Introducción a los Sistemas Distribuidos (75.43)
TP #3: Enlace

Esteban Carisimo[1] y AMD[1]

[1]Facultad de Ingeniería, Universidad de Buenos Aires

6 de noviembre de 2018

Resumen

Las Software-Defined Networks han sido el tema dominante de la investigación y la innovación en Internet desde su aparición en 2008. Su irrupción en escena surgió de la necesidad de poder flexibilizar el uso del hardware para poder satisfacer las nuevas demandas de Internet. Su éxito fue casi inmediato, tal es así que Google en 2009 ya se encontraba utilizando esta tecnología para administrar el tráfico en su red global. Este trabajo práctico tiene como objetivo familiarizarse con los desafios por los cuales surgen las SDNs, el protocolo OpenFlow, a través del cual se programan los dispositvos de red. Dado que ahora los dispositivos son programables, también se buscará aprender a controlar el funcionamiento de los switches a través de APIs en diferentes lenaguajes de programación. De manera complementaria se estudiarán los datacenters, aprendiendo su arquitectura, la cual es a su vez una excusa, para ver los protocolos de capa de enlace. Finalmente se pedirá hacer una implementación de una SDN en un datacenter virtualizado.

Palabras clave— OpenFlow, Software-Defined Networking, Traffic Engineering, Datacenters

Conceptos previos específicos

Para abordar este trabajo no sólo se requiere el manejo conceptual de la escencia de los protocolos anteriormente vistos, sino que también se necesita conocer y dominar los siguientes temas puntuales

1. Control y Forwarding path
2. Concepto de flujo
3. IP blackholing
4. Firewall

1. Introducción

Internet ha evolucionado a velocidades inimaginadas, y a pesar de que hoy el sistema de comunicaciones se sigue basando en un paradigma diseñado a fines de los 60s, cada 5 o 10 años Internet introduce innovaciones significativas que llevan al cambio de su fisonomía. La reciente masificación de los contenidos multimedia, sumados al cloud computing y a la exhorbitante cantidad de nuevos dispositvos conectados luego del advenimiento de los smarthpones y los IoT, ha planteado nuevos desafios en Internet.

Este proceso a llevado a que Internet distribuya grandes volumenes de tráfico. A su vez, en este escenario, lo proveedores de contenidos monopolizan el centro de la escena. La mayoría los datos de los usuarios de Internet son almacenados en monumentales datacenters. Estos factores han motivado a un uso más eficiente de los recursos y ha readaptar las necesidades. A continuación se introducirán OpenFlow y las Software-Defined Networks (SDN), las cuales son el resultado de esta etapa de la historia de Internet donde se necesita más felixibilidad y dinamismo a la hora de la distribución de los paquetes. En este sentido se mostrará como en los datacenters se observa nítidamente esta necesidad para hacer un uso racional de la estructura y la inversión.

1.1. Software-Defined Netowrking: Introducción a la problemática

En la actualidad existe una innumerable cantidad de routers y switches que conforman la red global de Internet. Más aún, existe una gran variedad de dispositivos dependiendo de su envergadura, sus requerimientos y su ubicación en la red, lo que eventualmente determina su costo y su consumo energético. A pesar de que no existe una lista sumamente extensa de fabricantes, los dispostivos presentan diferencias de acuerdo a su manufacturante, más allá de que todos respeten los protocolos TCP/IP y eventualmente IEEE 802.3, IEEE 802.11, etcétera. Estas diferencias se deben a prestaciones extra que los fabricantes introducen en los equipos para cumplir con *buenas prácticas*, satisfacer calidad de servicio o políticas de seguridad. Poco se conoce de cómo o cuáles son las funciones complementarias que introducen los equipos, sumado que existen nulas o limitadas formas de interactuar con estas prestaciones extra.

Un ejemplo de suma importancia dentro de las funcionalidades extra que ejectuan los routers, son las *buenas prácticas* sobre los flujos a la hora del ruteo (ver RFC2991 [1]). Suponiendo de que un router tiene más de un camino de igual costo para acceder a un destino, éste debe enviar los paquetes pertenecientes al flujo siempre por el mismo camino. Por más que no exista una definición estrícta de que es un flujo, podemos interpretarlo como la 10-tupla formada por $(PortIn, VLANID, srcEth, dstEth, typeEth, srcIP, dstIP, protoIP, srcport, dstport)$. Entonces, para poder hacer esto, los routers no sólo deben actuar de acuerdo a la dirección IP destino, sino que también a otros campos de las cabeceras Ethernet, IP y TCP. Sin embargo, llegado este punto, los usuarios (lease los administradores de la red), no pueden decidir que directivas ejectuar ante diferentes 10-tuplas.

Este último ejemplo mencionado es de vital importancia en todo el campo de Internet, ya que el protocolo IP fue concebido para que el ruteo sólo implique la dirección destino, y consecuentemente a una entrada en la tabla de ruteo. Aquí es necesario rememorar que la redes de conmutación de paquetes y en especial Internet surgieron con el objetivo de interconectar computadoras a través de sistemas intermediarios (routers) sumamente sencillos. Sin embargo, la evolución de la red sumando capacidades multimedia e interactivas, ha llevado a que el ruteo basado simplemente en dirección destino sea insuficiente para proveer calidad de experiencia.

Todo este problema de inflexibilidad que existe sobre los switches y routers naturalmente lleva a un problema en la innovación, y en la capacidad de adaptar las redes a las necesidades específicas de una organización. Una alternativa que se puede lograr es generar un switch o un router por medio de funciones en software en algún sistema operativo, por ejemplo Linux, donde ya existen algunas distribuciones para esta finalidad. Sin embargo, implementar funciones de routeo y forwarding en software es varios ordendes de magnitud más lento que ejecturarlo en hardware, por medio de memorias de rápido acceso como lo pueden ser las TCAM. Más aún, también aparecerían problemas de escala cuando este swtich virtual deba manejar múltiples interfaces en simultaneo, tal como lo hace un dispositivo de red.

1.2. OpenFlow

Ante los mencionados problemas de flexibilidad y las necesidades de introducir políticas de ruteo más específicas surgió OpenFlow [2], como un protocolo para poder interactuar directamente con las tablas de ruteo. Si analizamos un swtich, este poseé una tabla CAM [1], la cual determina un puerto de salida en función de la dirección MAC destino. Estas memorias son de relativo bajo costo y de muy alta velocidad, entonces OpenFlow se propone poder reutilizar las memorias de los switches de manera inteligente. Como se mencionó anteriormente, hoy el ruteo en base a direcciones destino es insuficiente, entonces la idea es poder generar políticas en base a flujos, por lo cual cada una de las entradas en la tabla correspondería a un flujo. Entonces, la propuesta de OpenFlow consiste en 3 partes:

1. Una tabla de flujos
2. Un canal encriptado de comunicaciones hacia un controlador
3. El protocolo OpenFlow

Tanto routers como switches tienen su arquitectura dividida en dos: el plano de control y el plano de datos. En el plano de control operan las decisiones de como administrar el dispositivo, las cuales corren en software, en cambio en el plano de datos, se administra la recepción y envío de paquetes y opera en el plano de hardware. La idea de OpenFlow es crear un protocolo de comunicación para poder controlar el plano de control, que eventualmente gestiona las entradas en las tablas de ruteo del plano de datos.

Sumado al concepto de una tabla de ruteo manipulable surge la idea de un controlador, un elemento externo capaz de configurar la tabla de flujos, tal como se lo muestra en la Figura 1. La relación entre el switch y el controlador puede ofrecer dos dinámicas: Configuración inicial, donde el controlador asigna funciones una única vez cuando se instala el

[1] CAM: Content Addressable Memory

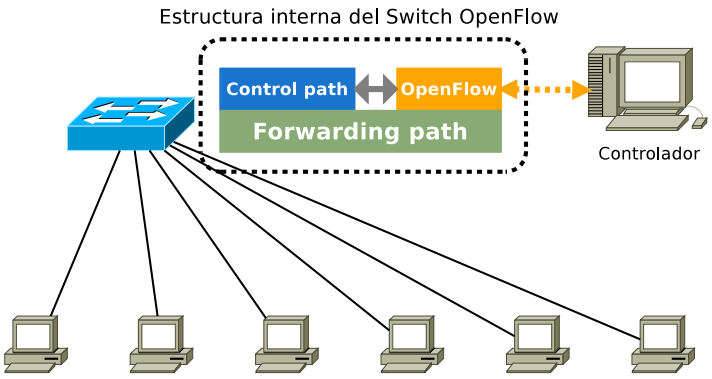


Figura 1: .

port IN	VLAN tag	eth			IP			TCP	
		src	dst	Type	src	dst	Proto	src	dst

Cuadro 1: Los 10 campos que se pueden utilizar para individualizar un flujo en OpenFlow

switch, o dinámica, en donde constantemente se definen nuevas políticas. Es importante destacar que estos switches no aprenden, por lo cual sin interacción con el controlador, nunca enviarán un paquete.

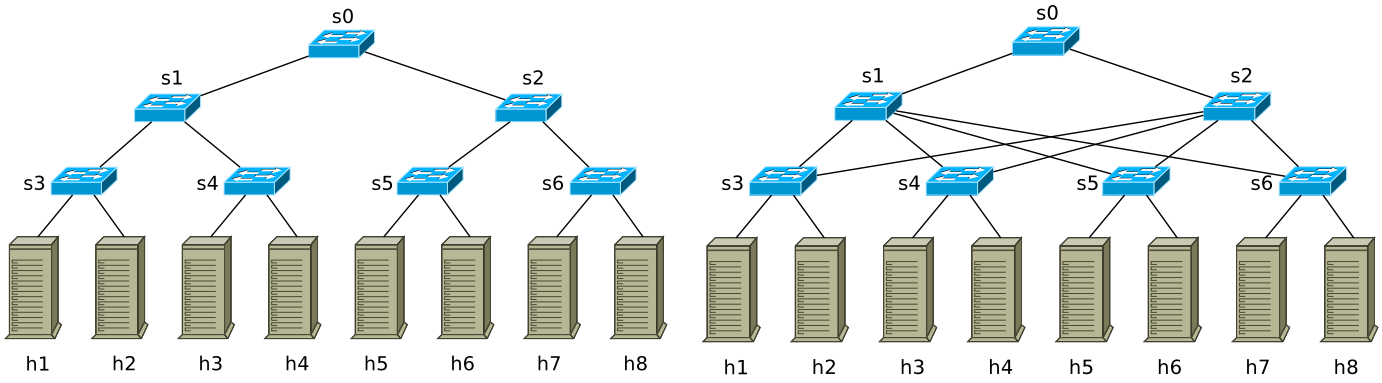
Dado que OpenFlow se enfoca a poder controlar los flujos, los switches que manejan este protocolo pueden ejecutar multiples acciones según si los paquetes coinciden con una entrada en la tabla de flujos. Es necesario destacar, que las tablas de flujos admiten *wildcards*. Por ejemplo en las accciones por flujo se pueden tomar políticas de ingeniería de tráfico, reescribir campos de la cabecera si fuera necesario (NAT) o descartar paquetes en caso de un ataque (Blackholing). Para esto los switches cuentan con hardware específico para la lectura de 10 campos, los cuales aparecen en la Tabla 1, por medio de los cuales se pueden individualizar un flujo.

1.3. Datacenters: Arquitectura y necesidades

En la actualidad existe un considerablemente grande número de datacenters, dependiendo de que consideremos nosotros como un datacenter. La gran mayoría de los datancenter suelen ser pequeños *clusters*, los cuales están ubicados alrededor de todo el mundo, y cuya misión es llevar poder de cálculo cerca del usuario. Por otra parte, existe un menor, pero de todas maneras relevante, número de *mega* datacenters, los cuales albergan gran cantidad de nodos y capacidad de almacenamiento. El costo de estos últimos datacenters, las necesidades, la disponibilidad de recursos humanos, la provición de suminitros energéticos, y razones de seguridad y geografía hacen que naturalmente haya un número menor. Sin lugar a dudas, la envergadura de estos descomunales datacenters hace que los desafios de la ingeniería y la computación sean claramente diferentes.

Los grandes datacenters albergan decenas a centenas de miles de computadoras, las cuales se agrupan de diferentes maneras para formar conjuntos de procesamiento denominados *clusters*. La gran capacidad de cómputo y almacenamiento que se aloja dentro de estos edificios hace que gran cantidad de usuarios hagan uso de esta capacidad instalada. Este es el ejemplo de los usuarios que concurren a un servicio web, donde miles o millones de usuarios en simultaneo consume un serivicio masivo, y por lo tanto generan diferentes tipos de interacciones con los recursos del datacenter. En este caso tenemos un sinfin de ejemplos, comenzando desde portales de venta online, las redes sociales, la transmisión de eventos deportivos o la inscripción a las materias de la Facultad de Ingeniería. Tambien existe el caso antagónico, pero no menos exigente, donde un reducido número de usuarios hace uso de los recursos, pero de manera intensiva, como pueden ser las investigaciones relacionadas con las ciencias de la atmósfera o el estudio de la genética. Por último, el crecimiento de la disponibilidad de estos recursos ha llevado a que nuevas disiplinas que antes eran ajenas al mundo de la computación también hagan uso de los datacenters, como puede ser el caso de los museos.

El desafio de la escala y la concurrencia plantea un desafio central, cómo poder hacer uso óptimo de la estructura, resiliente y sin cuellos de botella. Más aún, el tráfico de los datacenters no solamente se da prodcuto de los usuarios que concurren desde el exterior a solicitar un servicio, sino que dichas solicitudes suelen desencadenar tráfico entre diferentes nodos internos del datacenter. Un pequeño ejemplo es una página web, con PHP, donde la consulta a la base de datos lleva a concurrir a otro nodo.



(a) Topología de árbol binario

(b) Topología de *Fattree*

Figura 2: .

1.3.1. Arquitectura de un datacenter

Desentendiendonos de la problematica que surgiere conectar el datacenter con múltiples conexiones hacia el exterior, podemos ver que si tenemos suficiente capacidad de entrada, de todas maneras será un desafio administrar los flujos por dentro del datacenter. Los datacenters interconectan sus nodos, exclusivamente o casi excluvisamente, a través de switches. Para poder hacer uso de toda la infraestrcutura computacional, los datacenters cuenta con una red que dispone de caminos múltiples para acceder desde un nodo a otro. El diseño de una red de caminos múltiples se conoce como sobreprovisión, en inglés *oversubscription*, y esta dada por una tasa que mide cuantos caminos disponibles existen entre dos nodos. Naturalmente, a mayor tasa de sobreprovisión, habrá más tendido de cables, y lógicamente mayor costo y compleijidad. Dentro de la sobreprovisión nosotros nos enfocaremos en la estrcutura de *Fat-tree*, la cual se encuentra explicada en detalle el artículo Al-Fares *et al.* [3]. Nosotros simplemente nos enfocaremos en esta estructura ya que es la mas popular dentro de los datacenters, y a su vez proveé un gran equilibrio entre costo y beneficio. La Figura 2 compará las arquitecturas de un árbol binario y de un datacenter construido con Fat-tree

Ante la sobreprovisión surge la primer pregunta a los conocimientos que tenemos hasta ahora, **¿Cómo podemos hacer uso de todos los enlaces de una red de capa de enlace sobreprovisionada cuando el protoloco de Spanning Tree (STP) genera un árbol binario?**

Pensemos por un momento como funciona STP y veremos que luego de ejectuar este protocolo, no tendremos *loops* a nivel de capa de enlace, por lo cual no tendremos problemas a la hora de generar *broadcast*. Sin embargo, en este caso habremos deshabilitado la mayoría de los enlaces redundantes, de manera que la sobreprovisión será en vano, y sólo servirá en caso de falla de los enlaces. Entonces el resultado es que no tendremos mejora a nivel de congestión y la topología será de árbol binario. Si miramos la Figura 2, vemos que el árbol binario es la estructura activa luego de correr STP mientras que el Fat-tree es la infraesctutura deisponible.

Habiendo visto que STP es parte del problema y no de la solución, debemos recurrir a otras formas de manipular las tramas de capa de enlace, por lo cual, naturalmente, podemos utilizar OpenFlow. Esta no es la única alternativa, ya que tanto Cisco, como el resto de los fabricantes, ofrece switches para datacenters, los cuales son capaces de manejar enlaces redundantes, los cuales implican *loops*. De todas maneras, OpenFlow presenta mayor flexibilidad ya que el control de los flujos es independiente de la marca del hardware.

2. Propuesta de trabajo

El objetivo del trabajo será construir un pequeño datancenter bajo la topología *Fat-tree*, donde se pedirá utilizar OpenFlow para poder hacer uso de múltiples enlaces en simultáneo. Para poder plantear este escenario se emulará el comportamiento del datancenter y de la infraestructura a través de mininet. El trabajo cuenta, con mútiples pasos y requisitos y a continuación iremos introduciendo uno por uno. Además, la catedra proveerá una máquina virtual (VM) donde ya estarán instalados los programas, como también incluidos los archivos accesorios.

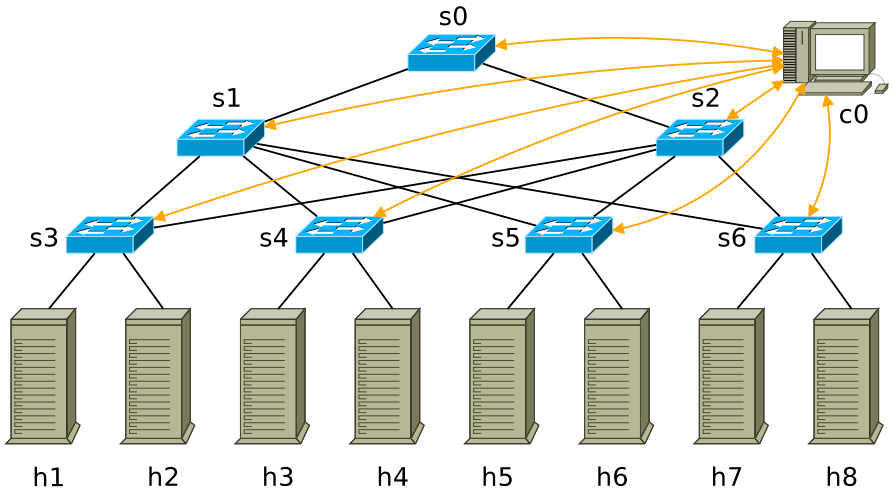


Figura 3: .

2.1. Base conceptual de la simulación

En la introducción hemos incorporado nuevos conceptos y tecnologías, por lo cual, vale la pena hacer una pequeñas enumeración de estos conceptos, para luego ver como se incorporan a la propuesta de trabajo. Entonces, hasta ahora hemos mencionado

- **OpenFlow** Es un protocolo que surge como respuesta a necesidades más específicas a la hora del ruteo. Este protocolo permite que a través de un controlador central se definan políticas de como se deben enviar y clasificar los paquetes. La ventaja radical que ofrece OpenFlow es la independencia del hardware para realizar estas acciones.
- **Datacenters** Los datacenters son elementos vitales de la estructura actual de Internet que cuentan con decenas de miles de computadoras en un único edificio. El uso eficiente de los nodos lleva a un desafio de red, principalmente dado por la interconexión. Los switches son quienes dominan en estas redes, y STP muestra ser contraproducente para hacer uso eficiente de los recursos.
- **Firewall** Es un dispositivo de seguridad de red que monitorea el tráfico de red entrante y saliente y decide si permite el paso o bloquea un tráfico específico en función de un conjunto definido de reglas de seguridad. Los firewalls han sido una primera línea de defensa en seguridad de red durante más de 25 años. Establecen una barrera entre las redes internas seguras y controladas que pueden ser de confianza y las que no son de confianza fuera de las redes, como Internet. Un firewall puede ser hardware, software o ambos.[2]

2.2. mininet

Mininet [4] es un conocido simulador de redes, cuya aparición se debe a la necesidad de contar con un simulador capaz de operar con Switches OpenFlow. El ingreso revolucionario de las SDN y la posibilidad de correr simulaciones, llevo a un gran crecimiento de mininet, por lo cual hoy es muy sencillo encontrar gran cantidad de información disponible.

Mininet se utiliza de una forma radicalmente diferente al resto de los simuladores de red, ya que los dipositivos incluidos y su interconexión se definen a través de un script en python. En nuestro caso en particular, nosotros queremos poder simular un datacenter tal como se muestra en la Figura 3. Esta topología será provista por la catedra, y se encuentra lista para usar en la VM. Para poder correr esta topología, usando STP a modo de prueba debemos hacer lo siguiente.

1. Abrir una terminal e iniciar el controlador, indicándole que los switches correrán STP (la VM tiene por defecto la version **carp** de pox, para cambiar a la version **betta**, solo es necesario cambiar al branch correcto 'git checkout betta')

```
$ pox/pox.py samples.spanning_tree
```

[2] Quê es un firewall?: `https://www.cisco.com/c/en/us/products/security/firewalls/what-is-a-firewall.html`

2. Abrir una nueva terminal e iniciar mininet con la topología de Mesh

```
$ sudo mn --custom ~/mininet/custom/mesh.py --topo mesh --mac --arp --switch
    ovsk --controller remote
```

2.3. Controladores de OpenFlow

Hemos hablando de OpenFlow y del controlador, pero no hemos mencionado aún como hacer uso del controlador, de manera tal de ejecutar rutinas. Aunque OpenFlow es el protocolo, por lo general los controladores corren aplicaciones las cuales les permiten hacer uso del protocolo. Estas aplicaciones cuentan con API, las cuales se encuentran disponibles en Java, Python y C, aunque ésta última ha sido dejada de lado paulatinamente durante los últimos años.

En la catedra proponemos hacer uso de un controlador POX, el cual tiene una API en Python, aunque esta la libertad de usar otros controladores ya sea Frenetic (Python) o Beacon (Java).

2.4. Balance de carga: Equal Cost Multiple Path (ECMP)

En la introducción se mencionó la RFC2991, en donde ante la posiblidad de que un router tenga más de un camino con el mismo costo para alcanzar un destino, cada *flujo* debe ser siempre enviado por el mismo camino. Esta técnica se llama Equal Cost Multiple Path (ECMP), y también es sumamente popular a la hora del balance de carga a nivel de capa de enlace dentro de los datacenters. Nuestro objetivo aquí será poder llevar a cabo esta técnica de manera tal de hacer uso de todos los enlaces, y principalemente de manera equitativa. La Figura 4 muestra como entre h1 y h5 existen dos caminos con exactamente el mismo costo, por lo cual, dado un flujo, el camino debe ser el mismo. Sin embargo, ante flujos distintos entre h1 y h5 se puede hacer uso del camino restante.

La técnica ECMP es de las técnicas más difundidas por su simplicidad

2.5. Link-Layer Data Protocol

Entonces, ahora nos estamos proponiendo no utilizar STP, y a su vez, que los switches sean capaces de enviar paquetes por diferentes puertos de acuerdo al costo de llegar a un destino. Si miramos detenidamente nos daremos cuenta que este se trata de un problema de ruteo clásico, entonces es valido que nos hagamos las próximas preguntas: ¿Cómo sabe cada switch a donde tiene que enviar cada trama? ¿Cómo puede conocer el costo de los caminos? Más aún, para conocer el costo de los caminos ¿cómo conoce la topología?

La problemática de hacer un sistema de ruteo distribuido es resuelto de diferentes maneras, ya sea OSPF, RIP, BGP o el mismo STP, no obstante todos requiren el envio de mensajes de comunicación entre los equipos. En el caso de las SDN, los switches intercambian tramas denominadas Link-Layer Data Protocol[3] (LLPD), por las cuales pueden conocer la topología completa. De esta manera con los LLDP se puede correr Dijkstra, y sobre esto hacer ECMP.

3. Preguntas a responder

1. ¿Cuál es la diferencia entre un Switch y un router? ¿Qué tienen en común?
2. ¿Cúal es la diferencia entre un Switch convencional y un Switch OpenFlow?
3. ¿Se pueden reemplazar todos los routers de la Intenet por Swithces OpenFlow? Piense en el escenario interASes para elaborar su respuesta

4. A entregar

Los grupos (no más de 3 personas), deberán elaborar un código capaz de simular el funcionamiento de un datacenter. Esto incluye, el desarrollo de una topología del tipo *FatTree*, y del controlador el cual deberá funcionar utilizando todos los enlaces, en particular haciendo uso de la técnica ECMP. El controlador, ademas de ser capaz de distribuir el tráfico de forma uniforme entre todos los enlaces disponibles, deberá tener la capacidad de reconocer y mitigar un ataque de denegación de servicios por medio de la implementación de un Firewall.

El día de la entrega se deberá ejecutar el controlador y la simulación en clase, mostrando su funcionamiento y dando evidencias de que se usan todos los enlaces.

[3] `https://en.wikipedia.org/wiki/Link_Layer_Discovery_Protocol`

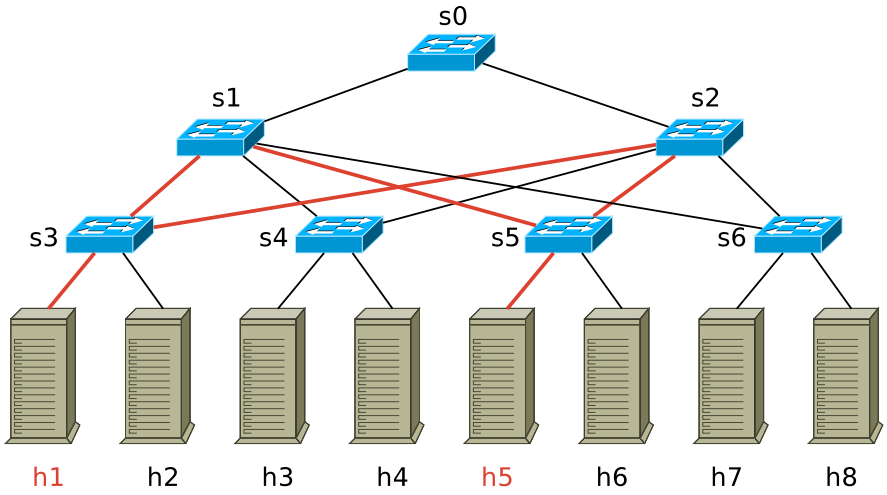


Figura 4: .

Se pedirá entregar el código y se exige no distribuirlo. Facilitar la distribucuón de código, como así también el uso de código desarrollado por otros, van en contra de la conducta que la Facultad de Ingeniería pretende de los alumnos. Más aún, nuestro objetivo en la catedra es fomentar el aprendizaje y el interés por temas actuales e innovadores, por lo cual pretendemos que el desarrollo del trabajo sea motivador y enriqucedeor para los alumnos.

4.1. Topología

La topología a desarrollar deberá ser configurable, siendo el parámetro, la cantidad de niveles del árbol. La raiz del mismo es el nivel 0, y las hojas el nivel $2^{(alturadelarbol-1)}$. La raiz debe tener conectada 3 hosts que funcionarán como clientes de nuestros datacenter, cada una de las raices a su vez, tendrán conectadas un host cada una, que funcionará como proveedor de contenido.

En la figura 5 se muestra un ejemplo de como sería la topología con un árbol de altura 3.

Los tres hosts clientes se comunicarán con los hosts del datacenter usando los protocolos ICMP, TCP y UDP. Para estos úlmitos dos se utilizará la herramienta *iperf* vista en clase.

4.2. Controlador

El controlador debe ser capaz de distribuir el tráfico uniformemente considerando todos los conceptos aprendidos hasta el momento. Para ello debe indicar al dispositivo de que forma llenar su CAM table, haciendo uso de la técnica ECMP y teniendo en cuenta el concepto de flujo antes visto.

4.2.1. Firewall

Se debe desarrollar un Firewall capaz de mitigar ataques de denegación de servicios distribuido. Para simplificar la implementación, solo se considera a un ataque de denegación de servicios cuando la cuota de paquetes UDP que se reciben con un mismo destino, excede un limite predefinido por unidad de tiempo. Es indiferente si los paquetes provienen de uno u otro host. Frente a esta situación, se debe aplicar la técnica de blackholing, evitando que los paquetes alcancen los hosts que proveen el contenido. Si luego de cierto tiempo, el caudal de paquetes disminuye, se debe permitir el transito de los mismos nuevamente.

5. Informe

El informe debe seguir un formato adecuado y el mismo debe contener como mínimo las siguientes secciones:

1. Introducción teórica
2. Objetivo

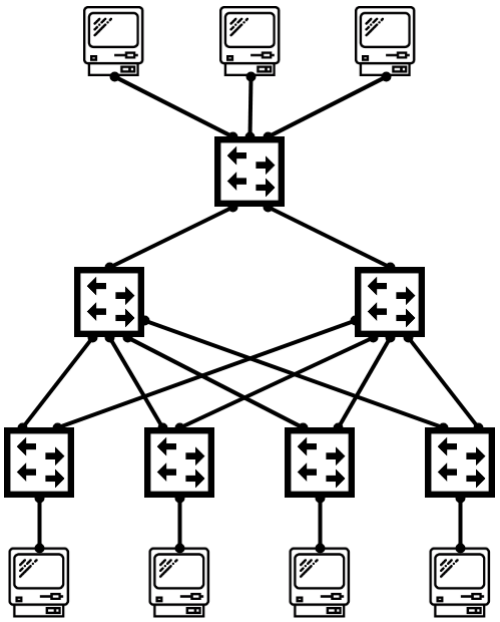


Figura 5: .

3. Desarrollo
4. Pruebas realizadas
5. Conclusiones
6. Anexos (Código)

6. Links Útiles

- Mininet: `http://mininet.org/walkthrough/`
- pox: `https://noxrepo.github.io/pox-doc/html/`
- Openflow: `https://www.opennetworking.org/technical-communities/areas/specification/open-datapath/`
- Open vSwitch: `http://www.openvswitch.org/`
- Visualizador de topologías: `http://demo.spear.narmox.com/app/?apiurl=demo#!/mininet`

Referencias

[1] D. Thaler and C. Hopps. Multipath Issues in Unicast and Multicast Next-Hop Selection. RFC 2991 (Informational), November 2000.

[2] Nick McKeown, Tom Anderson, Hari Balakrishnan, Guru Parulkar, Larry Peterson, Jennifer Rexford, Scott Shenker, and Jonathan Turner. Openflow: enabling innovation in campus networks. *ACM SIGCOMM Computer Communication Review*, 38(2):69–74, 2008.

[3] Mohammad Al-Fares, Alexander Loukissas, and Amin Vahdat. A scalable, commodity data center network architecture. In *ACM SIGCOMM Computer Communication Review*, volume 38, pages 63–74. ACM, 2008.

[4] Bob Lantz, Brandon Heller, and Nick McKeown. A network in a laptop: rapid prototyping for software-defined networks. In *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*, page 19. ACM, 2010.