75.43 - Introducción a los sistemas distribuidos

Software Defined Networks

Alumno	Número de Padrón	Email
Robles, Gabriel	95897	*grobles@fi.uba.ar*
Cerquetti, Franco	98695	*fd.cerquetti@gmail.com*
Blázquez O., Sebastián A.	99673	*sebastian.blazquez96@gmail.com*
Alvarez Windey, Ariel	97893	*ajalvarez@fi.uba.ar*

Repositorio de código:
https://github.com/aalvarezwindey/fiuba-iasd-tp3-sdn

Introducción

En este trabajo se buscó emular la red de un datacenter a fin de poder estudiar y utilizar herramientas necesarias para la implementación de redes definidas por software (SDNs).

En particular, se desarrolló una topología de red del tipo *FatTree*, donde se tiene un punto de entrada al datacenter siendo este el nodo raíz del árbol y los hosts del datacenter ubicados uno en cada hoja del árbol (siendo los nodos los switches de la red). En esta topología la cantidad de nodos en cada nivel aumenta de manera exponencial, duplicando la cantidad de nodos en el nivel anterior. Además cada nodo, tiene un enlace directo a todos los nodos del nivel anterior y todos los nodos del nivel siguiente. Con la particularidad que el nodo raíz se conecta a tres hosts (simulando los clientes o consumidores del datacenter) y los nodos hoja a un único host (simulando los servidores).

SDNs y Openflow vs. Protocolos de ruteo clásicos

La principal diferencia que podemos encontrar en ruteo clásico y SDNs, es que el control plane y el data plane, en el caso de SDN, se encuentran separados, los routers interactúan con un controlador externo mediante el protocolo de Openflow para actualizar sus forwarding tables, en este caso llamadas flow tables, y no influyen en el cálculo de las mismas en ningún momento. En cambio, en los protocolos de ruteo clásico, el control plane y data plane se implementan dentro de un mismo dispositivo, los algoritmos que utilizan son establecidos por los fabricantes, por lo que no hay mucha flexibilidad al momento de establecer las reglas para las forwarding tables, ya que dependen únicamente de la dirección IP destino. La ventaja principal de las SDN sobre protocolos de ruteo clásicos es que ofrecen una gran flexibilidad al momento de configurar las flow tables, ya que tiene a su disponibilidad los diferentes headers de la capa de transporte y de red, por lo que se puede realizar un mejor balance de carga o un ruteo dependiendo una gran variedad de atributos, lo que permite a los ISPs (o cualquier administrador) un mejor manejo de su red y sistemas autónomos.

Quizás algo a notar es que OpenFlow viola lo que la arquitectura de capas propone. No es agnóstico a los otros protocolos del stack de protocolos de Internet. Esto de hecho se ve reflejado en la implementación de un controlador OpenFlow, donde hay que ir desarmando un paquete para accediendo a datos de protocolos de nivel superior. Permiten gran flexibilidad a costa de una posible complejidad en dicha implementación.

Suposiciones y/o asunciones

A medida que se fue desarrollando el trabajo fue necesario tomar varias suposiciones dado el gran alcance que tiene OpenFlow en las configuraciones que uno puede hacer en su propia SDN. Esto fue necesario para mantener la implementación lo más sencilla posible, pero sin dejar de cumplir lo que indica el enunciado:

1. A la hora de recibir un paquete en un switch, es necesario abrir las distintas capas del stack de protocolo de Internet para conformar lo que se define en el enunciado como *flujo* (esto es la tupla de IP origen, IP destino, puerto origen, puerto destino, protocolo de capa de transporte). En esta "conversión" de paquete a flujo, se supuso que solo llegarán frames de *Ethernet*.
2. Cuando se fue probando los avances del trabajo, descubrimos que los paquetes PING no usan protocolo TCP o UDP, sino que ICMP. Por esto tampoco tienen un puertos asociados. Para estos flujos se forzó que sean tratados como si el puerto origen y destino sea el 7.[1]
3. En el algoritmo para la elección de un camino frente a *N* caminos posibles para un flujo determinado se implementó de la siguiente manera.
    a. Dado los *N* caminos, se repartió un porcentaje de esos caminos para UDP y otro porcentaje para TCP.
    b. Luego frente a ese subconjunto se toma uno dependiendo el valor de la suma de los puertos (no se tuvo en cuenta el valor de las IPs)

    La suposición en este caso es que a nuestro datacenter llega igual cantidad de tráfico TCP que UDP, con lo que se reparten los caminos posibles totales en igual cantidad para ambos protocolos. Pero este valor sería fácilmente configurable (por ejemplo, si se observa que el 90% del tráfico de nuestro datacenter es TCP, posiblemente sea acertado balancear los flujos TCP en el 90% de los caminos posibles).
4. Suponiendo que el tráfico UDP sería menor, los paquetes ICMP eligen caminos entre los caminos posibles para UDP, a fin de no sobrecargar los caminos de TCP.
5. Suponiendo que siempre habrá un único host en cada switch hoja, dado el modo en cómo se inicializa la red, podemos suponer que en estos switches el host estará conectado al puerto de salida del switch de numeración *k+1*, siendo *k* la cantidad de enlaces que tiene ese switch con otros switches (es decir la cantidad de switches del nivel anterior). Este valor es fácilmente calculable si sabe la cantidad de niveles del árbol.
6. Para simplificar el problema, supusimos que no sabemos a priori en qué puertos del switch raíz están conectados los hosts "clientes" del datacenter. Por esto mismo, podrá iniciar una comunicación un host servidor con un host cliente sólo si antes la inició el cliente. Esto tiene lógica y realismo ya que los que inicial la comunicación son los clientes.

Dificultades encontradas

Entre las dificultades encontradas se destacó:

1. Desconocimiento de cómo probar ciertos eventos o cómo ir probando lo que se iba desarrollando.
2. Dificultad para dividir las tareas entre los integrantes del equipo, las tareas que se pudieron dividir estaban muy relacionadas entre sí y no se podía avanzar en paralelo.
3. Ya sea por falta de detalles en el enunciado o el desconocimiento de cómo se debía ir probando los avances y por lo tanto tener una base sólida de que lo desarrollado

funcionaba correctamente, se tuvieron que hacer muchas suposiciones sin saber si las mismas fueron correctas.

4. El manejo de los paquetes ICMP no fue sencillo de implementar. Si bien el enunciado no lo especificaba, creímos que posiblemente el desarrollo sea sometido a pruebas con el comando *ping* y por eso se buscó manejar estos paquetes.
5. El algoritmo de búsqueda de caminos fue difícil de diseñar. Evaluamos utilizar algún algoritmo como Dijkstra, aunque finalmente nos decidimos por una opción más sencilla y con diseño propio. Dicho algoritmo parte de los switches en el nivel más bajo y busca los links hacia el nivel superior, hasta llegar a la raíz. De esta forma logramos que dicho algoritmo tuviera una complejidad óptima, sin recurrir a librerías externas.

Conclusión

En el marco del Control Plane de la capa de red, estudiamos que existen mecanismos de Routing tradicionales, en donde el contenido de las forwarding tables de los routers es determinado por algoritmos que se ejecutan en los mismos routers, y normalmente el forwarding se hace en base a la dirección de destino (destination-based forwarding).

Estudiamos también que se puede realizar esto mismo con un enfoque más moderno: SDN, que propone separar el control plane en un controller remoto. Ahora las forwarding tables son calculadas y distribuidas a los routers por este controller. Surge entonces el concepto de OpenFlow, un estándar que revolucionó las SDN en general, permitiendo extraer el match-plus-action forwarding en el controlador y exponer una API para su uso. De esta forma se puede realizar en forwarding en base a varios campos de varias capas del stack de protocolos de internet.

A partir del trabajo práctico pudimos mirar de cerca el protocolo OpenFlow y aplicar algoritmos de ruteo para balancear la carga en un topología *Fat Tree*. Se calculó la forwarding table para cada switch en base a un flow dado (tupla ip origen, ip destino, puerto origen, puerto destino y protocolo). Nos pareció que en topologías sencillas como la planteada, se saca mucho provecho al tener un controlador central en el que se ejecuta el algoritmo de ruteo. Esto permite implementar de una forma relativamente sencilla un algoritmo ECMP que incluso pueda balancear de forma óptima el tráfico de datos. Por el contrario, con un algoritmo de ruteo clásico sería mucho más complicado lograr el mismo resultado.

Referencias

1. https://networkengineering.stackexchange.com/questions/37896/ping-port-number