Sebastián González 18588

Laboratorio 3

Algoritmos de Enrutamiento

Descripción del laboratorio

Este laboratorio consiste en comprender e implementar algunos algoritmos de enrutamiento. Tiene como objetivos conocer los algoritmos de enrutamiento que se utilizan actualmente y comprender cómo funcionan las tablas de enrutamiento. Se utiliza alumchat.xyz para la simulación de esta práctica siendo cada uno de los usuarios un nodo que conforma la topología. Al final del laboratorio se realizará una prueba donde se verificará la funcionalidad de los algoritmos en clase.

Descripción e implementación de los algoritmos

Flooding:

Como su nombre lo indica, este algoritmo “inunda” la red con mensajes, ya que cada nodo al recibir un mensaje, lo manda hacia todos y cada uno de sus vecinos, exceptuando el que le mandó el mensaje a él, y repitiendo el proceso de mandar hacia todos los vecinos hasta llegar a su objetivo. Para la implementación de este algoritmo se toma el archivo con la topología y se obtienen los vecinos del nodo actual, luego tomamos la data del mensaje y la mandamos hacia todos estos vecinos, verificando que no se mande a la persona que nos la mandó a nosotros, siempre luego de hacer esto se verifica si el mensaje ya ha llegado a su destino.

Distance Vector Routing:

Método de enrutamiento popular utilizado por ARPANET, DECNET, IPX y Appletalk. Además lo usaba RIP el cual era el único que se utilizaba en el internet hasta 1988. Este algoritmo consiste en que cada router comparte su distance vector table cada cierto tiempo a solamente sus vecinos. Al recibir una tabla, el router compara la tabla que tenía con la que recibió y actualiza datos si es necesario. En esa comparación, el algoritmo utiliza la ecuación Bellman Ford.

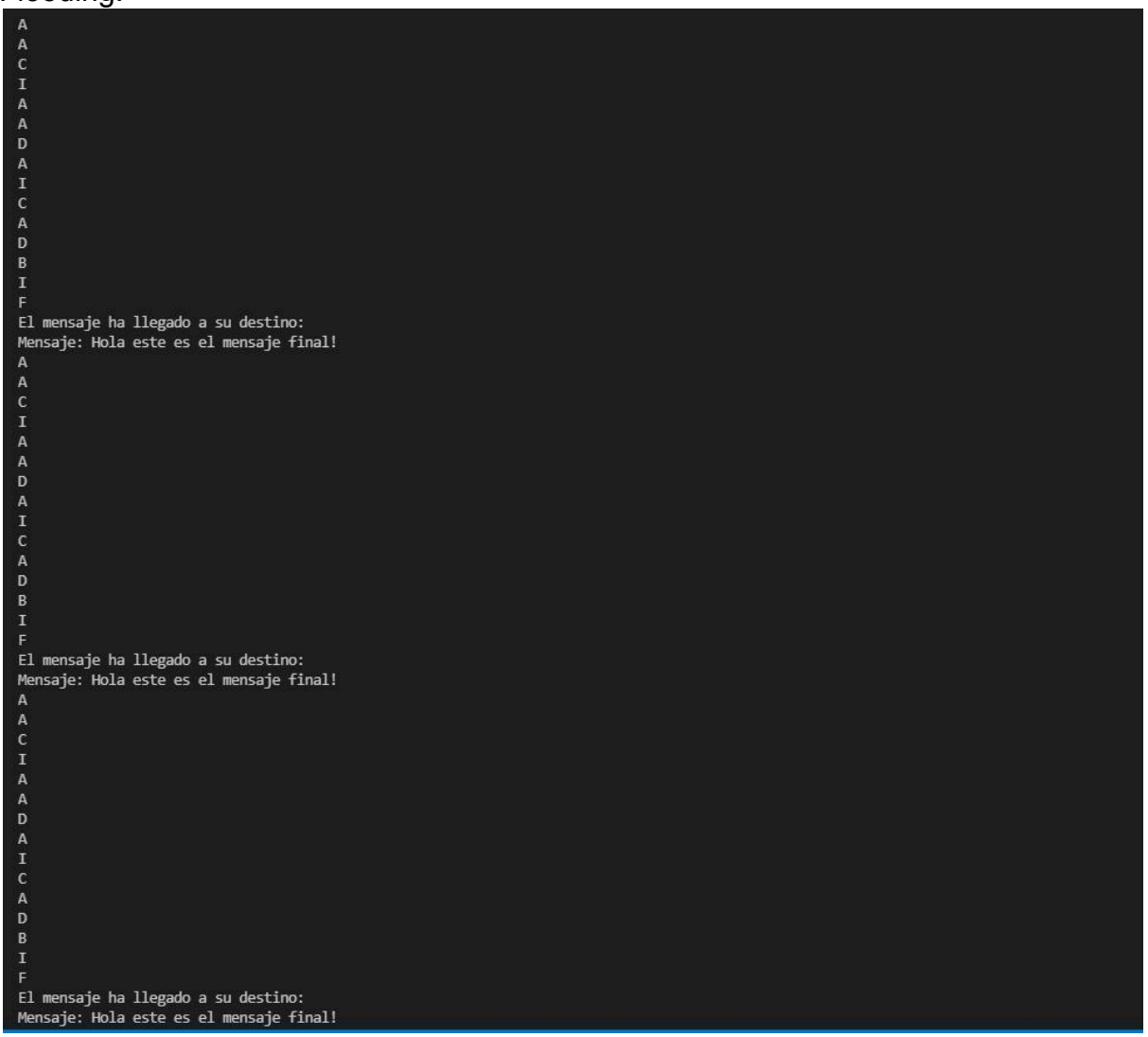
Esta ecuación básicamente determina si la distancia entre el nodo inicial y el destino es menor o mayor que la distancia entre el nodo inicial y el destino pasando por otro nodo intermedio. Si la distancia menor calculada es menor que la que tiene el router en su tabla, este la actualiza.

La implementación de este algoritmo se encuentra en el archivo clientDVR.py y un ejemplo en dvr-ejemplo.py donde se usan diccionarios de Python como las tablas de

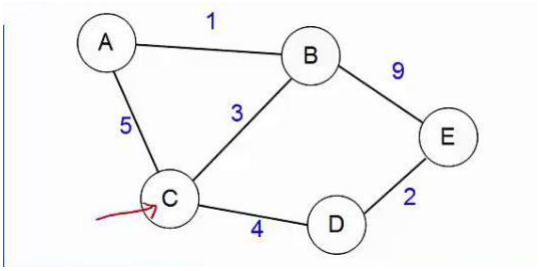
los routers y utiliza la fórmula de Bellman Ford para actualizarlas al recibir tablas de sus vecinos.

Resultados

Flooding:



DVR:



Topografía

```
ejem = {
    "A": 0,
    "B": 1,
    "C": 5,
}
ejem2 = {
    "A": 1,
    "B": 0,
    "C": 3,
    "E": 9
}
ejem3 = {
    "A": 5,
    "B": 3,
    "C": 0,
    "D": 4
}
ejem4 = {
    "E": 2,
    "C": 4,
    "D": 0
}
ejem5 = {
    "B": 9,
    "E": 0,
    "D": 2
}
```

Creación de nodos y enlaces

```
dvr = DistanceVectorRouter('sol', 'A')
dvr.table = ejem
dvr.BellmanFord(ejem3, 'C')
dvr.BellmanFord(ejem2, 'B')

dvr2 = DistanceVectorRouter('sol', 'B')
dvr2.table = ejem2
dvr2.BellmanFord(ejem3, 'C')
dvr2.BellmanFord(ejem, 'A')
dvr2.BellmanFord(ejem5, 'E')


dvr3 = DistanceVectorRouter('sol', 'C')
dvr3.table = ejem3
dvr3.BellmanFord(ejem2, 'B')
dvr3.BellmanFord(ejem, 'A')
dvr3.BellmanFord(ejem4, 'D')

dvr.BellmanFord(ejem3, 'C')
dvr.BellmanFord(ejem2, 'B')
```

Creación de routers e Iteraciones del algoritmo

```
{'A': 0, 'B': 1, 'C': 5, 'D': 9}
{'A': 0, 'B': 1, 'C': 4, 'D': 9, 'E': 10}
{'A': 1, 'B': 0, 'C': 3, 'E': 9, 'D': 7}
{'A': 1, 'B': 0, 'C': 3, 'E': 9, 'D': 7}
{'A': 1, 'B': 0, 'C': 3, 'E': 9, 'D': 7}
{'A': 4, 'B': 3, 'C': 0, 'D': 4, 'E': 12}
{'A': 4, 'B': 3, 'C': 0, 'D': 4, 'E': 12}
{'A': 4, 'B': 3, 'C': 0, 'D': 4, 'E': 6}
{'A': 0, 'B': 1, 'C': 4, 'D': 8, 'E': 10}
{'A': 0, 'B': 1, 'C': 4, 'D': 8, 'E': 10}
```

Discusión

En la imagen de resultados del flooding algorithm se puede observar los caminos que toma y cómo el mensaje que a pesar de que logra llegar a su objetivo, este puede llegar a ser un poco ineficiente, porque el mensaje es distribuido una cantidad muy grande de veces, y llega al destino más de una vez. Se recomendaría que una vez llegado el mensaje al destino ya no siguieran mandando los mensajes, sin embargo no existe una manera de comunicar a todos los nodos de que el mensaje ya ha llegado.

Para el algoritmo de distance vector routing se hizo una prueba local en el archivo dvr-ejemplo.py en donde se crearon nodos y enlaces entre dichos nodos para simular el grafo mostrado en la primera imagen. En la tercera imagen se puede observar que se instancian 3 routers A, B y C (también representados en el grafo). Luego se itera sobre el algoritmo dvr (Bellman Ford) en desorden para demostrar cómo cambian los valores de la tabla. Finalmente, se muestran las tablas calculadas como diccionarios en cada iteración. Se puede identificar de que router es cada tabla al ver que nodo tiene valor 0, ya que un nodo se asigna 0 a si mismo. Las primeras dos tablas pertenecen al nodo A y se puede apreciar que al recibir las tablas de C y B (sus vecinos) este actualiza los valores al nodo C y agrega al nodo E a su tabla ya que no era visible en la primera iteración. También se ve como los demás nodos actualizan sus tablas en base a la información que recibe y termina en dos iteraciones más en el nodo A en donde actualiza el valor de la distancia al nodo D. Se puede verificar que cada tabla tiene los valores mínimos para cada nodo.

Por último se puede ver en la última imagen como se implementó el algoritmo en alumchat. Esta vez los valores de las distancias entre nodos/usuarios es medido con tiempo. Un nodo manda un mensaje a sus vecinos los cuales proceden a contestarle de regreso para simular un mensaje de echo. Estos mensajes contienen información sobre qué nodo lo envió, el tipo de mensaje y el tiempo en el que se envió. Al recibir una respuesta el nodo actualiza su gráfica adecuadamente.

Comentarios

Los algoritmos implementados pueden llegar a parecer bastante ineficientes, sin embargo no hay mucho que se pueda llegar a hacer, ya que el hecho de que cada nodo no tenga el conocimiento de toda la topología limita bastante para que se pueda realizar un enrutamiento de manera óptima.

Es bastante interesante ver la implementación de estos algoritmos. Todos hacemos usos de algoritmos como estos diariamente pero no lo sabemos y no le ponemos atención a la complejidad y los obstáculos que pueden tener. También es interesante cómo van progresando estos algoritmos de enrutamiento mientras pasa el tiempo pero aun así todos tienen sus limitaciones.

Conclusiones

- Es importante lograr optimizar lo más posible los algoritmos de enrutamiento, como lo es por ejemplo el limitar enviar el mensaje a la misma persona de la cual lo ha recibido, ya que la complejidad que llegan a alcanzar estos es bastante alta.
- Todos estos algoritmos tienen sus ventajas y desventajas. El uso de cada uno depende mucho de la situación en las que se desean utilizar. Por lo mismo, la popularidad y frecuencia de uso de estos algoritmos han variado en el internet a través de los años.

Referencias

https://www.geeksforgeeks.org/fixed-and-flooding-routing-algorithms/

https://slixmpp.readthedocs.io/en/slix-1.6.0/getting_started/echobot.htm

l https://github.com/fritzy/SleekXMPP/tree/develop/examples

https://www.geeksforgeeks.org/distance-vector-routing-dvr-protocol/

https://www.geeksforgeeks.org/bellman-ford-algorithm-dp-23/