creación de SSH

Para poder hacer cambios en el código fuente con el Gerrit de OpenDayLight se requiere al menos una llave o clave de SSH, de lo contrario no se podrán hacer cambios ni trabajar en un entorno de desarrollo como es el caso de Eclipse. El método para la generación de claves SSH es diferente según el tipo de sistema operativo.

La clave que se registra con Gerrit debe ser idéntica a la que se va a utilizar más adelante para traer los cambios o modificar el código. Por ejemplo, si se tienen diferentes máquinas virtuales o computadores con claves SSH generadas para cada uno de ellos debe ser única, de lo contrario Gerrit no funcionará.

NOTA: Para obtener más información sobre las claves SSH para Ubuntu, consulte https://help.ubuntu.com/community/SSH/OpenSSH/Keys. Para la generación de claves SSH para Windows, consulte https://help.github.com/articles/generating-ssh-keys.Eclipse tiene una opción para generar claves SSH, la cual se describirá en otra sección.

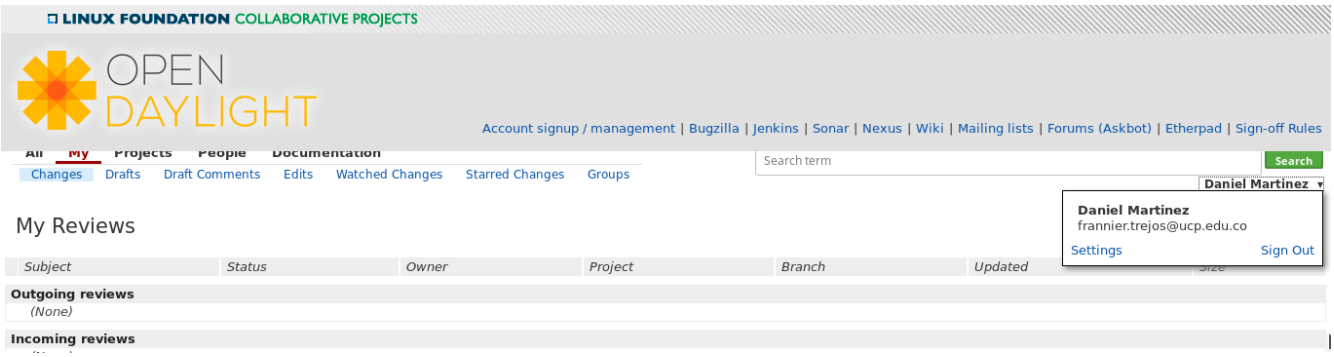
Para un sistema que ejecuta Ubuntu o sistemas operativos Fedora / CentOS / RHEL, para generar una clave SSH basta con los siguientes comandos:

```
mkdir ~/.ssh
chmod 700 ~/.ssh
ssh-keygen -t rsa
```

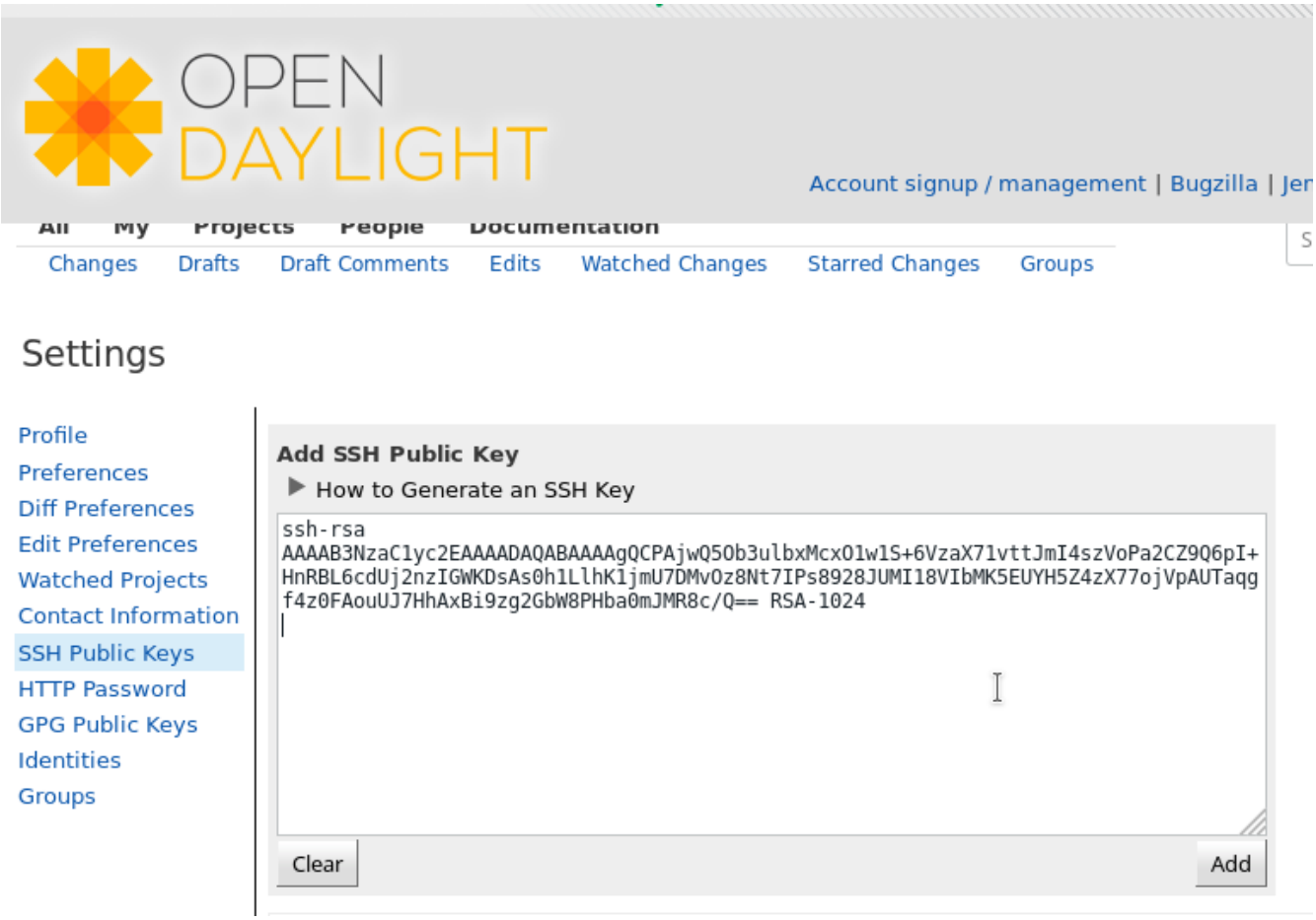
En caso de querer añadir una frase de verificación a la clave SSH se ejecuta al final el comando “ssh-keygen -p”

3. Adición de la clave SSH a Gerrit

Para añadir la clave SSH recién creada a Gerrit, se inicia sesión para acceder al repositorio de OpenDayLight en el siguiente enlace: https://git.opendaylight.org/gerrit. Luego se accede a la opción “Settings” como se muestra a continuación:



Se dará clic a la opción “SSH Public Keys” que aparece en el panel izquierdo. En el recuadro “Add SSH Public Key” se pegará la clave SSH previamente creada y finalmente se dará clic en “Add”:



NOTA: En caso de seguir los comandos anteriores para la creación de una clave SSH en GNU/Linux, se pegará la información del archivo “id_rsa.pub” ubicado en la dirección “.ssh/id_rsa.pub” en /home.