Universidad del Valle de Guatemala
Redes
Sección 21

Laboratorio 3

Algoritmos de Enrutamiento

https://github.com/JaniMariQuesiRami/lab3y4-redes

21016 Javier Chavez
21085 Andres Quezada
21631 Mario Cristales

Descripción de la práctica

En esta práctica, se implementan y prueban algoritmos de enrutamiento en una red simulada utilizando el protocolo XMPP. Cada nodo de la red se representa como un cliente que puede enviar y recibir mensajes para formar su tabla de enrutamiento. El objetivo es comprender cómo funcionan estos algoritmos y su aplicación en la actualización dinámica de rutas, asegurando que los mensajes lleguen a su destino a través de la mejor ruta disponible.

Algoritmos utilizados y su implementación

Se implementaron los algoritmos de enrutamiento Flooding y Link State Routing. Cada uno de estos algoritmos se programó de forma modular, permitiendo su ejecución independiente y pruebas en diferentes entornos. Los nodos en la red utilizan estos algoritmos para intercambiar información sobre las rutas y vecinos, actualizando dinámicamente las tablas de enrutamiento y adaptándose a cambios en la topología de la red.

Flooding

El algoritmo Flooding envía mensajes a todos los nodos vecinos sin discriminación, replicando los mensajes hasta que todos los nodos de la red los reciban. Esta técnica garantiza que la información llegue a todos los rincones de la red, aunque no optimiza el uso de los recursos, ya que puede generar un tráfico excesivo al enviar mensajes duplicados por múltiples rutas.

Nuestra implementación se basa en los siguientes pasos:

- Inicialización:

  Cada nodo es una instancia de nuestra clase de nodos para flooding, que extiende de nuestra clase base Nodo. Cada una de estas instancias de nodo tiene un nombre y una lista de vecinos, que se establecen al parsear la topología de la red desde el archivo de configuración, donde se ingresa para cada nodo sus vecinos y sus pesos.

- Proceso de Flooding:

  El método flood() se utiliza para enviar un mensaje desde un nodo a todos sus vecinos. Este es el mensaje que contiene información sobre la ruta que ha seguido hasta el momento y el peso acumulado de esa ruta. Cuando un nodo recibe un mensaje, primero verifica si ya ha sido parte de esa ruta. Si no lo ha sido, lo agrega y propaga el mensaje a sus vecinos, menos al nodo desde el que recibió el mensaje (evitando la redundancia). El nodo receptor almacena todas las rutas posibles desde el emisor original en su mapa de rutas, lo que permite más adelante determinar la mejor ruta en función del peso total acumulado.

- Selección de la Mejor Ruta:

  Para determinar la mejor ruta hacia un nodo específico, evaluamos todas las rutas recibidas y seleccionamos la que tiene el menor peso acumulado, ya que esto significa que es la que menos ha recorrido. Esta implementación garantiza que, incluso si un nodo recibe múltiples copias del mismo mensaje por diferentes rutas, siempre seleccionará la ruta más eficiente en términos de peso.

Link State Routing

El algoritmo Link State Routing es un algoritmo que recopila información de todos los nodos para conocer toda la red. Cada nodo calcula la ruta más corta hacia los demás nodos usando Dijkstra.

Nuestra implementación se basa en los siguientes pasos:

- Inicialización:

  Cada nodo es una instancia de nuestra clase de nodos para Link State Routing, que extiende de nuestra clase base Nodo. Cada una de estas instancias de nodo tiene un nombre y una lista de vecinos, que se establecen al leer la topología de la red desde un archivo de configuración, donde se especifican los vecinos y los pesos de las conexiones para cada nodo.

- Proceso de Link State Routing

  El método linkState() es el responsable de calcular las rutas más cortas desde el nodo actual a todos los demás nodos en la red. Este método utiliza el algoritmo de Dijkstra para buscar la ruta más corta. Primero, se inicializa la distancia a todos los nodos como infinita y solo el nodo actual tiene una distancia de 0. Luego, selecciona el nodo no visitado con la distancia más corta, calcula las distancias a sus vecinos, y actualiza las distancias si se encuentra una ruta más corta.

- Selección de la mejor ruta

  Para determinar la mejor ruta hacia un nodo específico, el método caminoMasCorto() reconstruye la ruta óptima utilizando la información calculada previamente. Este método sigue la "pista" de nodos desde el destino hasta el origen utilizando la información almacenada en el paso anterior. El resultado final es el camino más corto y su distancia total desde el nodo de origen hasta el nodo de destino.

Resultados

FLOODING

```
PS C:\Users\PC\Documents\Uni\Redes\lab3> yarn start A 123456
yarn run v1.22.19
warning package.json: No license field

$ node src/index.js A 123456
Seleccione el método de enrutamiento:
1. Inundación
2. Estado de Enlace
1
INFO: El nodo A utilizará la dirección: sara21016
INFO: Conectado exitosamente como sara21016@alumchat.lol/exampleINFO: Conectado exitosamente como sara21016
Enviar (1) o Recibir (2) o Salir (3):
2
INFO: Escuchando mensajes entrantes...
INFO: Mensaje recibido de cha21016-test5@alumchat.lol/example: {"type":"message","from":"javier21016","to":"sara21016@alumchat.lol","hops":2,"payload":"Hola amiguito A"}
INFO: Este mensaje es para este nodo
INFO: Sesión cerrada para sara21016
Done in 29.43s.
PS C:\Users\PC\Documents\Uni\Redes\lab3>
```

```
PS C:\Users\PC\Documents\Uni\Redesyarn start B 123456
yarn run v1.22.19
warning package.json: No license field

$ node src/index.js B 123456
Seleccione el método de enrutamiento:
1. Inundación
2. Estado de Enlace
1
INFO: El nodo B utilizará la dirección: javier21016
INFO: Conectado exitosamente como javier21016@alumchat.lol/example
INFO: Conectado exitosamente como javier21016
Enviar (1) o Recibir (2) o Salir (3):
1
Ingrese el nodo receptor: A
Mensaje: Hola amiguito A
INFO: Mejor ruta desde B hacia A: B -> D -> A
INFO: Peso total: 3
INFO: Mensaje enviado a cha21016-test5@alumchat.lol: {"type":"message","from":"javier21016","to":"sara21016@alumchat.lol","hops":1,"payload":"Hola amiguito A"}
Enviar (1) o Recibir (2) o Salir (3):
```

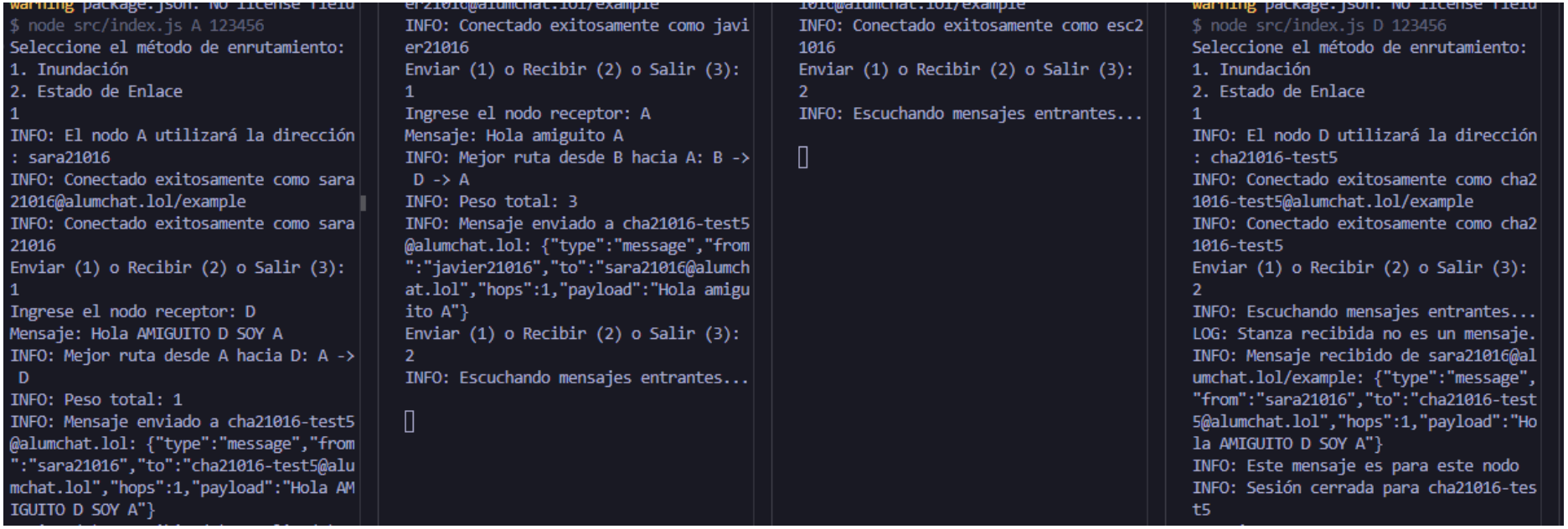
```
PS C:\Users\PC\Documents\Uni\Redes\lab3>      yarn start C 123456
yarn run v1.22.19
warning package.json: No license field

$ node src/index.js C 123456
Seleccione el método de enrutamiento:
1. Inundación
2. Estado de Enlace
1
INFO: El nodo C utilizará la dirección: esc21016
INFO: Conectado exitosamente como esc21016@alumchat.lol/example
INFO: Conectado exitosamente como esc21016
Enviar (1) o Recibir (2) o Salir (3):
2
INFO: Escuchando mensajes entrantes...
```

```
es\lab3>      yarn start D 123456
yarn run v1.22.19
warning package.json: No license field

$ node src/index.js D 123456
Seleccione el método de enrutamiento:
1. Inundación
2. Estado de Enlace
1
INFO: El nodo D utilizará la dirección: cha21016-test5
INFO: Conectado exitosamente como cha21016-test5@alumchat.lol/example
INFO: Conectado exitosamente como cha21016-test5
Enviar (1) o Recibir (2) o Salir (3):
2
INFO: Escuchando mensajes entrantes...
INFO: Mensaje recibido de javier21016@alumchat.lol/example: {"type":"message","from":"javier21016","to":"sara21016@alumchat.lol","hops":1,"payload":"Hola amiguito A"}
INFO: El mensaje no es para este nodo.
INFO: Redirigiendo mensaje al siguiente salto: sara21016@alumchat.lol
INFO: Mensaje enviado a sara21016@alumchat.lol: {"type":"message","from":"javier21016","to":"sara21016@alumchat.lol","hops":2,"payload":"Hola amiguito A"}
INFO: Sesión cerrada para cha21016-test5
Done in 28.29s.
PS C:\Users\PC\Documents\Uni\Redes\lab3>
```

```
es\lab3>      yarn start E 123456
yarn run v1.22.19
warning package.json: No license field

$ node src/index.js E 123456
Seleccione el método de enrutamiento:
1. Inundación
2. Estado de Enlace
1
INFO: El nodo E utilizará la dirección: cha21016-test6
INFO: Conectado exitosamente como cha21016-test6@alumchat.lol/example
INFO: Conectado exitosamente como cha21016-test6
Enviar (1) o Recibir (2) o Salir (3):
2
INFO: Escuchando mensajes entrantes...
LOG: Stanza recibida no es un mensaje.
```

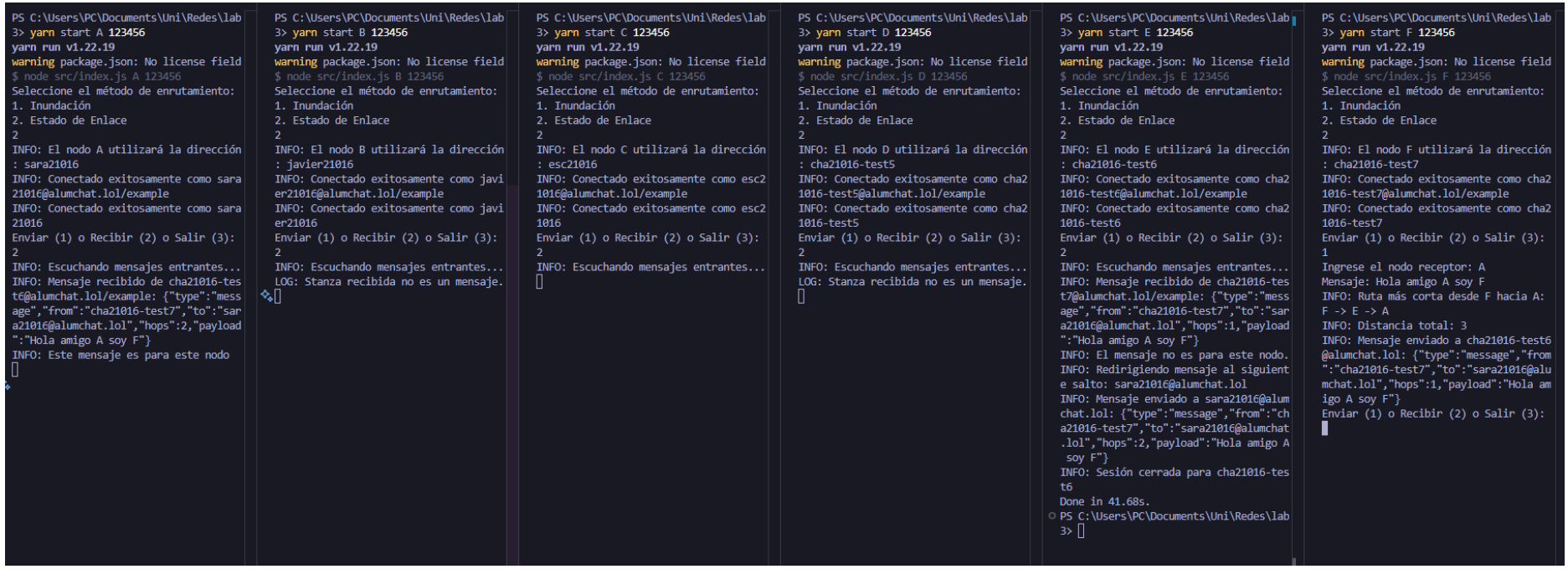
```
es\lab3>      yarn start F 123456
yarn run v1.22.19
warning package.json: No license field

$ node src/index.js F 123456
Seleccione el método de enrutamiento:
1. Inundación
2. Estado de Enlace
1
INFO: El nodo F utilizará la dirección: cha21016-test7
INFO: Conectado exitosamente como cha21016-test7@alumchat.lol/example
INFO: Conectado exitosamente como cha21016-test7
Enviar (1) o Recibir (2) o Salir (3):
2
INFO: Escuchando mensajes entrantes...
```

Prueba 1: Mensaje de B hacia A. 2 hops con un peso de 3.

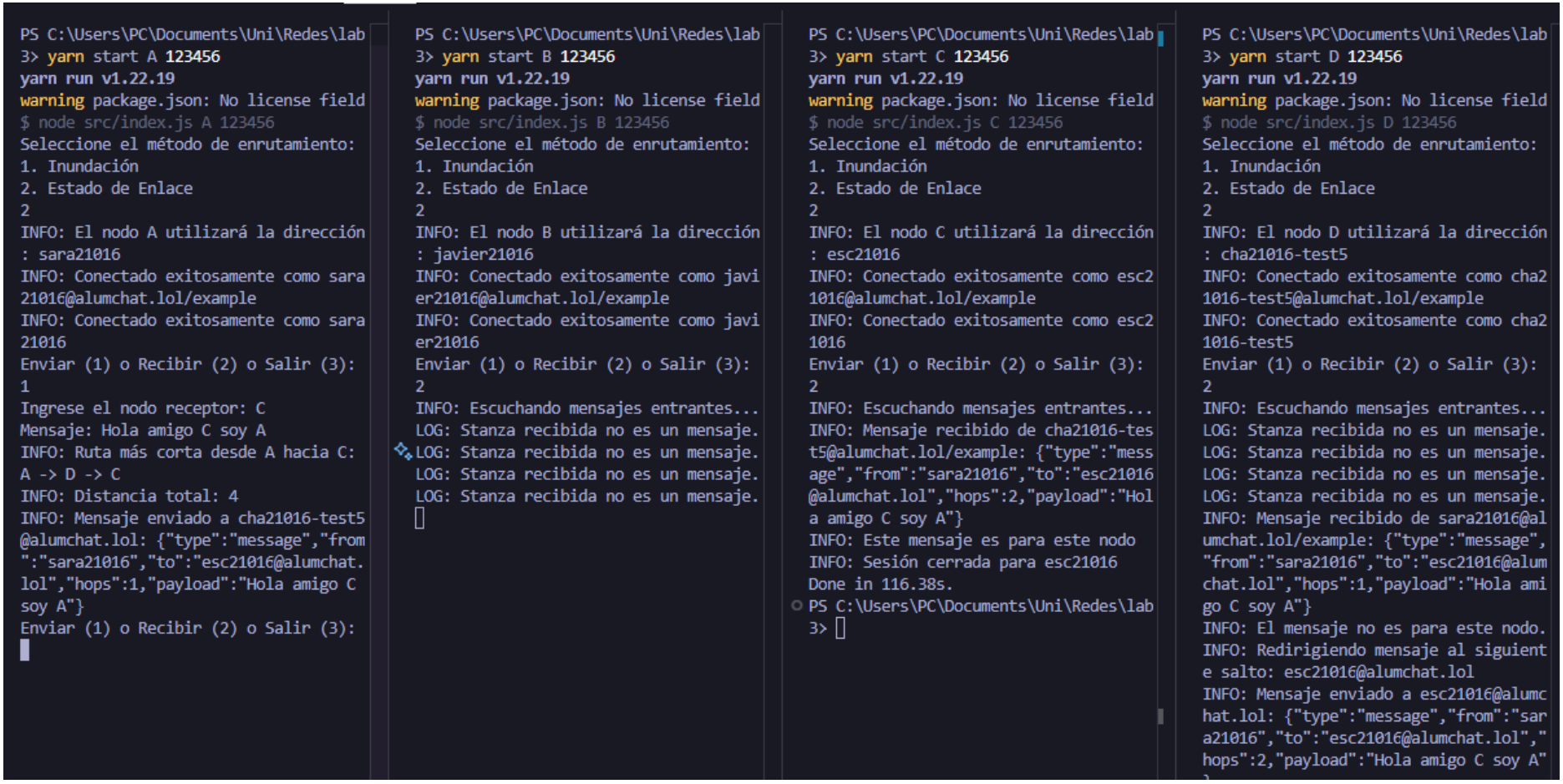


Prueba 2: Mensaje de A hacia D. 1 hop con un peso de 1.

LINK STATE



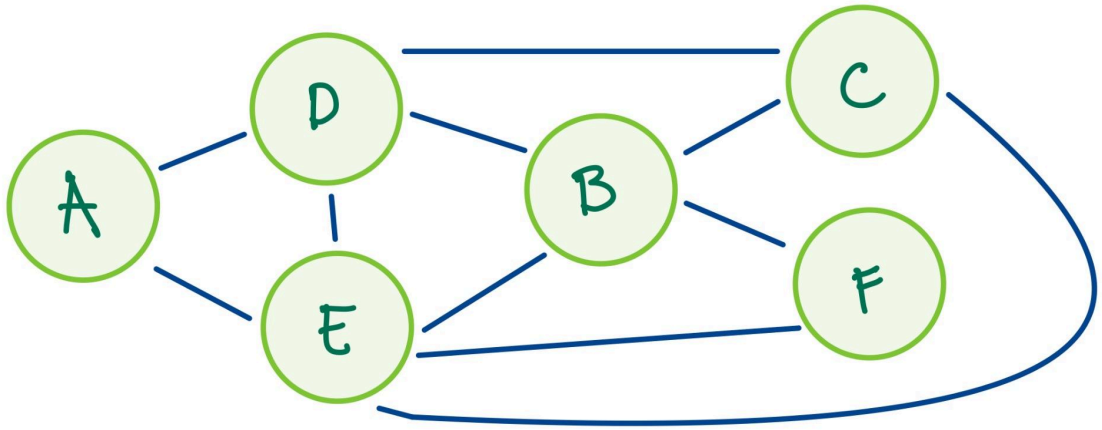
Prueba 1: Mensaje de F hacia A. 2 hops con peso de 3.



Prueba 2: Mensaje de A hacia C. 2 hops con peso de 4.

Discusión

Topología



Los nodos están interconectados con múltiples caminos posibles. Esta estructura permite comparar el rendimiento de dos algoritmos de enrutamiento: Flooding y Link State Routing.

El algoritmo Flooding en esta topología envía mensajes a todos los nodos vecinos sin considerar la eficiencia de la ruta. En nuestro caso, cuando un nodo como A envía un mensaje, este se transmite a sus vecinos D y E, quienes a su vez propagan el mensaje a sus propios vecinos, y así sucesivamente. Este proceso se repite hasta que todos los nodos reciben el mensaje.

Por otro lado, Link State Routing proporciona una solución más optimizada para encontrar rutas eficientes. En esta topología, cada nodo, como A, recopila información sobre todos los enlaces en la red y ejecuta el algoritmo de Dijkstra para calcular la ruta más corta a todos los demás nodos. Por ejemplo, para llegar a F, A calculará la ruta A -> E -> F si esta es la de menor costo total en términos de peso.

A diferencia de flooding, este método evita la transmisión de mensajes innecesarios y garantiza que se utilicen las rutas más cortas posibles. Sin embargo, Link State Routing requiere que cada nodo mantenga y actualice información global sobre la red, lo cual puede ser costoso en términos de memoria y procesamiento, especialmente en redes más grandes.

En general, la elección entre Flooding y Link State Routing depende de los requisitos específicos de la red. Flooding es simple y robusto en redes pequeñas o en situaciones donde la topología cambia rápidamente y la latencia no es crítica. Link State Routing, sin embargo, es más eficiente en términos de uso de recursos y tiempo de entrega de mensajes en redes más grandes y estables, como la que se modela en la topología dada. Dependiendo de las características de la red y los resultados de nuestras pruebas, [coloca una conclusión sobre cuál algoritmo es más adecuado para esta topología].

Conclusiones

- Los métodos de enrutamiento suelen requerir equilibrar entre la eficacia y la precisión. Métodos detallados como el algoritmo de Dijkstra proporcionan las rutas más eficientes, pero utilizan más recursos y capacidad de transmisión. Alternativamente, métodos que conservan recursos, como el algoritmo de Distance Vector, pueden no ser tan precisos en redes intrincadas, aunque resultan ser eficaces para topologías en constante cambio.

- A través del trabajo realizado en esta práctica y las simulaciones ejecutadas, logramos comprender de forma clara cómo los algoritmos de enrutamiento permiten a un emisor identificar a sus nodos adyacentes y comunicar datos a través de la ruta más eficiente, siempre logrando el objetivo de enviar un mensaje a un destinatario pero sin necesariamente saber información sobre el.

- La elección entre un algoritmo u otro depende en gran medida de las características específicas de la red en la que se implementarán, ya que cada uno tiene requisitos y rendimientos diferentes según el entorno.

Comentario grupal

La práctica nos pareció interesante sobre todo la implementación del link state, porque pudimos entender un protocolo de mayor complejidad. Sin embargo creemos que hubiera sido más sencillo sin usar xmpp, ya que esto solo le agregaba una capa de dificultad innecesaria ya que después pasamos más tiempo resolviendo errores de xmpp que de la implementación como tal objetivo de este laboratorio.

Referencias

TutorialsPoint. (2024) Flooding in Computer Network. Recuperado de: https://www.tutorialspoint.com/flooding-in-computer-network

Thomas, K. (2024) Link State Routing: Understanding Basics, Protocol & Algorithm. Recuperado de: https://unstop.com/blog/link-state-routing