FACULTAD DE INGENIERIA
Universidad de Buenos Aires

Trabajo Práctico 3: Software Defined Networks

[75.43] Introducción a los Sistemas Distribuidos
Segundo Cuatrimestre de 2020

Apellido y Nombres	**Número de padrón**	**Correo electrónico**
Bellaera, Leonardo	100973	lbellaera@fi.uba.ar
Fandos, Nicolás	101018	nfandos@fi.uba.ar
Ferreyra, Javier	100680	jferreyra@fi.uba.ar
Gatti, Nicolás	93570	gatti2602@gmail.com
Penna, Sebastián	98752	spenna@fi.uba.ar
Torraca, Facundo	101046	ftorraca@fi.uba.ar

Índice

1. Introducción

En el siguiente informe, realizamos una descripción de lo estudiando, analizado e implementado para resolver el trabajo sobre el desarrollo de una red definida por software (SDN). El problema se basaba en crear una topología de tipo Fat-Tree, y luego utilizar OpenFlow para manejar los múltiples enlaces de la red. A su vez el controlador debe poder detectar flujos de paquetes que compartan origen, destino y protocolo y asignarles algún camino mínimo específico que recorran durante un tiempo determinado. La idea es que ante la posibilidad de elegir distintos caminos mínimos alternativos, estos sean seleccionados de forma tal que se logre un balance de carga a nivel de capa de enlace para que los links sean usados de manera equitativa (usando la técnica conocida como Equal Cost Multiple Path, o ECMP).

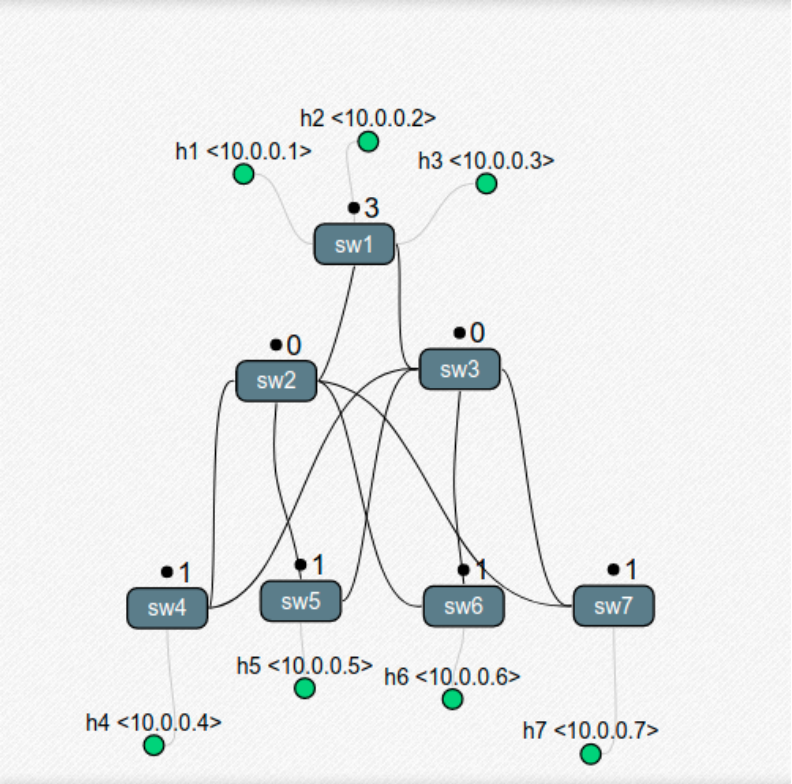

Figura 1: Ilustración de topología Fat-Tree

La estructura del trabajo se desarrolló y probo mediante las herramientas POX y Mininet. En particular Mininet provee lo necesario para generar, probar y utilizar la topología implementada, mientras que a través de POX se utiliza OpenFlow en los controllers y switches configurados.

El código del programa adjunto se encuentra dockerizado y puede ejecutarse usando en una terminal de linux los siguientes comandos:

```
$ docker-compose up -d
```

```
$ docker-compose exec mininet /tmp/pox/pox.py controller --ttl=60
```

El último parámetro –ttl es propio y sirve para asignarle un tiempo de vida a las reglas en las tablas de los switches. En caso de omisión se le asigna el valor por defecto (60 segundos).

En una segunda terminal ingresar y ejecutar lo siguiente para correr mininet con la topología de Fat-Tree:

```
$ docker-compose exec mininet mn --custom /tmp/topology/fat_tree.py --topo
↪ fat_tree,3 --mac --arp --switch ovsk --controller remote
```

En el comando luego de ingresar el parámetro –topo se le asigna el nombre de la topología seguido de la cantidad de niveles que se quiere que tenga el Fat-Tree, para que tenga más o menos niveles reemplazar el 3 por otro numero. En caso de omisión se toma el valor por defecto 3.

2. Comparación de SDNs y Openflow contra protocolos de ruteo clásicos

En el esquema clásico el ruteo se hace a través de algoritmos que corren en cada router, es decir, cada router es responsable por mantener su tabla de forwarding actualizada. Esto implica que dentro de cada router se debe destinar recursos al procesamiento y calculo de estas tablas, lo que además implica que cada router debe sondear la red para refinar y mejorar las decisiones que toma.

El uso de un controlador centralizado, utiliza un protocolo estandarizado, normalmente Openflow, para configurar los switch y para recibir el estado de los mismos. Dado esto permite centralizar la toma de decisiones de la red y hacer switches con funciones mas simples y especializadas. En este modelo el controlador es quien mapea la red y toma las decisiones de ruteo.

Entre las ventajas de este modelo se pueden destacar:

- En una SDN el controlador esta hecho en software, por lo que el mismo se puede modificar para ser mejorado o responder a cambios de configuración prácticamente a demanda, ya que solo implica compilar una nueva versión del software en el servidor del controlador. En cambio, en el modelo clásico esta tarea implicaba reprogramar (o reemplazar) cada switch en la red.

  Así mismo, el software puede ser implementado en un servidor, en una granja de servidores o en un cluster tercerizado, dependiendo de las necesidades. En el modelo tradicional esto no es posible.

- En el modelo clásico el ruteo se usa analizando la información de la capa de red. En el modelo de una SDN el controlador puede enrutar un switch utilizando información de las distintas capas, pudiendo identificar flujos de comunicación entre los distintos host y utilizar estos campos para generar caminos mas eficientes.

- En el modelo de SDN, el controlador ofrece una API para conectar distintas app de clientes, lo que permite ampliar las capacidades de monitoreo y configuración de la red sin afectar el funcionamiento de los switches.

Como corolario de las ventajas se puede indicar que al estar construido en software independiente, se produce la separación de las tareas de ruteo, o Dataplane, de las tareas de determinación de los caminos y configuración de los switches, o control plane. Esto permite independizar al fabricante de hardware del de software y brindar un ecosistema mucho mas rico de opciones y configuraciones de acuerdo a las necesidades.

3. Suposiciones y/o asunciones tomadas para realizar el trabajo

Durante el desarrollo del trabajo se establecieron y siguieron las siguientes asunciones:

- Todos los host se encuentran dentro de la misma subred y pueden verse entre si, los switches no controlarán el trafico a nivel de subred.

- Consideramos que un tiempo adecuado por defecto para que una regla establecida en la tabla de routing se mantenga es 60 segundos. Pasado este tiempo si el flujo continua este tomará, si existiera, una nueva ruta mínima la cual se respetará por los siguientes 60 segundos. Este tiempo es parametrizable y puede cambiarse usando "–ttl=n" en el comando de ejecución del controlador, reemplazando n por un número cualquiera.
- Para determinar una ruta entre un host y otro se definió a la red de switches como un grafo no dirigido no pesado y se tomo un camino mínimo que se define de forma aleatoria en caso de tener varias alternativas. El algoritmo utilizado es una implementación similar al algoritmo de BFS (Breadth First Search).
- Al caerse o insertarse un nuevo link (enlace) en la red, en esta se reiniciarán las tablas de routing para definir nuevas rutas de flujos que puedan incluir al enlace nuevo en el caso de que haya sido insertado o bien si fue eliminado entonces asegurarnos de que no se intente enviar algún paquete por un enlace que ya no exista.
- Para balancear la carga a la hora de elegir un camino mínimo, este es seleccionado de manera random usando la estructura de grafo mencionada en un punto anterior, de esta forma si la red fuera lo suficientemente grande no debería haber ningún camino que tenga prioridad sobre otro a la hora de establecer la ruta de un flujo.
- Consideramos únicamente el manejo de paquetes IPv4, descartando a los que no pertenezcan a este protocolo.

4. Dificultades encontradas

- Al desconectar un nivel entero y dejar la red no conexa para luego pasado un tiempo volver a introducir un switch que vuelva a conectar a la red, a esta le lleva un tiempo considerable (aproximadamente entre 30 y 60 segundos) volver a detectar todos los links que unían a la red para que vuelva a funcionar un **pingall**. Por otro lado si la red nunca llega a desconectarse y en los niveles hay al menos un switch entonces todo funciona correctamente.
- Si no se tiene una computadora con suficiente capacidad de procesamiento y memoria puede pasar que si la red es muy grande (el fat tree tiene muchos niveles) el controlador falle al detectar todos los links y no se pueda realizar un pingall. Esto lo notamos en un caso en el que en una computadora con menores recursos no podía correr correctamente una red de 6 niveles o más pero al ejecutar el mismo código en una computadora más potente todo funcionó correctamente.
- Como el controlador tarda un tiempo en aprender todos los links (enlaces), puede fallar algún envío de mensajes entre hosts si se realiza muy rápido antes de que termine de descubrirlos. De todas formas, con esperar unos segundos alcanza para que todo funcione correctamente.

5. Conclusión

Este trabajo nos llevó a investigar el protocolo de switching OpenFlow y nos ayudó a familiarizarnos con algunos de los principales conceptos que se utilizan hoy en día en el ruteo de paquetes en grandes servidores. Podemos afirmar que se logró cumplir el objetivo del trabajo ya que se implementó un controlador que logra aprender la topología de la red en base a los mensajes OpenFlow que arriban, y logra proveerle a los switches las reglas para sus tablas de flujos calculando los caminos mínimos para que los paquetes lleguen a destino.