Universidad del Valle de Guatemala

Redes

Sección 20

Algoritmos de Enrutamiento

Laboratorio 3

Diego Andrés Alonzo Medinila - 20172

Arturo Heberto Argueta Ávila - 21527

Renato Guzmán

Guatemala, 2024

Descripción de la práctica

La práctica tuvo como objetivo la implementación de los algoritmos de enrutamiento, Link State y Flooding. De manera que se desarrollaron clases que permitirían modelar los algoritmos presentados, a su vez se desarrollaron dichos algoritmos diseñados con una implementación de XMPP dado que se implementarán dichos algoritmos para simular nodos dentro de una red utilizando el server *alumchat.lol*. Para este laboratorio se crearon diferentes instancias de estos algoritmos de manera que de esta forma se simularan los nodos como tal. Así mismo, se desarrolló un CLI de manera que se pudiera observar cómo es que se iría desarrollando la ejecución de los algoritmos, simulando y demostrando sobre todo el paso de mensajes de un nodo a otro.

Descripción de los algoritmos

Flooding

El algoritmo flooding consiste en enviar un mensaje desde un origen a todos los nodos de la red, de manera que se reenvía el mensaje a los nodos vecinos y estos los envían a sus nodos vecinos más no al nodo padre que les envió el mensaje. Esto permite que se propague en toda la red el mensaje, sin embargo, presenta un problema al generar envíos duplicados o bucles en la red ya que, puede que dentro del grafo que componga la red se genere un bucle infinito. Es por ello que se debe de gestionar para llevar cierto control de si ya se envió el mensaje a los nodos vecinos o no fue así.

Link State

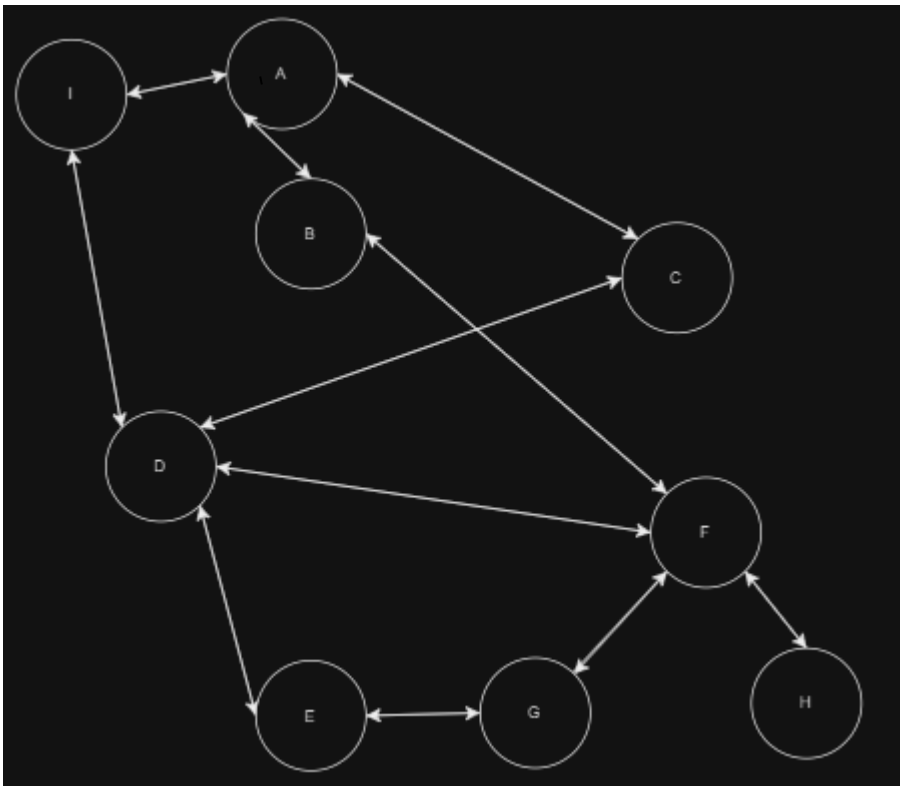
El algoritmo Link State utiliza de base el algoritmo de Dijkstra que calcula la mejor ruta en base al menor costo acumulado a lo largo de la ruta. Así mismo, se encarga de crear una simulación o una versión de la topología completa de la red, y para enviar los mensajes se estructura la ruta óptima. De manera que para establecer las rutas se debe de descubrir los vecinos y los enlaces de cada nodo, luego, compartir los paquetes de estado de enlace en la red, y estos paquetes son los que brindan la capacidad de calcular las rutas más cortas usando Dijkstra, formando las tablas de enrutamiento, a su vez, se adaptará para actualizar las tablas de enrutamiento a partir de cómo varíe la topología de la red.

Resultados

Flooding

Topología

La topología es la siguiente

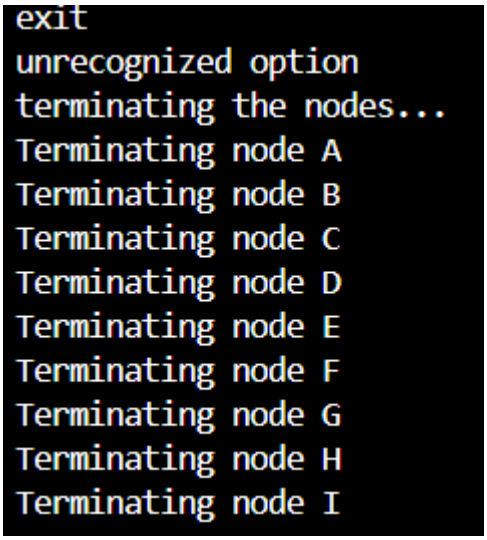


Nodos

```
{"A":["B","I","C"],"B":["A","F"],"C":["A","D"],"D":["I","C","E","F"],"E":["D","G"],"F":["B","D","G","H"],"G":["F","E"],"H":["F"],"I":["A","D"]}
{"A":"arg21527test01@alumchat.lol","B":"arg21527test02@alumchat.lol","C":"arg21527test03@alumchat.lol","D":"arg21527test04@alumchat.lol","E":"arg2
1527test05@alumchat.lol","F":"arg21527test06@alumchat.lol","G":"arg21527test07@alumchat.lol","H":"arg21527test08@alumchat.lol","I":"arg21527test09
@alumchat.lol"}
1. uso de flooding network
2. Uso de Linked State Routing Network
1
Node conected into network : arg21527test01@alumchat.lol
Node conected into network : arg21527test02@alumchat.lol
Node conected into network : arg21527test03@alumchat.lol
Node conected into network : arg21527test04@alumchat.lol
Node conected into network : arg21527test05@alumchat.lol
Node conected into network : arg21527test06@alumchat.lol
Node conected into network : arg21527test07@alumchat.lol
Node conected into network : arg21527test08@alumchat.lol
Node conected into network : arg21527test09@alumchat.lol
```

Enviar mensajes

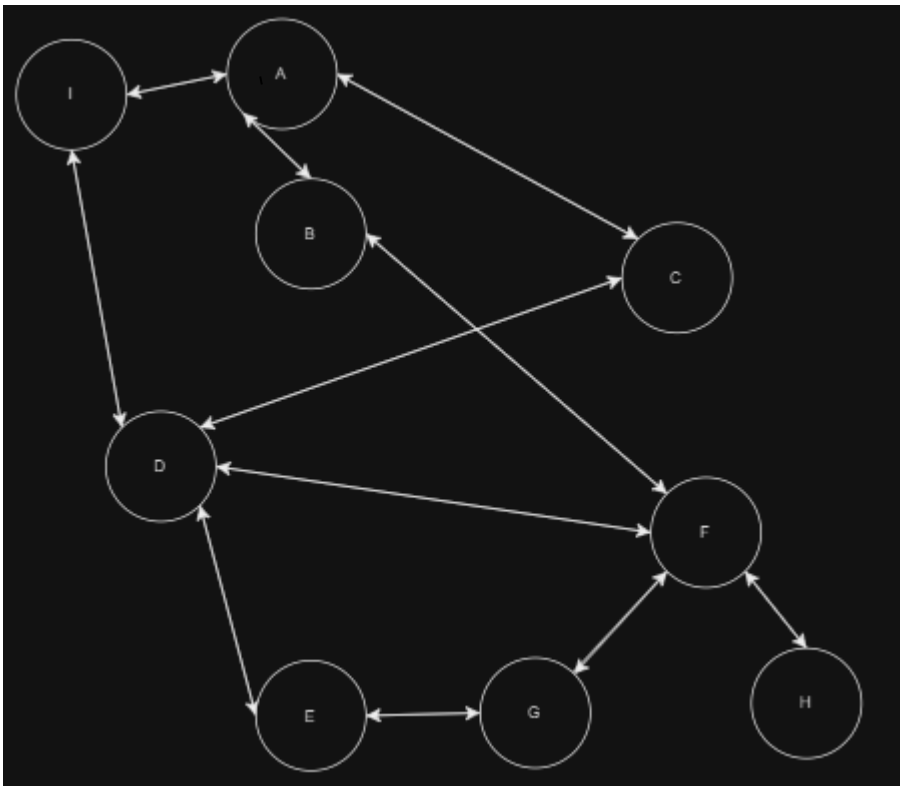
Salir y cerrar nodos



Flooding

Topología

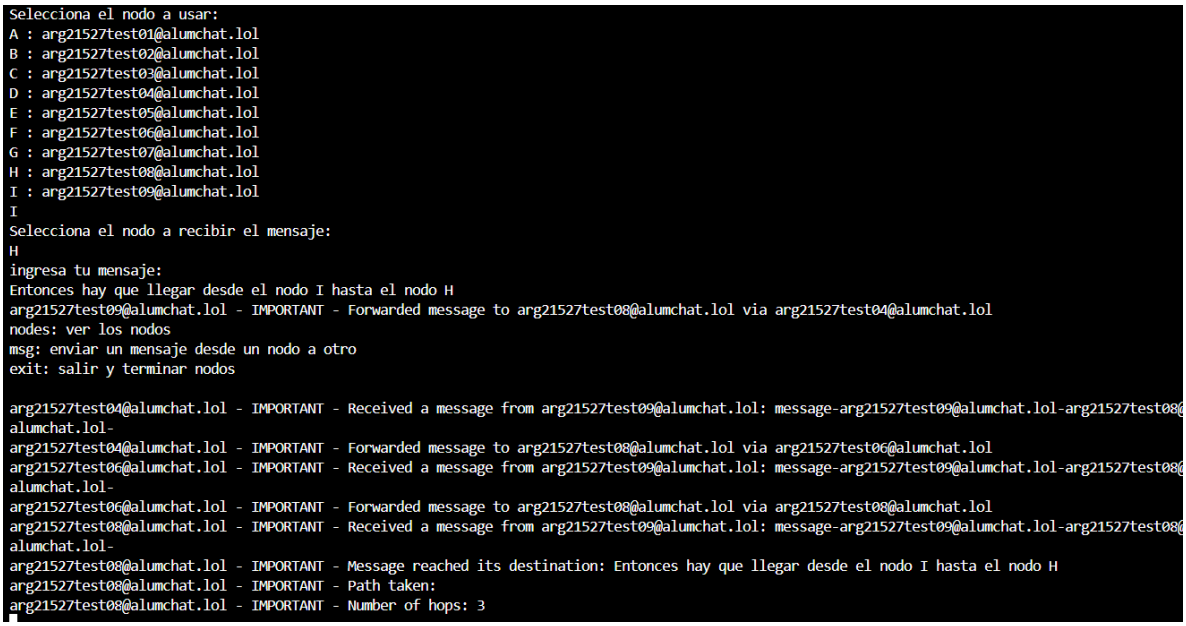
La topología es la siguiente


I
A
B
C
D
F
E
G
H

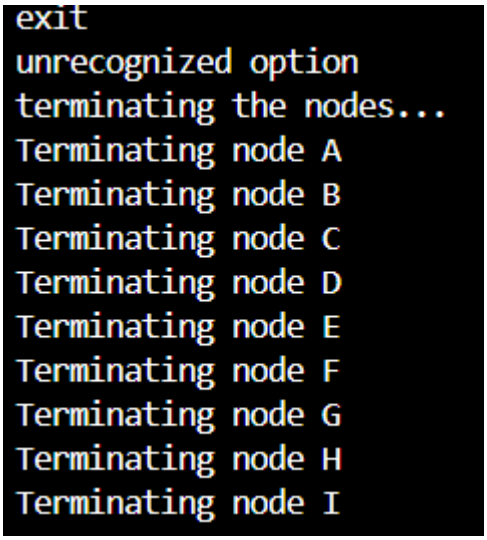
Nodos

```
2
Node conected into network : arg21527test01@alumchat.lol
Node initialized: arg21527test01@alumchat.lol
Node conected into network : arg21527test02@alumchat.lol
Node initialized: arg21527test02@alumchat.lol
Node conected into network : arg21527test03@alumchat.lol
Node initialized: arg21527test03@alumchat.lol
Node conected into network : arg21527test04@alumchat.lol
Node initialized: arg21527test04@alumchat.lol
Node conected into network : arg21527test05@alumchat.lol
Node initialized: arg21527test05@alumchat.lol
Node conected into network : arg21527test06@alumchat.lol
Node initialized: arg21527test06@alumchat.lol
Node conected into network : arg21527test07@alumchat.lol
Node initialized: arg21527test07@alumchat.lol
Node conected into network : arg21527test08@alumchat.lol
Node initialized: arg21527test08@alumchat.lol
Node conected into network : arg21527test09@alumchat.lol
Node initialized: arg21527test09@alumchat.lol
All nodes are online. Starting LSR propagation...
arg21527test08@alumchat.lol has finished sharing state
arg21527test03@alumchat.lol has finished sharing state
arg21527test09@alumchat.lol has finished sharing state
arg21527test02@alumchat.lol has finished sharing state
arg21527test05@alumchat.lol has finished sharing state
arg21527test07@alumchat.lol has finished sharing state
arg21527test01@alumchat.lol has finished sharing state
arg21527test04@alumchat.lol has finished sharing state
arg21527test06@alumchat.lol has finished sharing state
 System is ready
```

Enviar mensajes



Salir y cerrar nodos



Discusión

El algoritmo menos complejo de implementar fue Flooding debido a que se enviaba el mensaje a los nodos vecinos y esto no era complicado de realizar ya que solo era recibir y enviar el mensaje. Por consiguiente el más difícil de implementar fue el Link State debido a que este requería de una implementación de Dijkstra y manejar la ruta óptima así como el manejo de toda la topología para poder calcular la ruta óptima.

Sin embargo, el hecho de que fuera más complejo que Flooding presentó una ventaja y es el hecho de que era mucho más eficiente que el algoritmo de Flooding, de manera que siempre calculaba la ruta óptima. Esto en la vida real sería muy útil ya que realmente el algoritmo de

Flooding es poco práctico al ser muy costoso al no buscar la reducción de costes e incluso puede provocar una saturación de mensajes ya que podrían generarse bucles infinitos si no existe un chequeo de envío de mensajes adecuado. Sin embargo, tiene su utilidad al momento de buscar un mapeo topológico por lo que realmente sería útil al momento de explorar la red, aunque se necesitaría imperativamente utilizar un manejo perfecto de recursos para prevenir bucles infinitos.

Conclusión

El mejor algoritmo debido a su cálculo de pesos ideales basados en Dijkstra es Link State, este algoritmo presenta una ventaja al prevenir bucles infinitos y una utilización de recursos óptima. A su vez, el algoritmo de Flooding tiene la utilidad de ser de exploración ya que permite averiguar la topología de la red por lo que tiene sentido que una mezcla de ambos dentro de una red sea ideal; de manera que se utilice flooding para establecer la topología de la red y Link State para el envío de mensajes ya que se tiene la tabla de enrutamiento.