Redes de Computadoras

Curso 2017

Grupo 76

Informe - Obligatorio 3

Autores:
Joaquín Villar
Federico Luongo
Federico Godoy

Supervisores:
Martín Giachino
Jorge Visca

November 13, 2017

Contents

1 Introducción **2**

2 Parte A **3**

2.1 Implementación 3

2.2 Pruebas y resultados 4

2.3 Solución encontrada y Comentarios 5

3 Parte B **6**

3.1 Implementación 6

3.2 Pruebas y resultados 7

3.3 Solución encontrada y Comentarios 8

1 Introducción

En esta tarea se realizó la implementación de un alogritmo de enrutamiento de vector distancias y un alogritmo de enrutamiento de estado de enlace. Para esto se realiza una introducción para comprender el funcionamiento de cada uno de ellos y sus diferencias. Ambos protocolos pertenecen a la clase llamada IGP(Interior Gateway Protocol) que representa a un conjunto de protocolos de enrutamiento dinámico que se utilizan para el intercambio de información relacionada con el anuncio de rutas dentro de un sistema antónomo. Definiendo como sistema autónomo a un grupo de redes IP, como por ejemplo un ISP de una empresa. Estos protocolos de enrutamiento implementan el esquema de comunicación entre los routers, un protocolo permite que un router comparta información con otros routers acerca de las redes que conoce así como de su proximidad y costo hacia otros routers. La mayoría de los protocolos pertenecen a alguna de estas dos categorías.

Vector Distancia

En este tipo de protocolos se involucran dos factores la distancia de un router origen a sus destinos y un vector que indica que dirección debe tomar para llegar a ellos. Es un algoritmo distribuido, esto implica que la información de enrutamiento solo se intercambia entre vecinos que estan conectados directamente. Cada router que aprende una ruta sabrá quien se la hizo saber pero sin embargo no sabrá como su vecino aprendió dicha ruta es por esto que el router no conoce la topología de la red mas allá de sus propios vecinos. Para esto los routers envían periodicamente copias de las tablas de enrutamiento entre vecinos implica que es iterativo y se realizan cálculos con la suma de los vectores distancia de router a router. Los algoritmos correspondientes al protocolo de vector distancia se basan en la ecuación de Bellman-Ford. Debido a esta naturaleza se presentan ciertos problemas con bucles de enrutamiento como el de conteo infinito que se detallará en la seccion de pruebas de la parte A. Para solucionar esto existen algunos mecanismos como: Poison Reverse, Split Horizon, Triggered Updates, Max Hop Count y Hold down timers. En particular en este laboratorio se implementó la técnica de Poison Reverse.

Estado de Enlace

En este tipo de protocolos se requiere que todos los routers conozcan todos los caminos existentes para llegar a todos los routers de la red. Es necesario mantener una base de datos con la información de la toplogía de la red, a partir de la misma se permite que todo router que utilice dicho protocolo se ejecutará un algoritmo que determinará el camino mínimo para llegar al resto de los routers de la red poniendo como origen el router donde se esté ejecutando el algoritmo. Generalmente para obtener la ruta más corta a otros routers se utiliza el algoritmo Dijkstra. Como cada rotuer comparte la misma base de datos a medida que se incrementa el tamaño de la red se pordría llegar a pensar que estos algoritmos exigen de cierta manera a los dispositvos que los ejecutan. Sin embargo debido a que todos los routers tienen la misma información esto la convierte en una operación mas robusta y escalable a redes de dimensiones más grandes.

2 Parte A

2.1 Implementación

En esta parte se implementó algoritmo de vector distancias el cual esta basado en la ecuación de Bellman-Ford, también se implementa la técnica de reversa envenenada para resolver el problema de conteo infinito y de esta forma resolver de forma más rápida la convergencia entre nodos. A continuación se presentan los detalles de la implementación realizada en la clase RouterNode que representa a un nodo en la red. Se utiliza un HashMap de enteros que representa los costos a los vecinos de dicho nodo (costs), de la misma forma se tiene un HashMap donde se registra el primer nodo por el cual debe pasar para llegar a un destino específico (routes). En el HashMap (neighbours) que esta definido igual que los anteriores se guardan los id de los vecinos. En la estructura vectors se mantiene el costo general a todos los nodos de la red, se lo implementó como un HashMap doble donde la primer clave representa el id del nodo y su valor correspondiente es otro HashMap que contiene los costos a los demás nodos. Es necesario aclarar que se decidió utilizar las estructuras HashMap para poder tener un tamaño variable de cantidad de nodos cuando se realicen las distintas ejecuciones del Simulador. En ejecución estas estructuras utilizan una variable MaxNode a la hora de ser

recorridas, es importante indicar que esta variable se actualiza cada vez que se está en conocimiento de un nuevo nodo. Por último se cuenta con un booleano que permite habilitar o no la ejecución o no del algoritmo con reversa envenenada. Cuando se da de alta el nodo en el sistema se cargan las estructuras con los costos recibidos cuando se instancia la clase, una vez cargadas las estructuras se ejecuta al método SendNeighbours. En este procedimiento se envía a todos los vecinos del ruter que están guardados en la estructura neibohurs el nuevo hash de costos. Luego cuando un router recibe un costos de un vecino se encarga de actualizar la tabla de vectors para luego re calcular sus costos a partir de los nuevos costos recibidos. En el caso de que la reversa envenenada este deshabilitada luego de calcular los costos se envían estos a los vecinos, si la reversa esta habilitada se envían los costos a sus vecinos pero en caso de que alguno de sus costos están calculados a través del vecino al que se van a enviar los costos este costo se pasa como infinito. Por ultimo en el procedimiento updateLinkCost se actualiza el costo de uno de los vecinos del router y luego se re calculan los costos y se envía a los vecinos.

2.2 Pruebas y resultados

Para testear el funcionamiento de este algoritmo en primer lugar se utilizaron las pruebas provistas en el EVA, estas inculyen tres escenarios que modelan una red con 3 , 4 y 5 nodos. Para comenzar se realizaron pruebas sobre los tres escenarios deshabilitando la opción de cambios de costos, en esta prueba inicial se comprobó que los vectores de distancias devueltos por el algoritmo se correspondieran con las topologías en cada caso. Una vez corregidos alunos defectos encotrados se pasó a verificar el funcionamiento del método UptdateLinkCost, en este caso se habilitó la opción de realizar cambios en los costos de las aristas y se verificó que los cambios fueron procesados correctamente por el algoritmo y los vectores de distancia coincidan con los cambios realizados. Luego se probó la funcionalidad de la reversa envenenada, se decidió ejecutar cada prueba sin la opción de reversa envenenada y luego habilitando dicha opción. Luego de estas pruebas se observaron diferencias en la cantidad de iteraciones ejecutadas por el algoritmo en cada caso, se pudo verificar que la reversa envenenada hace mas eficiente el algoritmo dado que la cantidad de iteraciones se reduce sensiblemente evitando los loops de enrutamiento

producto de un cambio de costo en un enlace. Específicamente en el escenario de simulación de tres nodos la ejecución del algoritmo sin reversa envenenada se realiza en 319 milisegundos mientras que hablitando esta opción el tiempo se reduce a 54 milisegundos aproximadamente. Con esta funcionalidad resolvemos el problema de conteo infinito que se presenta en el escenario de tres nodos, en este caso se ven involucrados dos nodos vecinos en el bucle pero cuando el bucle implica a tres o más nodos la reversa envenenada no resuelve el problema. Para verificar esto se realizó la prueba indicada en la figura 1 en la cual se actualizó el costo a infinito del enlace entre los nodos 0 y 3 y en este caso comprobamos que el algoritmo queda en loop y no finaliza su ejecución y por lo tanto la técnica de reversa envenenada no resuelve el problema en el escenario planteado.

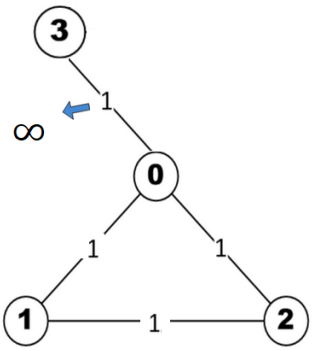

Figure 1: Bucle de 3 nodos

2.3 Solución encontrada y Comentarios

Para este algoritmo surgieron distintos problemas en el momento de la implementación, uno de ellos fue que en el momento de re calcular costos si únicamente se tiene en cuenta los costos de nuestros vecinos, en este caso se perdían los costos de ir a los vecinos. Para esto se guarda una estructura con los costos de los vecinos y antes de re calcular se fija si en los costos actuales si el costo de ir a un vecino es mayor al costo de la arista original del vecino entonces se cambia este costo y la ruta. Esto solucionó el problema que se perdía el calculo con los costos originales de ir a un vecino directamente.

3 Parte B

3.1 Implementación

En este algoritmo se utilizaron distintas estructuras para poder representar y mantener la información de toda la topología en cada router. Por ejemplo se utilizo una matriz variable en la que se guarda una fila por cada router de la topología que se va descubriendo y sus costos. Luego se utiliza una estructura igual a la anterior en la que se guarda una fila por router que se va descubriendo y los costos que tiene de ir a sus vecinos. Para almacenar los costos del router se utilizo un hash map en el cual se almacenan los costos que se van calculando para llegar a todos los routers, también se utilizo una estructura para guardar las rutas que se utilizan para ese costo. En lo que trata del algoritmo se utilizo la técnica de flowing para que a todos los routers reciban la información de toda la topología, esta técnica trata de que toda información que es recibida por un router este se lo envía a todos los routers vecinos y así sucesivamente hasta que todos obtuvieron dicha información. En nuestro caso lo implementamos de forma de que en caso de recibir costos y estos son distintos a los que antes se tenían para ese router entonces se aplica la técnica para todos los vecinos menos para el que router perteneciente de los costos, de esta forma nos aseguramos que se compartan todos los costos hasta que todos tengan la misma información. Para poder tener la información de los vecinos de cada router se utilizó la misma técnica pero para que se pueda identificar que se están enviando los costos de vecinos y no costos mínimos se agrega un campo mas en la estructura con valor negativo para poder identificar los costos de los vecinos, esto se utiliza la primera vez que se crea el router y cuando hay un update link costs. En el caso del update link cost se actualiza el costo de los vecinos, se envía estos costos como costos de vecinos a los vecinos, se re calcula los nuevos costos y luego se enviar los costos nuevos a los vecinos. Para re calcular los costos se utiliza el algoritmo de Dijkstra el cual necesita de toda la topología para poder implementarlo. En nuestro caso decidimos que para implementar el algoritmo primero asegurarnos de que el router ya recibió toda la topología para esto recorremos la estructura de vectores y nos aseguramos que esta exista para todos los routers encontrados. Para la implementacion del algoritmo de Dijkstra se utilizaron varias estructuras temporales en las cual se va almacenando los costos

mínimos encontrados para un camino en especial, es decir para encontrar el costo mínimo de un camino hay que recorrer todos los costos iniciales (costos de ir a los vecinos) y seleccionar el mínimo de estos por el cual no se haya pasado antes. Luego se selecciona el costo mínimo que tiene el router (el cual se llega a partir del camino mínimo elegido anteriormente) y se guarda este camino en los costos. Esto se ejecuta hasta que se haya pasado por todos los routers de la topología.

3.2 Pruebas y resultados

Al igual que en la parte A se comenzó probando el algoritmo sin realizar cambios en los costos hasta verificar que el algoritmo retorne resultados que se correspondan con las topologías de los tres escenarios previstos. Luego de esto se habilitó la opción de cambio de costos asociados a enlaces y en los 3 escenarios planteados no se encontraron problemas y se pudo observar que este algoritmo respondió mas rápidamente a los cambios de costo con respecto al de la parte A. Se implementó un caso en el cual se cambia el costo de una arista a infinito el cual representa la desconexión de un nodo en la red y en un evento posterior se deshace dicho cambio volviendo el costo al original, en este caso se observo que el dicho cambio modifica los costos y las rutas asociadas correctamente y luego del segundo evento de cambio se observan como las rutas y los respectivos costos convergen a la solución inicial esperada de forma correcta. Para este caso también se ejecutó la prueba de la figura 1 y como resultado el algoritmo converge al resultado esperado a diferencia del algoritmo de Vector Distancia que no finalizó. Como último caso de prueba se modificó el simulador de cinco nodos agregando dos nodos más conectados entre sí pero sin conexión hacia los nodos originales del escenario, formando de esta manera un grafo con dos componentes sin conexión entre ambas. La prueba consistió en generar un evento que conecte uno de los nodos de la componente conexa menor con un nodo perteneciente a la componente mayor y verificar que realmente el algoritmo devuelva como respuesta la representación los nodos de las dos componentes del grafo con rutas existentes entre la totalidad de los mismos. Vale aclarar que luego de esto se procedió a realizar la ejecución de este test en el algoritmo de la parte A y el resultado también fue satisfactorio. Por lo tanto deducimos que este algoritmo tiene aspectos positivos tales como: es más efectivo en cuanto a convergencia, cada nodo conoce la totalidad de la topología y no

presenta problemas de conteo infinito. Pero se es consciente que genera un mayor overhead dado que el almacenamiento e intercambio de información entre nodos es mucho mayor al de Vector Distancia.

3.3 Solución encontrada y Comentarios

En este segundo algoritmo la estructura de rutas es para un uso meramente ilustrativo, es decir para poder observar en pantalla la ruta que esta eligiendo para ese costo ya que no es necesario para el algoritmo tener esta información a diferencia de el algoritmo de la parte A que si lo necesita. Luego de finalizada la implementación del algoritmmo se puede identificar mejoras en la forma de actuar del algoritmo como por ejemplo en el caso de que este se base en un entorno en donde los routers pueden desaparecer y aparecer nuevos routers se debería implementar un timer donde se ejecute el flowing para que siempre este actualizado con la realidad y se pueda identificar cuando un router nuevo o uno desaparece y esto comunicarlo al resto de los routers. Otra mejora podría ser ejecutar siempre el algoritmo de re calcular aunque no se tenga el conocimiento de toda la topología, esto debería ver si es realmente más eficiente ya que estaría forzándose a calcular siempre sin tener todos los valores lo que podría generar que se creen falsos cálculos y generando costo computacional mayor. Dado que la topología puede cambiar continuamente esta mejora debería ser implementada aunque genere un alto costo computacional ya que sino puede darse el caso en que un router nunca calcule sus costos.

References

[1] https://eva.fing.edu.uy/mod/resource/view.php?id=71140

[2] RFC 2453. RIP v2. https://tools.ietf.org/html/rfc2453

[3] RFC 2328. OSPF v2. https://www.ietf.org/rfc/rfc2328.txt

[4] Redes de computadoras - Un enfoque descendente - Kurosse y Ross - 5ta Edición -