UNIVERSIDAD DEL VALLE DE GUATEMALA

CC3067 Redes

Sección 10

Ing. Jorge Yass

Laboratorio 3 - Parte 1
Algoritmos de enrutamiento

SEBASTIAN ARISTONDO PEREZ 20880

JOSE DANIEL GONZALEZ CARRILLO 20293

GUATEMALA, 31 de agosto de 2023

Descripción de la práctica

Durante la presente práctica se realizó la implementación de tres algoritmos para enrutamiento. Estos algoritmos fueron *Link State*, *Distance Vector* y *Flooding*. Se realizaron programas enfocados principalmente orientados a la funcionalidad del algoritmo. Los algoritmos se desarrollaron en un lenguaje de programación en el cual se tuviera experiencia para usar el protocolo XMPP, debido a que se utilizará en el futuro para simular los nodos de una red. En el laboratorio, estos nodos se simularon usando múltiples instancias de los programas, los cuales corrieron en distintas terminales.

Para evidenciar el funcionamiento del programa, se ejecutaron las distintas fases de cada algoritmo, de forma que se pudiera evidenciar el cálculo de las rutas según la estructura de la red y el paso de mensajes de un nodo a otro, dependiendo del enrutamiento óptimo.

Previamente se indicó que la práctica se enfocó en el funcionamiento del algoritmo. Para fines del laboratorio y de realizar la implementación, se definió una estructura única para hacer las pruebas, la cual estaba “hardcodeada” en la clase del algoritmo de Link State. Sin embargo, para la parte 2 del laboratorio se implementará un flooding entre nodos de la red, para conocer la estructura de esta. Para el algoritmo de Distance Vector si se realizó el proceso de compartir las tablas de enrutamiento entre nodos por medio de la terminal.

Descripción de los algoritmos

El algoritmo de "flooding" es un método utilizado en redes informáticas para enviar un mensaje o paquete de datos desde un origen a todos los nodos en una red. La característica principal de este algoritmo es que el mensaje se reenvía a todos los nodos vecinos sin ninguna consideración sobre si el nodo ya ha recibido el mensaje anteriormente. Esto asegura que el mensaje se propague por toda la red, pero puede dar lugar a problemas como bucles infinitos si no se gestionan correctamente.Para lograr esta mensajería se siguen los siguientes pasos: Nodo de origen decide enviar un mensaje, reenvío de mensajes a nodos vecinos, recepción y reenvío de mensajes.(Terrell Hanna, 2021)

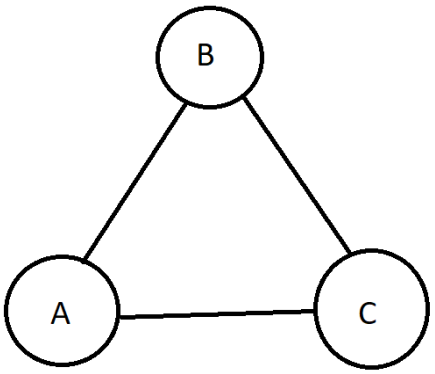
El algoritmo de vector de distancia (Distance Vector) es un enfoque utilizado en el ámbito de las redes informáticas para determinar las rutas óptimas para la transferencia de datos entre nodos en una red. Este algoritmo se utiliza en protocolos de enrutamiento, que son conjuntos de reglas y procedimientos que permiten a los routers o conmutadores tomar decisiones sobre cómo dirigir el tráfico de red de manera eficiente. Para lograr esta coordinación es necesario seguir los siguientes pasos: Inicialización del nodo, intercambio de información, Actualización de distancias, Actualización de vecinos y convergencia y evitar bucles y recepción de mensajes. (Antoniou, 2016)

El algoritmo Link-State (Estado de Enlace) es un algoritmo utilizado en enrutamiento de redes para determinar las rutas óptimas entre nodos en una red. Su objetivo principal es crear una imagen precisa y actualizada de la topología completa de la red, lo que permite a cada nodo calcular las rutas más cortas hacia todos los demás nodos utilizando el algoritmo de Dijkstra.

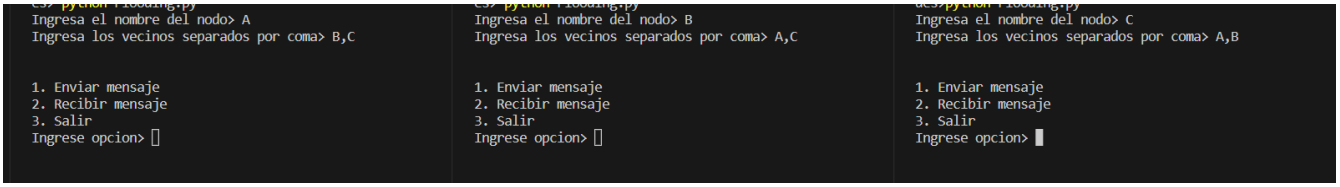
Los pasos para establecer las rutas son los siguientes: descubrimientos de vecinos y enlaces de cada nodo, compartir paquetes de estado de enlace a través de la red. Estos paquetes permiten que cada nodo calcule las rutas más cortas hacia todos los demás nodos utilizando el algoritmo de Dijkstra, formando así sus tablas de enrutamiento. Conforme cambia la topología de la red, los nodos actualizan sus paquetes y recalculan las rutas para mantener la precisión y eficiencia del enrutamiento.(WAN, 2023)

Resultados

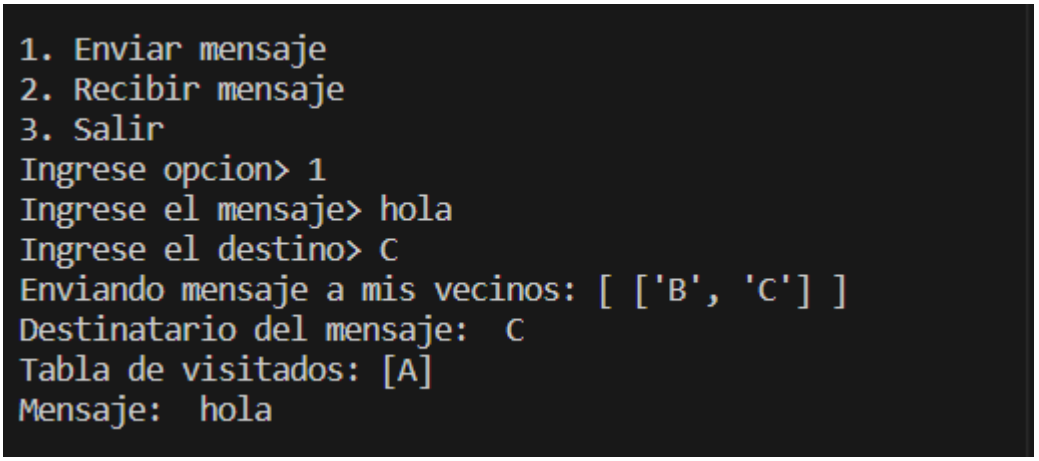
- Flooding
    - Topología



    - Creación de nodos



    - Propagación de mensaje de A hacia C



```
1. Enviar mensaje
2. Recibir mensaje
3. Salir
Ingrese opcion> 2
Ingrese el mensaje de la forma 'destino,mensaje,emisor
'> C,hola,A
Ingrese la tabla de visitados de la forma 'nodo1,nodo2
,nodo3'> A
Enviando mensaje a mis vecinos:  ['A', 'C']
Destinatario del mensaje: C emisor: A
Tabla de visitados: [ A, B]
```

- Evitar que los mensajes se enloopen

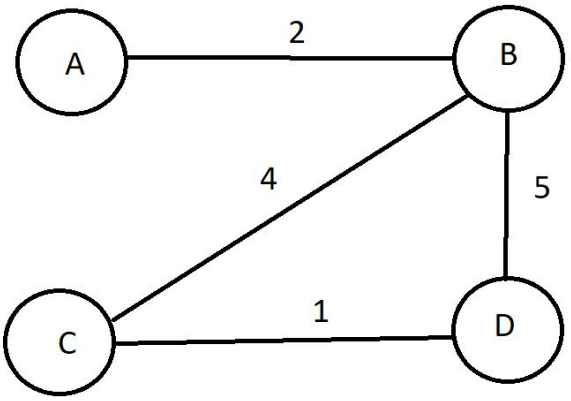
```
1. Enviar mensaje
2. Recibir mensaje
3. Salir
Ingrese opcion> 2
Ingrese el mensaje de la forma 'destino,mensaje,emisor'> C,hola,A
Ingrese la tabla de visitados de la forma 'nodo1,nodo2,nodo3'> A,B
Mensaje ya enviado, no es necesario reenviarlo
```

- Recepción de mensajes

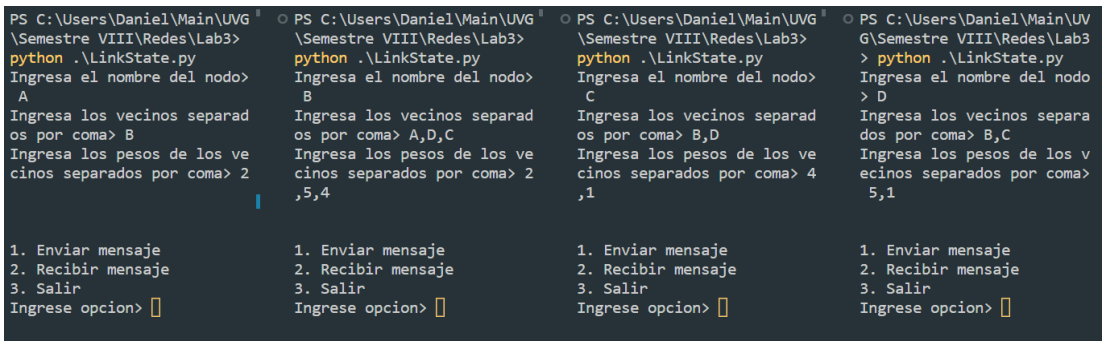
```
1. Enviar mensaje
2. Recibir mensaje
3. Salir
Ingrese opcion> 2
Ingrese el mensaje de la forma 'destino,mensaje,emisor'> C,hola,A
Ingrese la tabla de visitados de la forma 'nodo1,nodo2,nodo3'> A
Mensaje recibido de: A el mensaje es: hola
```

```
1. Enviar mensaje
2. Recibir mensaje
3. Salir
Ingrese opcion> 2
Ingrese el mensaje de la forma 'destino,mensaje,emisor'> C,hola,A
Ingrese la tabla de visitados de la forma 'nodo1,nodo2,nodo3'> A,B
Mensaje recibido de: A el mensaje es: hola
```

- LinkState
    - Topología



    - Creación de nodos



    - Envío de mensaje de A hacia D
      Enviar mensaje desde A:

```
PS C:\Users\Daniel\Main\UVG
\Semestre VIII\Redes\Lab3>
python .\LinkState.py
Ingresa el nombre del nodo>
 A
Ingresa los vecinos separad
os por coma> B
Ingresa los pesos de los ve
cinos separados por coma> 2



1. Enviar mensaje
2. Recibir mensaje
3. Salir
Ingrese opcion> 1
Ingrese el mensaje> Hola te
ngo ganas de un pastel
Ingrese el destino> D
El mensaje es Hola tengo ga
nas de un pastel
El siguiente nodo al que se
 debe mandar es B


1. Enviar mensaje
2. Recibir mensaje
3. Salir
Ingrese opcion>
```

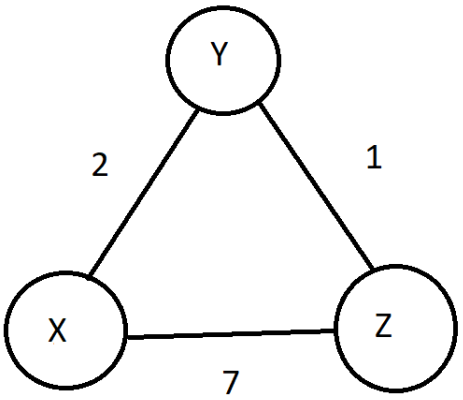
Recepción de mensaje por parte de B:

```
ython .\LinkState.py
Ingresa el nombre del nodo>
 B
Ingresa los vecinos separad
os por coma> A,C,D
Ingresa los pesos de los ve
cinos separados por coma> 2
,4,1


1. Enviar mensaje
2. Recibir mensaje
3. Salir
Ingrese opcion> 2
Ingrese el mensaje de la fo
rma 'emisor,destino,mensaje
'> A,D,Hola tengo ganas de
un pastel
Emisor:  A
Enviar mensaje: Hola tengo
ganas de un pastel
El siguiente nodo al que se
 debe mandar es D


1. Enviar mensaje
2. Recibir mensaje
3. Salir
Ingrese opcion>
```

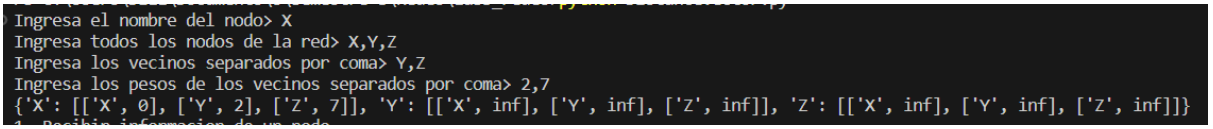
Recepción de mensaje por parte de D:

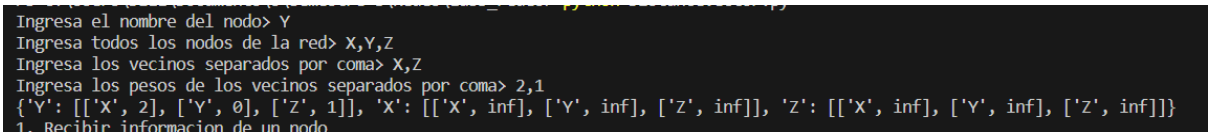
```
G\Semestre VIII\Redes\Lab3
> python .\LinkState.py
Ingresa el nombre del nodo
> D
Ingresa los vecinos separa
dos por coma> B,C
Ingresa los pesos de los v
ecinos separados por coma>
 5,1


1. Enviar mensaje
2. Recibir mensaje
3. Salir
Ingrese opcion> 2
Ingrese el mensaje de la f
orma 'emisor,destino,mensa
je'> A,D,Hola tengo ganas
de un pastel
El destino del mensaje era
 el nodo actual
Hola tengo ganas de un pas
tel


1. Enviar mensaje
2. Recibir mensaje
3. Salir
Ingrese opcion>
```

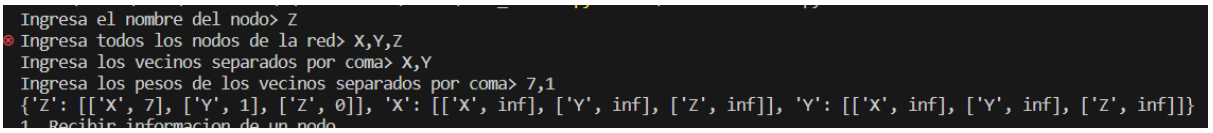
- Distance Vector
    - Topología



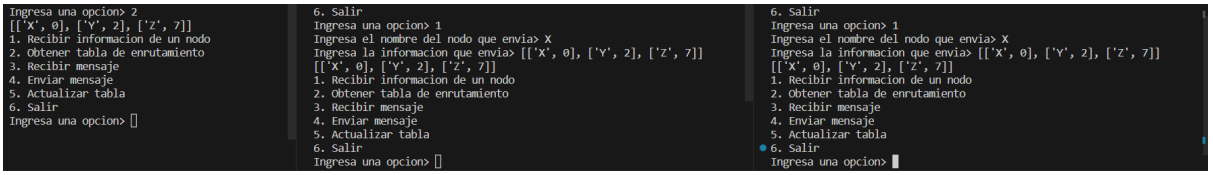
    - Creación de nodos

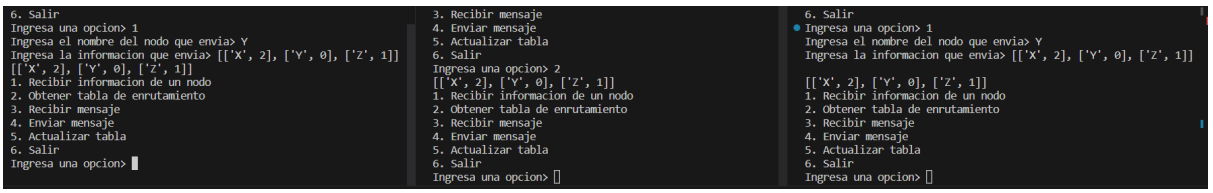


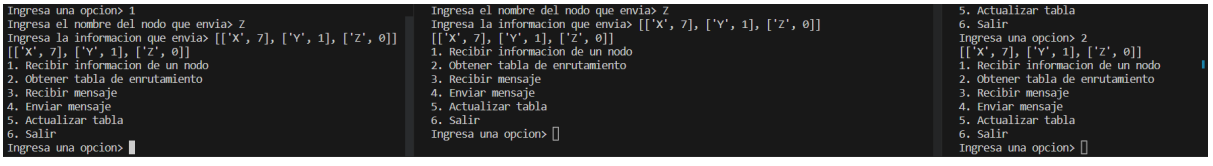




    - Enviar información de la tabla

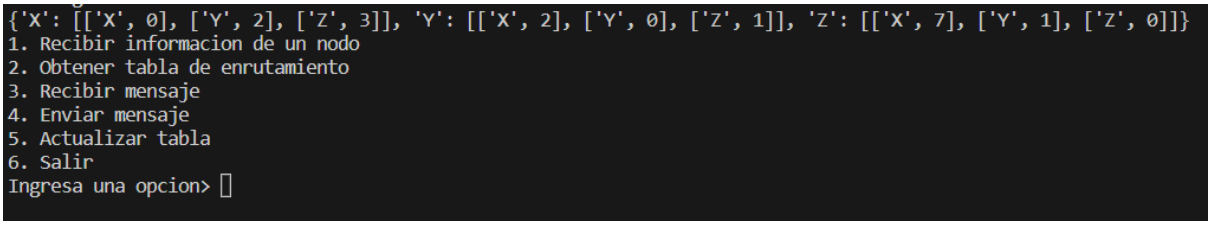




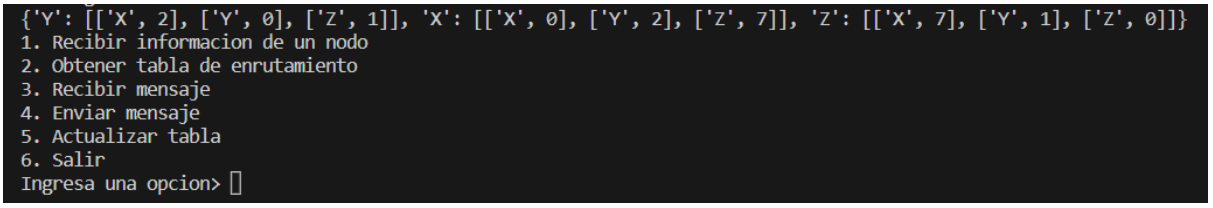


- Tabla resultante tras primer envío de información

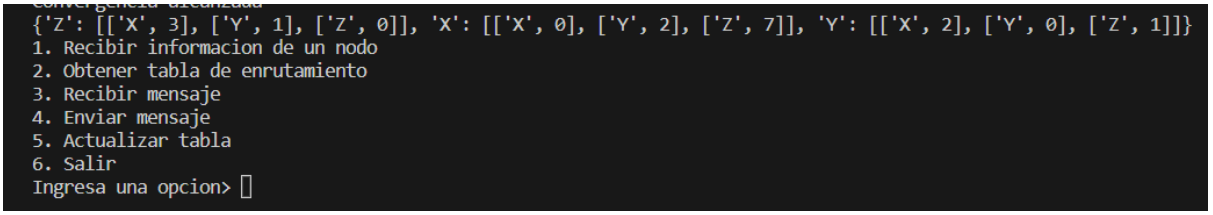
Nodo X



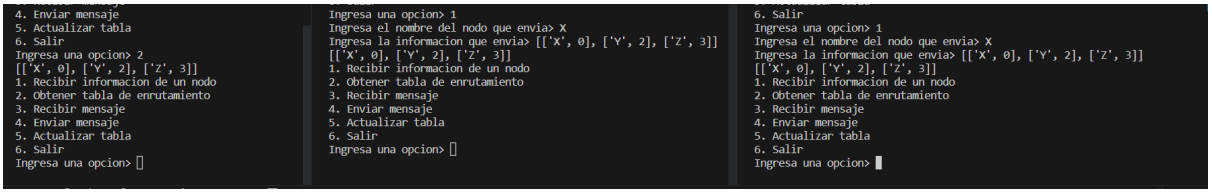
Nodo Y

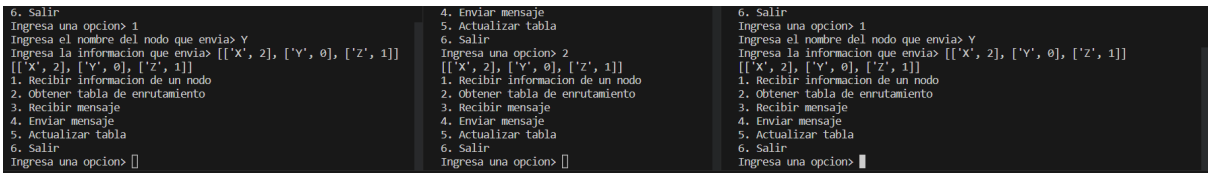


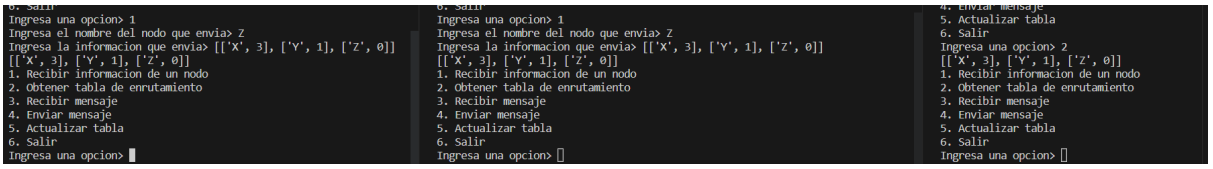
Nodo Z



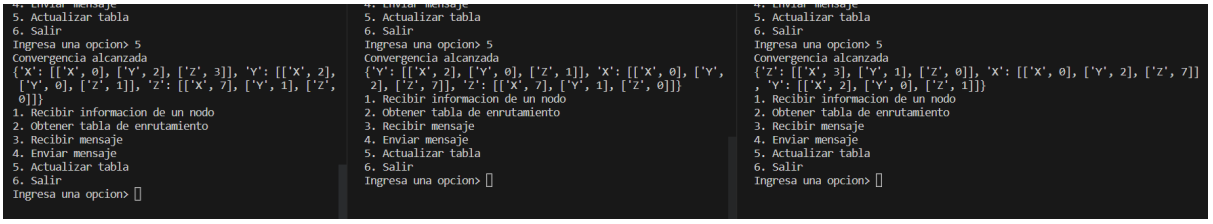
- Segundo envío de información



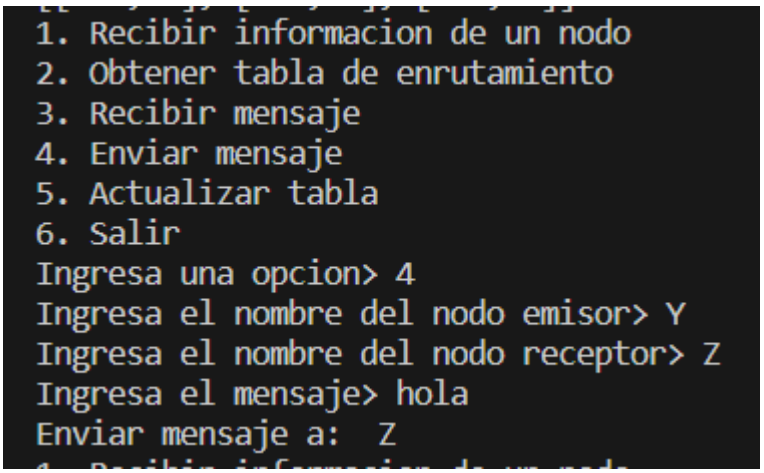


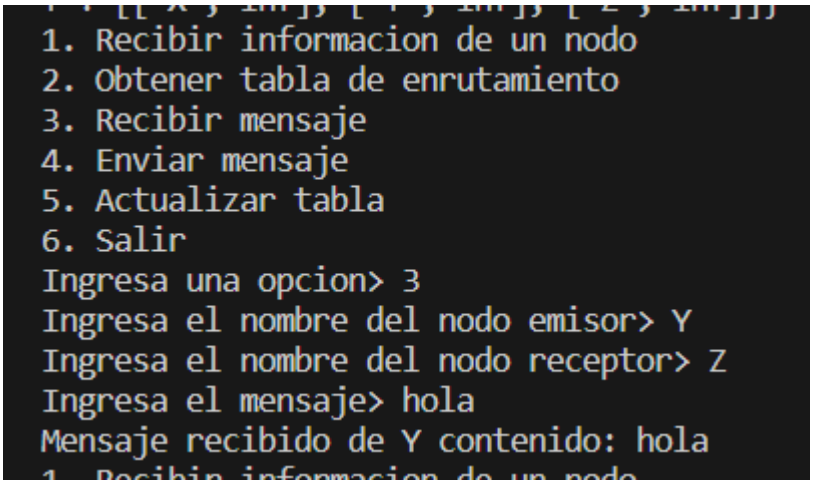


- Convergencia



- Enviar mensaje de Y a Z





Discusión

El algoritmo más sencillo en cuanto a implementación y concepto fue Flooding, esto debido a que solo se debía enviar un mensaje a todos los nodos vecinos y no se debían realizar cálculos muy complejos. Sin embargo, esta sencillez tiene una desventaja y es que cuando se deban usar los nodos como clientes de XMPP, se deberá realizar una implementación orientada al protocolo, cosa que no se realizó en este momento por los requerimientos del laboratorio.

En contraste, el algoritmo más complejo de implementar fue Distance Vector por algunos factores. El primero fue el dinamismo de la red, de forma que es necesario saber las tablas de enrutamiento de los demás nodos en múltiples tiempos, por lo que se deben realizar múltiples pasos de mensajes entre nodos hasta converger, armar la tabla de enrutamiento estable y empezar a mandar mensajes. Además, hacerlo con múltiples terminales para simular los nodos fue difícil, porque nosotros debíamos decidir a quién se le mandaba cada mensaje. Por otra parte, el uso de la

función de Bellman-Ford para calcular las nuevas entradas de la tabla es relativamente complejo computacional y programáticamente.

El algoritmo de Link State, fue de dificultad intermedia de implementar. Es un algoritmo bastante eficiente para realizar el envió de paquetes en una estructura estática. Tiene la ventaja de que no cambian los pesos de las conexiones, por lo que no se debe estar pidiendo información de los otros nodos constantemente, lo cual reduce el tiempo de cálculo de las rutas, permite enviar mensajes más rápidamente y permite mejor rendimiento en cierta forma de la red. Esto agilizó el desarrollo del algoritmo en el laboratorio, porque nos permitió realizar la estructura “hardcodeada” mencionada anteriormente. Sin embargo, en una implementación real, se debe hacer un flooding entre nodos para conocer la estructura de la red y no se sabe de antemano.

Al comparar los algoritmos se puede decir que Flooding es un algoritmo poco eficiente, porque se debe mandar el mensaje a todos los nodos vecinos y estos se los debe mandar a los vecinos y así sucesivamente. Para no generar mensajes que se queden en loop, es necesario agregar información al header del mensaje para saber por cuales nodos se ha pasado anteriormente, de forma que no se vuelva a mandar el mensaje si ya pasó por el nodo. Esto puede crear mensajes más pesados y puede reducir el rendimiento de la red. Link State, como se mencionó, es bastante eficiente en una estructura estática y Distance Vector, aunque es más tardado en calcular sus rutas, permite el dinamismo de la red, lo cual lo hace bastante útil en algo como internet, donde la cantidad de nodos no es fija para nada.

Conclusiones

- Para decidir qué algoritmo de enrutamiento utilizar, se debe tomar en cuenta la necesidad de escalamiento de la red, es decir la cantidad de nodos que tendrá, necesidades de rendimiento de esta y características como dinamismo de la red.
- El uso de terminales para simular los nodos de la red, dificultó el desarrollo de los algoritmos debido a que era muy difícil simular muchos nodos, la identificación de los nodos de la red y pasar mensajes en múltiples nodos.
- En los algoritmos implementados, es necesario agregar cierto tipo de información en el header de los mensajes enviados, para evitar bucles en la red y ayudar a mejorar el rendimiento.

Comentario grupal

Fue una práctica bastante interesante, en cuanto al desarrollo de algoritmos. Debimos realizar una buena evaluación de cómo implementar cada uno, teniendo en cuenta su capacidad de escalar para la siguiente parte del laboratorio. Esto causó dificultad porque a pesar de que la implementación no era tan difícil, se tuvieron que planear a futuro ciertas partes del código. Consideramos que se debería buscar un punto intermedio para próximas ediciones del curso, porque el uso de XMPP en la parte 2 causa confusión en la parte 1. A pesar de esto, fue importante para entender mejor los algoritmos, principalmente, Distance Vector.

Referencias

Terrell Hanna, K. (2021, June 1). *What is network flooding and how does it work?* SearchNetworking. https://www.techtarget.com/searchnetworking/definition/flooding

Antoniou, S. (2016). *Dynamic Routing Protocols: Distance Vector and Link State Protocols*. Pluralsight.com. https://www.pluralsight.com/blog/it-ops/dynamic-routing-protocol

WAN, L. (2023, August 3). *What are the advantages and disadvantages of distance vector and link state algorithms?* Www.linkedin.com. https://www.linkedin.com/advice/1/what-advantages-disadvantages-distance-vector-link-state