Universidad del Valle de Guatemala
Facultad de Ingeniería

Laboratorio 3 primera parte
Redes

Marco Antonio Jurado Velasquez 20308
Cristian Eduardo Aguirre Duarte 20231
Diego Alejandro Cordova Barrera 20212

Descripción

En esta práctica, los participantes se enfrentaron al desafío de implementar tres algoritmos de enrutamiento en el contexto de una red de nodos interconectados de arquitecturas de pocos nodos. Los algoritmos implementados fueron Flooding, Dijkstra y Distance Vector. Cada algoritmo aborda el problema del enrutamiento desde una perspectiva única, y se introdujeron modificaciones en el formato JSON enviado entre los nodos utilizado para representar la comunicación entre los nodos. Y cada uno de los algoritmos realizó los cambios necesarios para su funcionamiento.

El algoritmo de enrutamiento por Flooding implica que cada nodo en la red reenvía un mensaje entrante a todos sus vecinos, excepto al nodo desde el cual recibió el mensaje. La finalidad es que el mensaje se propague a través de la red hasta alcanzar su destino. Este algoritmo tiene aplicaciones en redes de emergencia donde el mensaje enviado debe llegar a todos los endpoints de la red conectados. Para esta práctica, se introdujo una modificación en el formato JSON, agregando una lista de personas que ya habían recibido el mensaje. Esto permitió optimizar el proceso de reenvío, evitando enviar el mensaje a nodos que ya lo habían recibido.

El algoritmo de Dijkstra es utilizado para encontrar la ruta más corta entre dos nodos en una red con pesos en las conexiones. El proceso implica calcular la distancia mínima desde un nodo de inicio hacia todos los demás nodos de la red. Esta distancia es determinada a través de sumar los pesos de las conexiones en el camino. El algoritmo de Dijkstra garantiza encontrar la ruta más corta y para esto fue necesario que cada nodo contará con una tabla de información con la distancia a sus nodos vecinos.

El algoritmo de enrutamiento por Vector de Distancia permite a los nodos comunicar sus tablas de enrutamiento entre sí. Cada nodo tiene una tabla que contiene información sobre la distancia estimada hacia otros nodos en la red. Con base en esta información, los nodos ajustan sus rutas y decisiones de enrutamiento. Las tablas se actualizan y propagan en la red a medida que los nodos intercambian información. Esto asegura que los nodos tomen decisiones de enrutamiento basadas en la información más actualizada.

La modificación del formato JSON para incluir información adicional, como la lista de personas que ya recibieron el mensaje en Flooding, permitió explorar formas realistas de optimizar la comunicación.

Resultados

Flooding

Figura no.1 Creación de mensaje para enrutamiento Flooding

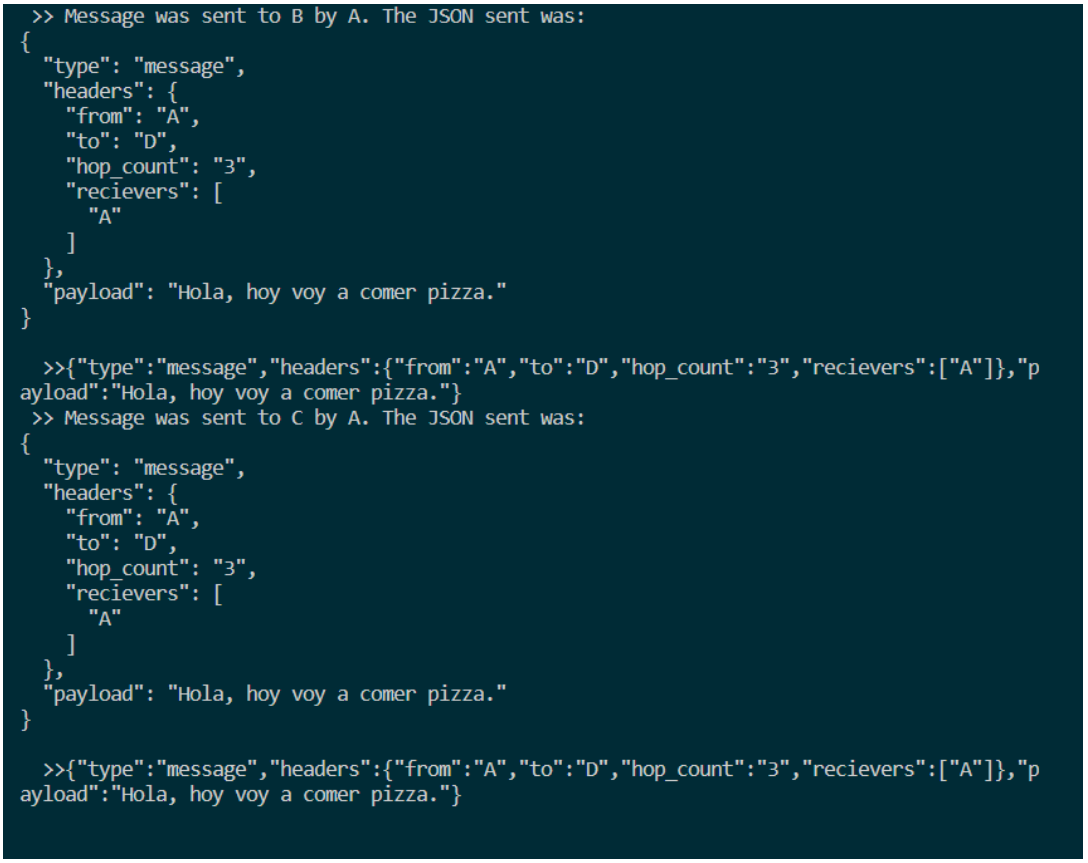


Figura no.2 Recibir mensaje por algoritmo de enrutamiento Flooding cuando el nodo no es el destinatario

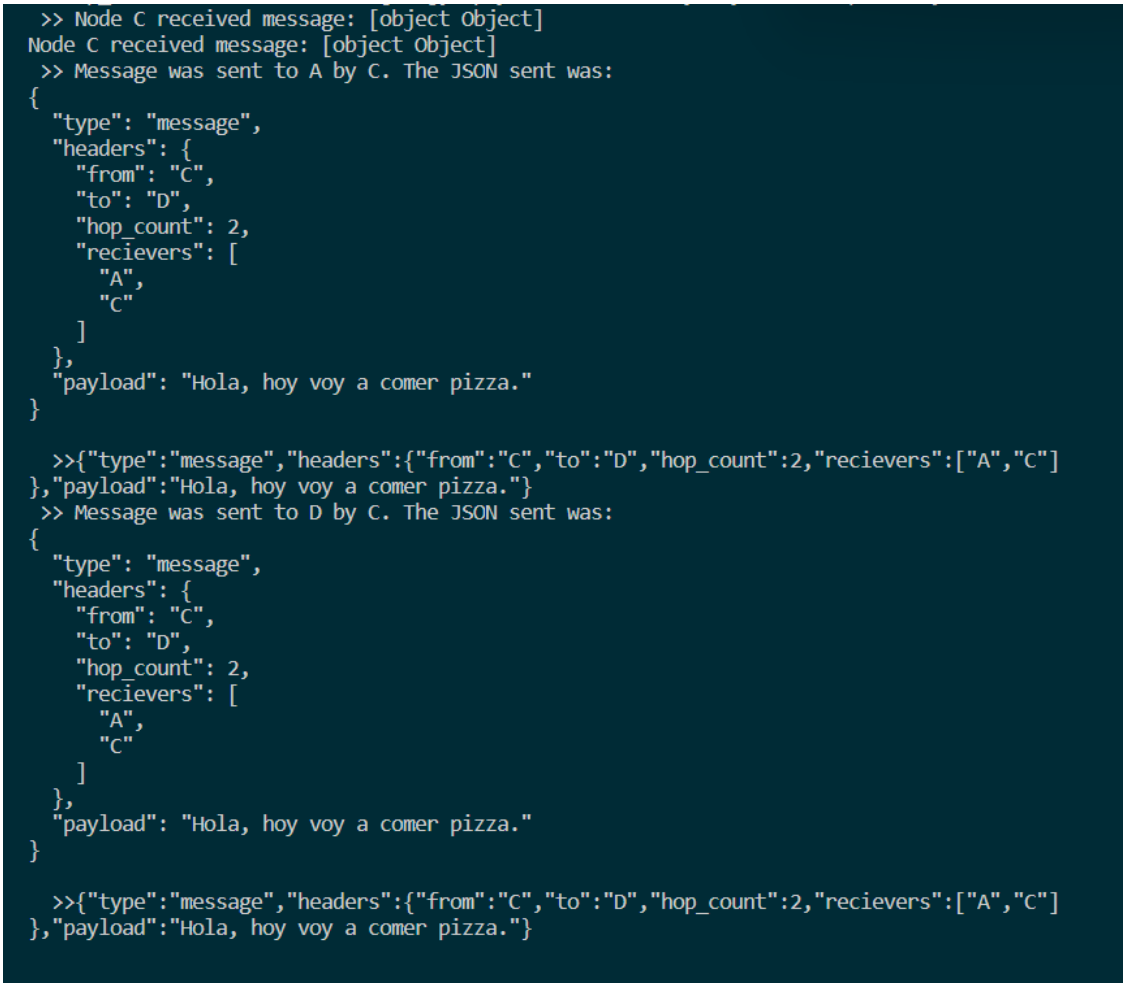


Figura no.3 Recibir mensaje por algoritmo de enrutamiento Flooding cuando el nodo es el destinatario

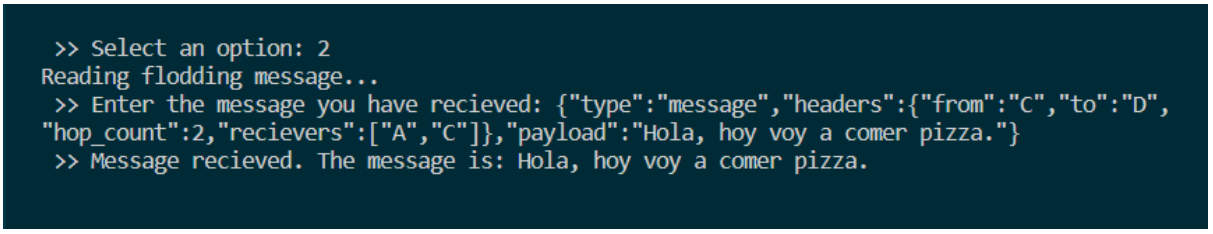
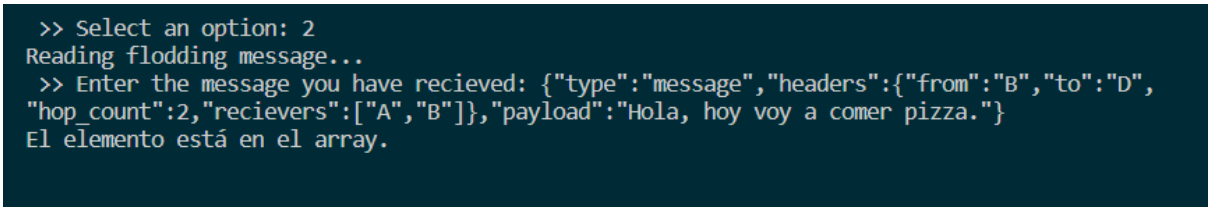


Figura no.4 Recibir mensaje en nodo A por algoritmo de enrutamiento Flooding cuando el nodo A es el que lo envió



Link State

Figura no. 5 Topología usada para simulación

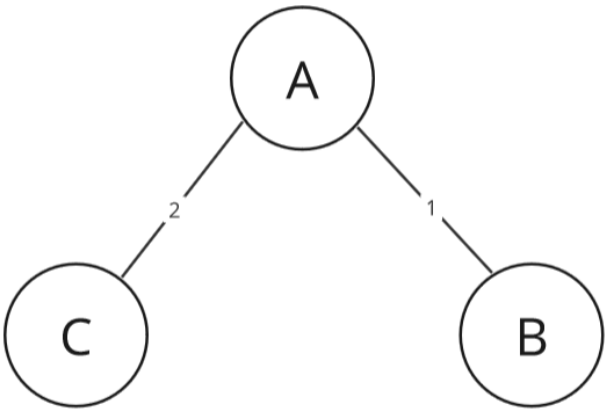


Figura no. 6 fase de descubrimiento de topología simple

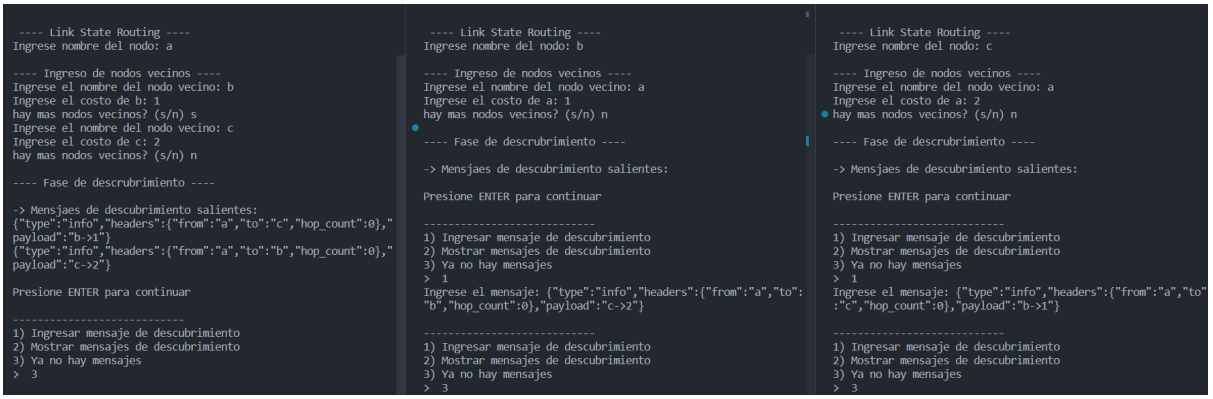
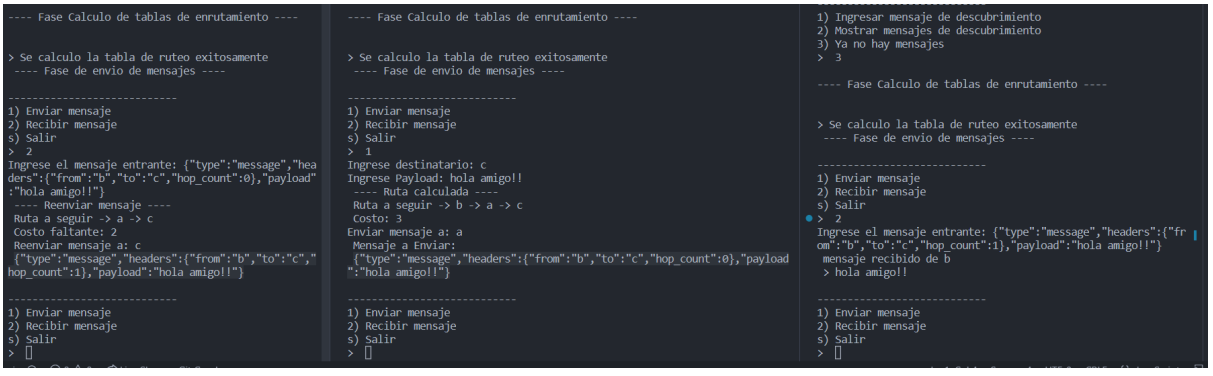


Figura no. 7 Fase de cálculo de tablas de enrutamiento y mensajería



En este ejemplo se simula el envió de mensajes de B hasta C por medio de A. Cabe mencionar que se muestra la ruta calculada al momento de querer enviar un mensaje y la ruta faltante al recibir un mensaje para reenvío.

Distance Vector

Figura no. 8 Topología usada para simulación

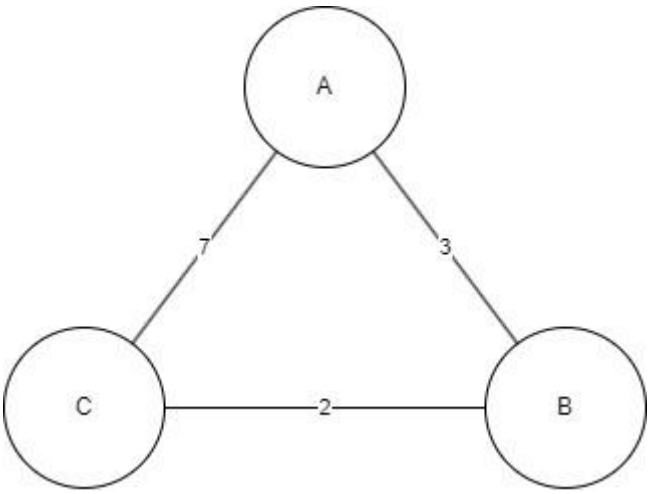


Figura no. 9 Fase de tabla de enrutamiento. Vectores iniciales de cada nodo.

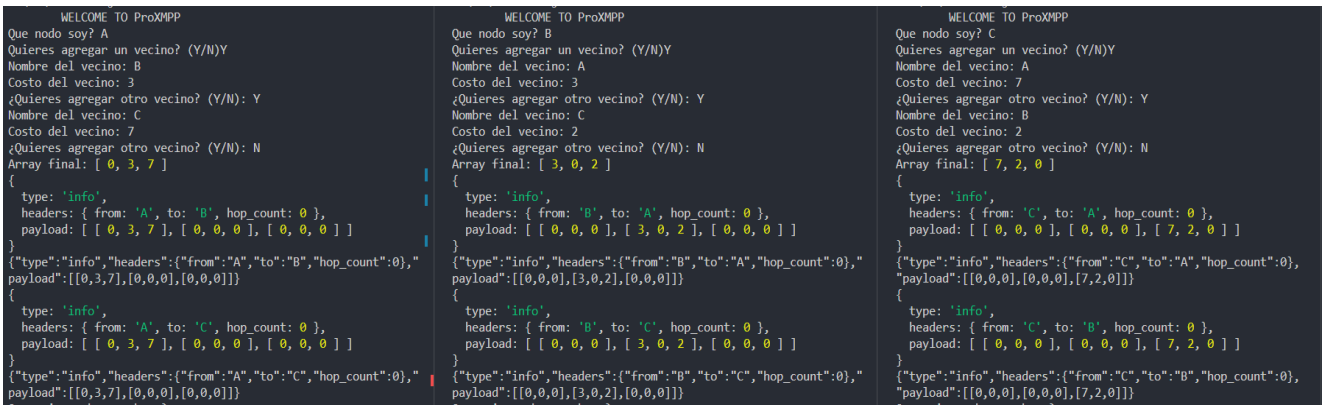


Figura no. 10 Fase de tabla de enrutamiento. Primera iteración de actualizaciones.

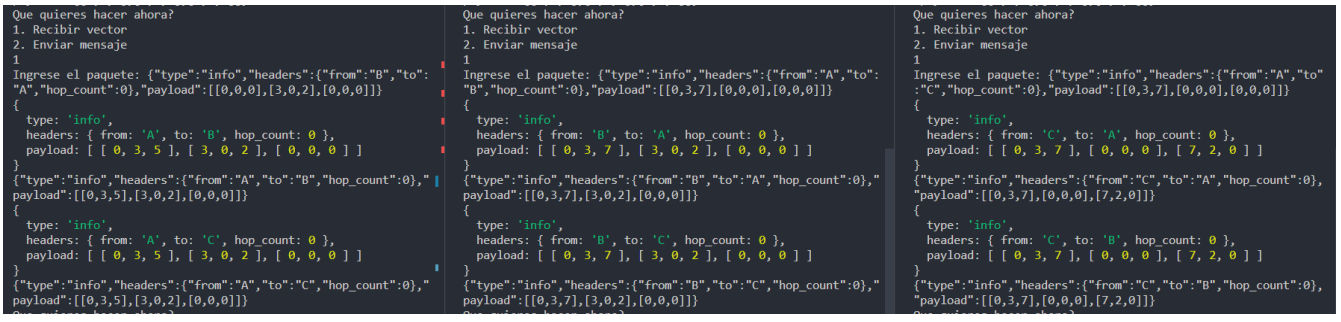


Figura no. 11 Fase de tabla de enrutamiento. Segunda iteración de actualizaciones.

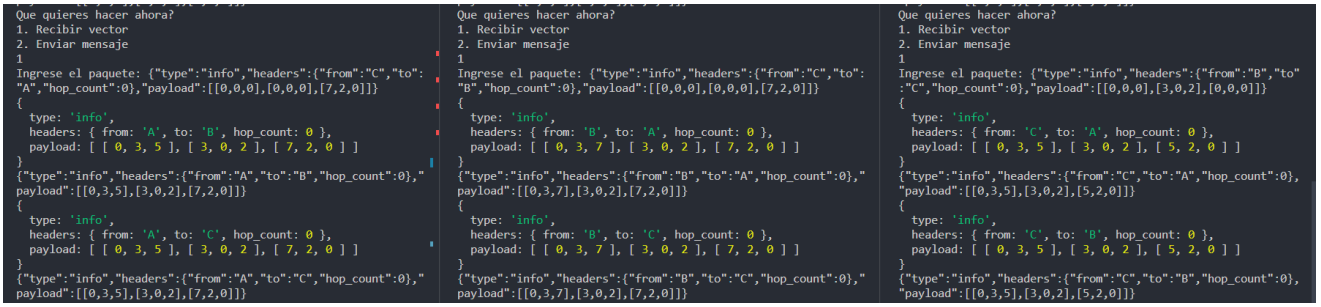


Figura no. 12 Fase de tabla de enrutamiento. Tercera iteración de actualizaciones.

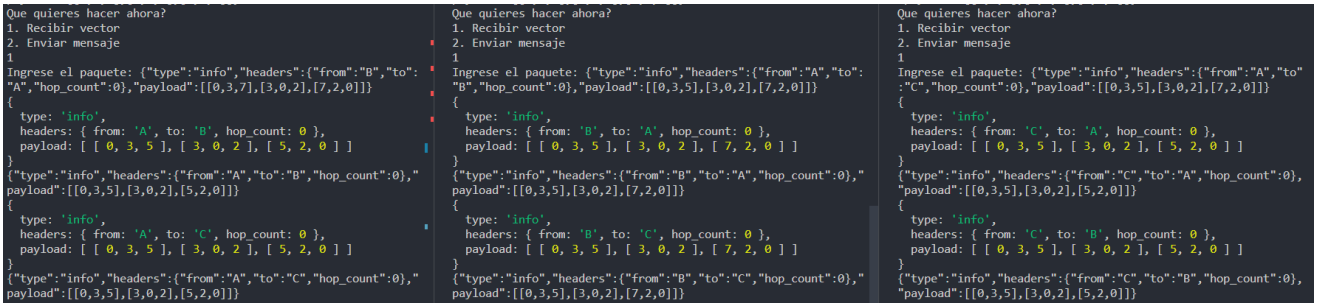


Figura no. 13 Fase de tabla de enrutamiento. Cuarta iteración de actualizaciónes.

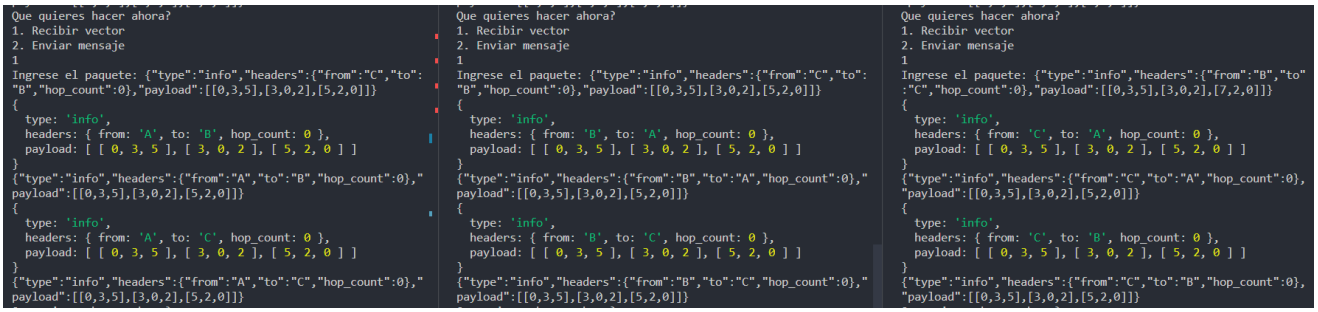
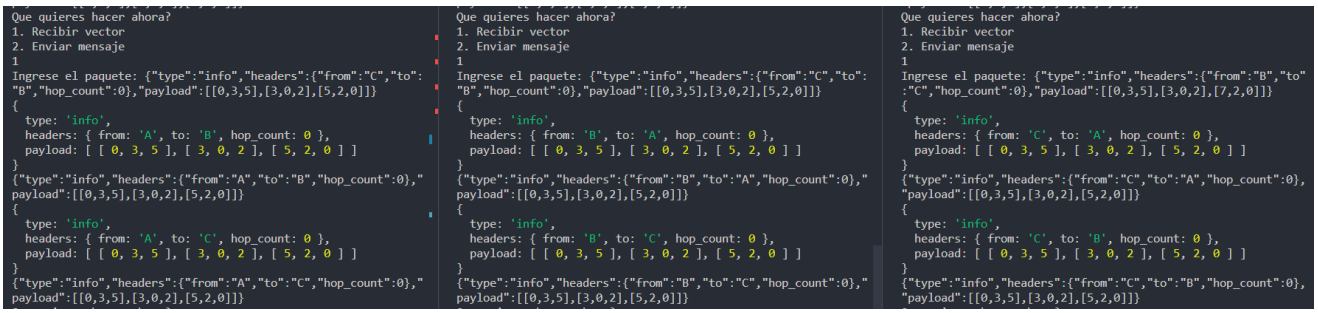
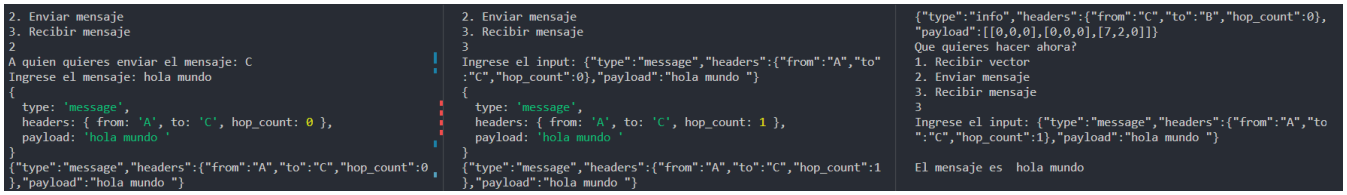


Figura no. 14 Fase de tabla de enrutamiento. Quinta iteración de actualizaciónes.



En este caso es la ultima iteración de la fase de construcción de la tabla de enrutamiento debido a que no hubieron más actualizaciones y todos los nodos tienen la misma tabla.

Figura no. 15, Fase de envío de mensajes por ruta más corta.



Se envía un mensaje desde A hacia C, en donde la ruta más corta es A-B-C

Discusión

En la ejecución de los algoritmos se logró observar la lógica de funcionamiento y la implementación de los mismos en un entorno offline donde el output generado por una consola representando un nodo era ingresado como input en otra consola representando otro nodo. De esta forma se logró representar la topología de nodos en la red cumpliendo un algoritmo de enrutamiento.

En cuanto al algoritmo de Flooding se usó una topología de 4 nodos la cual se puede observar en la sección de anexos del informe. En esta se muestran los resultados cuando el mensaje JSON es creado por el nodo A y este es enviado tanto a el nodo B como al C. Posteriormente ambos de estos nodos reciben el mensaje y envían a sus vecinos. Cabe destacar que el proceso de esto es que el nodo reduce el hop count del JSON y se agrega en el listado de aquellos nodos que ya han recibido dicho mensaje para asegurar no volver a recibirlo. Esto ocurre cuando B envía de regreso al nodo A el mensaje y este mismo no permite recibirlo o no realiza ninguna acción pues ya se encuentra en el listado de aquellos que ya tuvieron el mensaje. Luego en cuanto al nodo C podemos ver que este envía a sus dos vecinos siendo estos el nodo A y el nodo D. Finalmente podemos ver que el nodo D al recibir el JSON este pues observa que es el destinatario y por lo tanto ya no continúa la cadena de flooding pero recibe y muestra el mensaje que recibió en la consola.

En futuras implementaciones del algoritmo para la siguiente entrega del laboratorio parte 2 se puede hacer uso del hop count para asegurar así que el mensaje no sea propagado dentro de la red más de lo esperado y evitar tráfico en la misma. Esto puede ser útil pues se sabe que el algoritmo de Flooding puede llegar a ser algo robusto y demandante.

El algoritmo de Link-State posee varias etapas para su desarrollo, descubrimiento, cálculo de tablas y mensajería. Cada una con su complejidad respectiva y haciendo uso del algoritmo de *Dijkstra* para el cálculo de las tablas de enrutamiento. Cabe mencionar que el número de veces que se ejecuta dijkstra es directamente proporcional al número de nodos de la topología por lo que al aumentar los nodos

de la topología se incrementará considerablemente el *overhead* realizado al momento de calcular las tablas de enrutamiento.

Por otra parte el algoritmo Distance Vector es conocido por su simplicidad y bajo costo computacional, lo cual lo hace especialmente adecuado para redes más pequeñas y estáticas donde los cambios en la topología no suceden con frecuencia. Como se puede observar, su funcionamiento permite ir descubriendo la topología de manera secuencial y ordenada sin la necesidad de conocerla completamente. En cada iteración se van actualizando los costos de enlace de cada uno de los nodos hasta que ya no es necesario hacer nuevas actualizaciones.

Al realizar las pruebas pudimos identificar al ser un procedimiento secuencial y manual, se pudo notar que este algoritmo toma un alto tiempo de convergencia (completar la fase de descubrimiento) y puede generar una mayor complejidad en el control del sistema pues como los resultados lo indican, en una topología de 3 nodos le tomó al algoritmo 5 iteraciones en converger, esto sin tomar en cuenta que en dado caso hayan actualizaciones nuevas o se agreguen nodos en nuestra topología, este número de iteraciones puede aumentar significativamente.

En palabras más simples, el algoritmo de vector de distancia es eficaz para entornos más simples y controlados, pero su eficiencia y eficacia disminuyen en redes más complejas y dinámicas.

Comentario Grupal

Este Laboratorio nos ayudó bastante a entender cómo se manejan los protocolos y algoritmos de ruteo en un entorno real. Consideramos que el resultado del mismo fue exitoso y es un paso en la dirección correcta para la implementación con servidores XMPP. Así mismo nos permitió tener un mejor acercamiento con los trabajos que actualmente se hacen en la industria y nos dió una idea con la cual podemos identificar las ventajas y desventajas y utilizar cada algoritmo según sea necesario y útil.

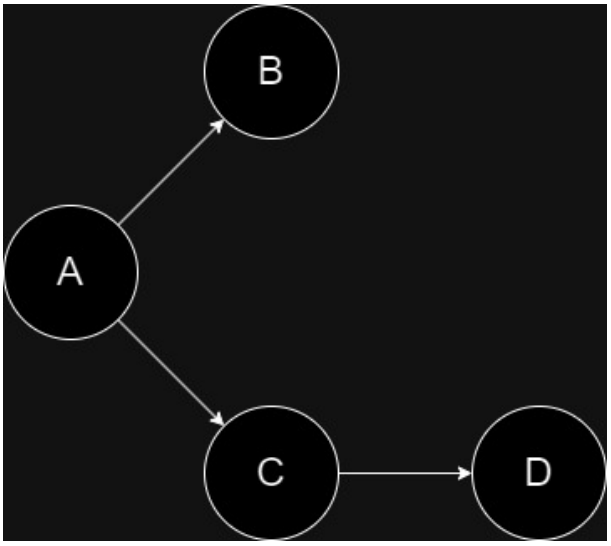
Conclusiones

1. La implementación del algoritmo de Flooding funciona correctamente como se puede observar en los resultados y se buscará implementar hop count en el mismo para evitar alta demanda de red al enviar un mensaje con enrutamiento de Flooding.
2. El algoritmo de Link-State es bastante eficiente ya que hace uso mínimo del medio de transporte (en este caso la red. Sin embargo, posee una mayor complejidad al momento de su implementación y de cálculos realizados.

3. El algoritmo de enrutamiento por vector de distancia es simple y de bajo costo computacional, lo que lo hace ideal para redes pequeñas y menos complejas. Sin embargo, su tiempo de convergencia lento, lo hacen menos adecuado para redes más grandes y dinámicas.

Anexos

Topología utilizada para algoritmo de enrutamiento Flooding



Nodo B enviando mensaje recibido siguiendo algoritmo de Flooding

```
 >> Select an option: 2
Reading flodding message...
 >> Enter the message you have recieved: {"type":"message","headers":{"from":"A","to":"D
","hop_count":"3","recievers":["A"]},"payload":"Hola, hoy voy a comer pizza."}
 >> Node B received message: [object Object]
Node B received message: [object Object]
 >> Message was sent to A by B. The JSON sent was:
{
  "type": "message",
  "headers": {
    "from": "B",
    "to": "D",
    "hop_count": 2,
    "recievers": [
      "A",
      "B"
    ]
  },
  "payload": "Hola, hoy voy a comer pizza."
}

  >>{"type":"message","headers":{"from":"B","to":"D","hop_count":2,"recievers":["A","B"]
},"payload":"Hola, hoy voy a comer pizza."}
```