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 nueva versión de la práctica, los participantes tuvieron como objetivo implementar algoritmos de enrutamiento dentro de una red simulada, utilizando el protocolo XMPP. Los algoritmos clave que se abordaron incluyen Flooding, Distance Vector y Link State Routing. Cada uno de estos algoritmos tiene un enfoque distintivo para actualizar y gestionar las tablas de enrutamiento, que son esenciales para determinar la ruta óptima para enviar mensajes a través de la red.

El algoritmo de Flooding, que forma parte del Link State Routing junto con Dijkstra, opera reenviando un mensaje entrante a todos los nodos vecinos, excepto al nodo emisor. Esta técnica asegura que el mensaje se difunda por toda la red hasta que llegue a su destino. Es especialmente útil en situaciones donde es esencial que un mensaje alcance a todos los puntos finales de la red.

Por otro lado, el algoritmo de Distance Vector se centra en que los nodos compartan sus tablas de enrutamiento entre ellos. Cada nodo mantiene una tabla que refleja la distancia estimada a otros nodos en la red. A medida que los nodos intercambian esta información, ajustan sus decisiones de enrutamiento, garantizando que las decisiones se basen en datos actualizados.

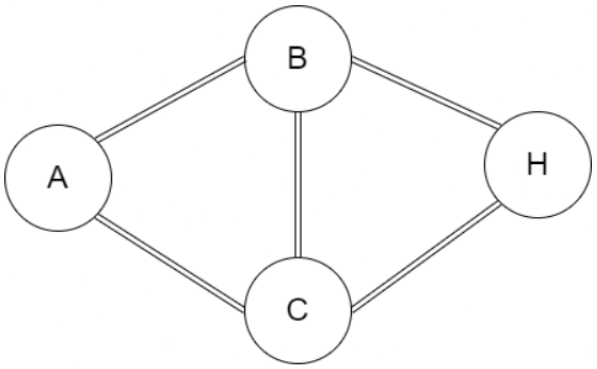
El desarrollo de la práctica se dividió en dos fases. Inicialmente, los algoritmos de enrutamiento se desarrollaron y se probaron de forma manual, proporcionando la información necesaria para la creación y uso de las tablas de enrutamiento. La segunda fase implicó simular esta infraestructura utilizando el chat alumchat.xyz. En este escenario, cada nodo representaba un cliente con una dirección específica @alumchat.xyz, permitiendo el envío y recepción de mensajes.

Para probar los algoritmos, se asignaron direcciones/nombres a los participantes, que utilizaron sus credenciales oficiales de *alumchat*. Se proporcionaron mapas de conexiones entre nodos y archivos con topologías específicas para inicializar los nodos. El desafío principal era garantizar que los algoritmos funcionaran de manera eficiente, adaptándose a cambios como nuevos nodos o nodos inactivos.

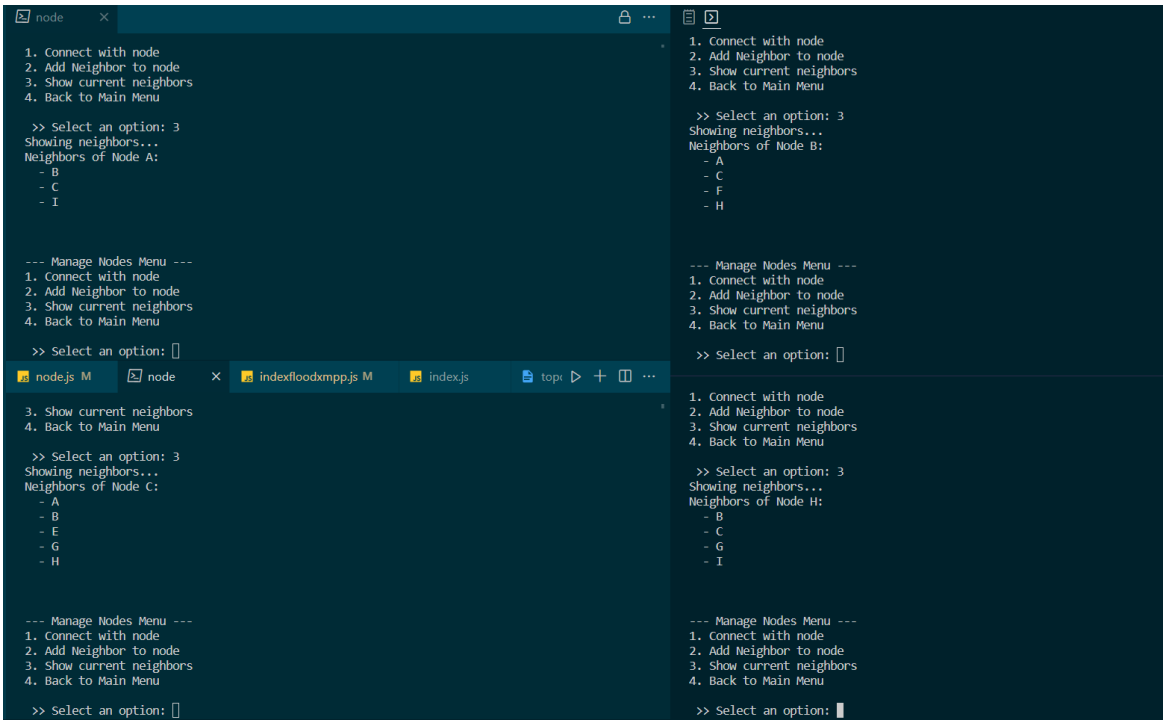
Resultados

Flooding

Topología usada para simulación de Flooding



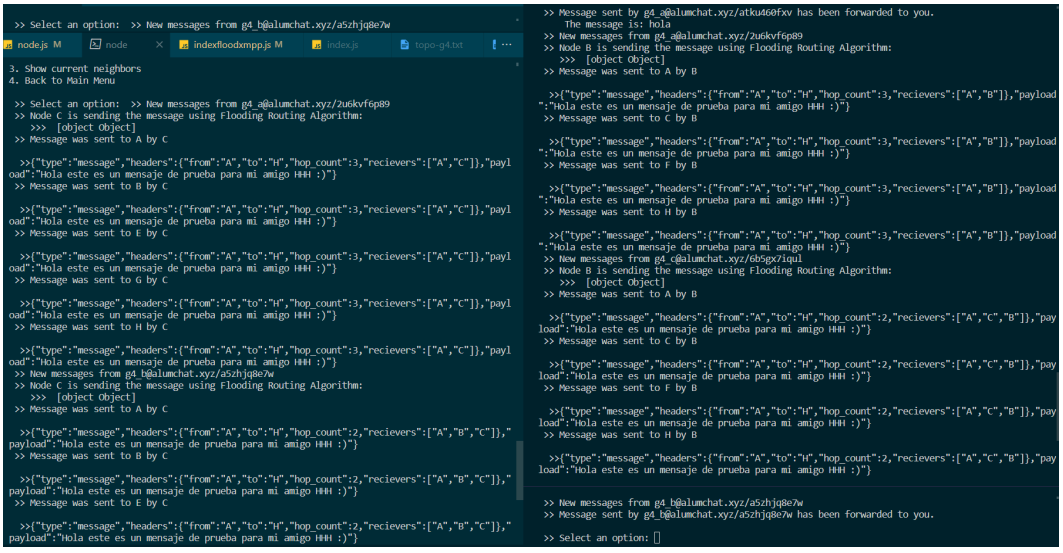
Inicio de sesión con nodos de topología para simulación



Enviar mensaje a H desde A

```
Starting Flooding algorithm...
 >> What message would you like to send:  Hola este es un mensaje de prueba para mi amigo HHH
:)
 >> Who do you want to send the message to?
     Enter the name:  H
 >> How many hops do you want the flooding to happen? Enter the ammount in numbers:  4
 >> Node A is sending the message using Flooding Routing Algorithm:
    >>>  [object Object]
 >> Message was sent to B by A

  >>{"type":"message","headers":{"from":"A","to":"H","hop_count":"4","recievers":["A"]},"paylo
ad":"Hola este es un mensaje de prueba para mi amigo HHH :)"}
 >> Message was sent to C by A

  >>{"type":"message","headers":{"from":"A","to":"H","hop_count":"4","recievers":["A"]},"paylo
ad":"Hola este es un mensaje de prueba para mi amigo HHH :)"}
 >> Message was sent to I by A

  >>{"type":"message","headers":{"from":"A","to":"H","hop_count":"4","recievers":["A"]},"paylo
ad":"Hola este es un mensaje de prueba para mi amigo HHH :)"}
```

Nodos B y C recibiendo mensaje

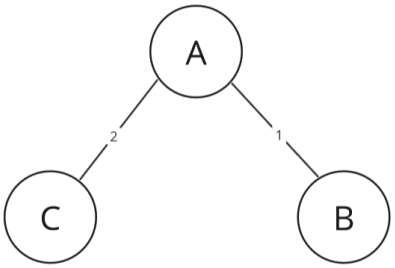


Nodo H recibiendo mensaje

```
 >> Select an option: 3
Showing neighbors...
Neighbors of Node H:
  - B
  - C
  - G
  - I



--- Manage Nodes Menu ---
1. Connect with node
2. Add Neighbor to node
3. Show current neighbors
4. Back to Main Menu

 >> Select an option:  >> New messages from g4_b@alumchat.xyz/a5zhjq8e7w
 >> Message sent by g4_b@alumchat.xyz/a5zhjq8e7w has been forwarded to you.
    The message is: Hola este es un mensaje de prueba para mi amigo HHH :)
 >> New messages from g4_c@alumchat.xyz/6b5gx7iqul
 >> Message sent by g4_c@alumchat.xyz/6b5gx7iqul has been forwarded to you.
    The message is: Hola este es un mensaje de prueba para mi amigo HHH :)
 >> New messages from g4_c@alumchat.xyz/6b5gx7iqul
 >> Message sent by g4_c@alumchat.xyz/6b5gx7iqul has been forwarded to you.
    The message is: Hola este es un mensaje de prueba para mi amigo HHH :)
 >> New messages from g4_b@alumchat.xyz/a5zhjq8e7w
 >> Message sent by g4_b@alumchat.xyz/a5zhjq8e7w has been forwarded to you.
```

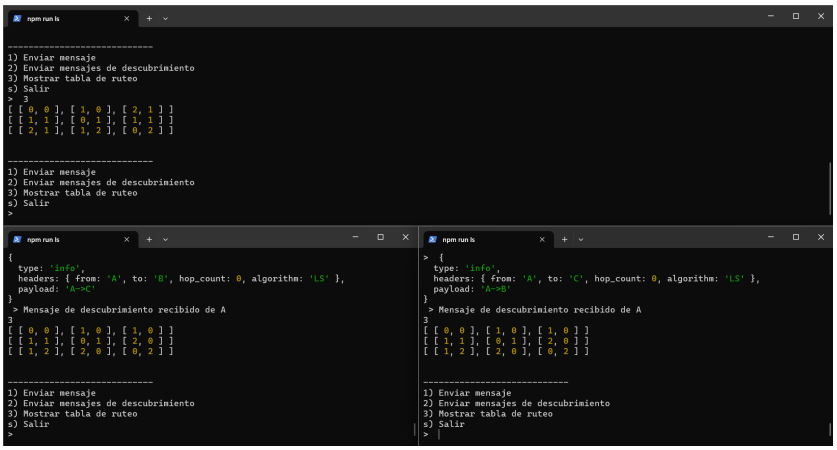
Link State

Topología usada para simulación

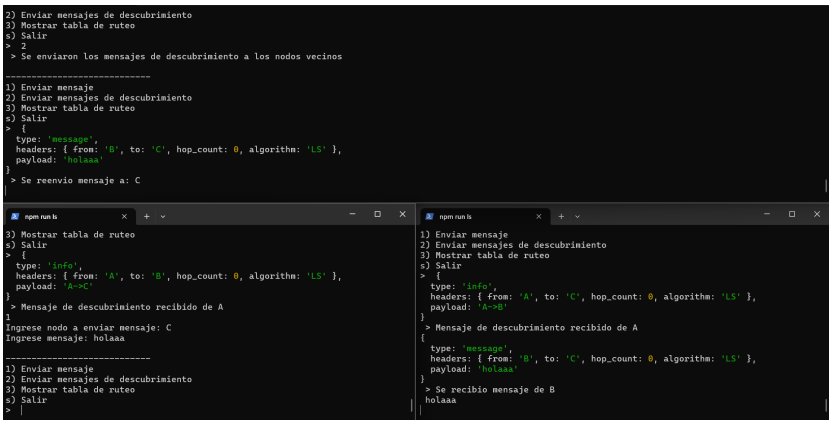


Fase de descubrimiento

Fase de cálculo de tablas de enrutamiento y mensajería

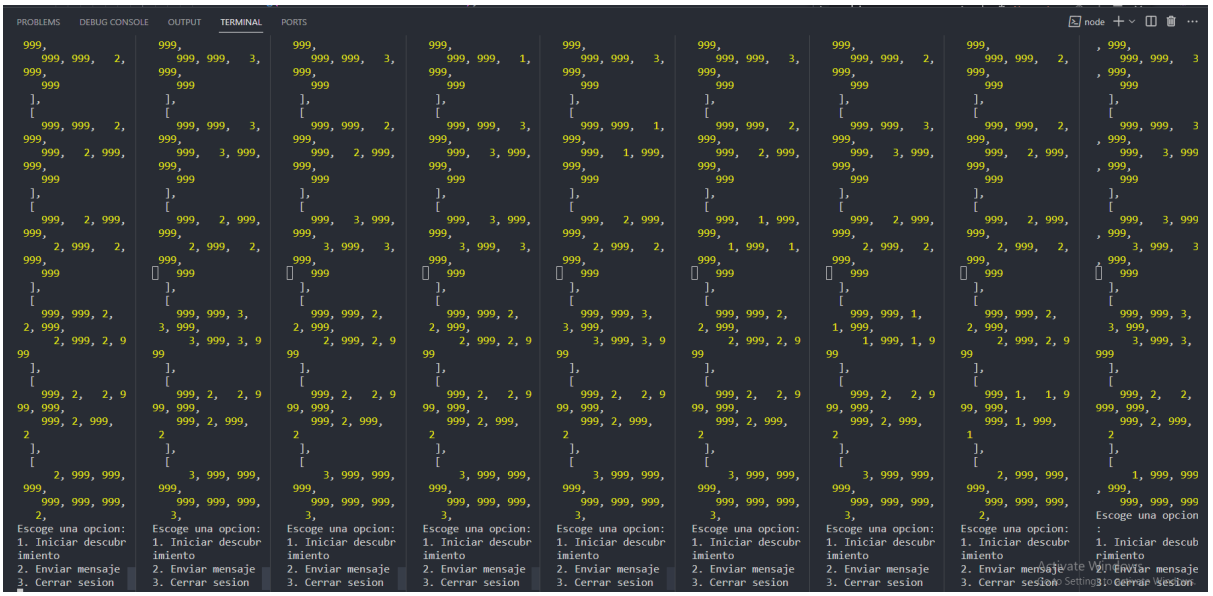


Fase de cálculo de tablas de enrutamiento y mensajería



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



Discusión

La implementación de los algoritmos de enrutamiento en una red simulada sobre el protocolo XMPP proporcionó una experiencia educativa enriquecedora, permitiendo una comprensión más profunda de su funcionamiento en un entorno más realista. A diferencia de la práctica anterior, donde se trabajó en un entorno offline y se utilizó una consola para simular la comunicación entre nodos, en esta ocasión se utilizó el chat alumchat.xyz para simular la infraestructura de la red, lo que añadió una capa adicional de complejidad y realismo al ejercicio.

El algoritmo de Flooding, en esta versión, demostró ser una herramienta poderosa para la propagación de mensajes a través de la red. La incorporación del hop count y la lista de nodos que ya han recibido el mensaje optimizan el proceso, evitando la redundancia y reduciendo el tráfico innecesario en la red. Esta optimización es crucial, ya que el Flooding, por su naturaleza, puede ser demandante y generar tráfico excesivo. Sin embargo, es importante destacar que, aunque eficiente, el Flooding debe ser utilizado con precaución en redes más grandes para evitar la sobrecarga.

El algoritmo Link-State, que integra Dijkstra y Flooding, mostró su complejidad en varias etapas: descubrimiento, cálculo de tablas y mensajería. Es importante destacar que la frecuencia con la que se ejecuta Dijkstra es proporcional al número de nodos. Por lo tanto, a medida que la topología crece, el overhead asociado con el cálculo de las tablas de enrutamiento también aumenta. Esta relación directa plantea desafíos en términos de escalabilidad y eficiencia, especialmente en redes más grandes.

Por otro lado, el algoritmo Distance Vector, aunque simple y de bajo costo computacional, presentó desafíos en redes más grandes. A pesar de ser adecuado para redes pequeñas y estáticas, su tiempo de convergencia puede ser extenso en redes más complejas. Durante las pruebas, se observó que en una topología de tres nodos, el algoritmo tardó cinco iteraciones en converger. Este tiempo puede aumentar significativamente con la adición de nuevos nodos o cambios en la topología, lo que podría llevar a ineficiencias en redes dinámicas.

En resumen, la práctica proporcionó una visión clara de las fortalezas y debilidades de cada algoritmo en diferentes contextos. Mientras que el algoritmo de Flooding se benefició de las optimizaciones introducidas, el Link-State y el Distance Vector mostraron las limitaciones inherentes a su diseño cuando se aplican a redes más grandes y dinámicas. Estas observaciones son esenciales para la toma de decisiones informadas en el diseño e implementación de sistemas de red en el mundo real.

Comentario Grupal

La transición de un entorno offline a una implementación basada en XMPP nos ofreció una perspectiva única sobre los desafíos y consideraciones que surgen al trabajar con tecnologías de comunicación en tiempo real. Esta experiencia nos ha preparado adecuadamente para futuras implementaciones en servidores XMPP y nos ha proporcionado una base sólida sobre la cual construir. Además, el laboratorio nos permitió analizar críticamente las ventajas y desventajas de cada algoritmo. A través de pruebas y observaciones, hemos ganado una comprensión más profunda de cuándo y cómo aplicar cada algoritmo de manera óptima según las necesidades específicas de una red.

Conclusiones

1. Es posible simular el comportamiento de una red utilizando un servidor XMPP y distintos clientes.
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.
4. La implementación del algoritmo de Flooding funciona correctamente como se puede observar en los resultados.

Anexos