Universidad del Valle de Guatemala
Redes
Elean Rivas - 19062
Javier Alvarez - 18051
Javier Ramirez - 18099

Laboratorio 3 - Segunda Parte
Algoritmos de Enrutamiento

Algoritmos utilizados

Los algoritmos implementados para este laboratorio fueron:

- Dijkstra: El algoritmo de Dijkstra es un algoritmo de búsqueda de rutas utilizado en teoría de grafos para encontrar el camino más corto desde un nodo de origen a todos los demás nodos en un grafo ponderado no dirigido o dirigido. El algoritmo de Dijkstra garantiza encontrar las rutas más cortas en grafos con pesos positivos ya que no funciona correctamente en grafos con pesos negativos debido a la posibilidad de ciclos negativos.
- Distance vector routing: Este algoritmo tiene como finalidad obtener las rutas más cortas entre un nodo y sus vecinos directos. Esto se hace mediante tablas que genera cada nodo en la topología, que además están sujetas a cambios en base a las tablas calculadas por sus vecinos más cercanos.

La mayor diferencia entre este laboratorio y el anterior es que en este laboratorio se hizo la funcionalidad de los algoritmos de enrutamiento al cliente desarrollado para funcionar con el protocolo XMPP.

Resultados

- Distance Vector Routing

  Para este ejemplo, se utilizó la siguiente topología, agregando el prefijo *“ram18099-”* a cada nodo para la correcta funcionalidad en el servidor de XMPP:

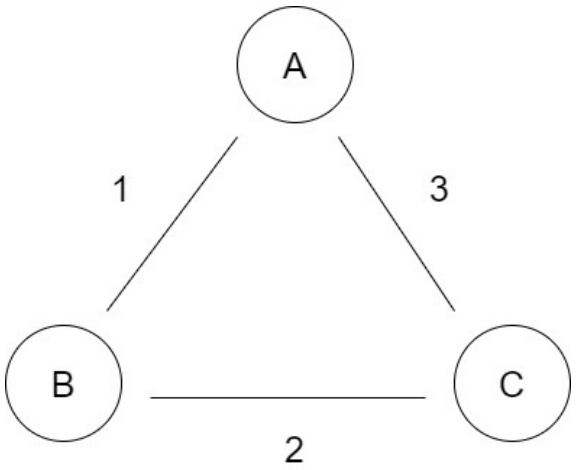

Figura 1: Topología simple de prueba para el algoritmo de Distance Vector.

Las tablas actualizadas (Initial Step y Update Step aplicados) para los nodos A, B y C fueron los siguientes:

```
========== ram18099-A Session ==========
1. Display User/s information
2. Manage contacts
3. Manage Account
4. Chat
5. Routing Options
6. Close session
> Enter option: 5

========== ROUTING OPTIONS ==========
1. Read JSON file
2. Write new Message
3. Show Distance Vector Table
4. Cancel
> Enter option: 3
======= DISTANCE VECTOR =======
Node: ram18099-A, Distance: 0, Hop: ram18099-A
Node: ram18099-B, Distance: 1, Hop: ram18099-B
Node: ram18099-C, Distance: 3, Hop: ram18099-C,ram18099-B
```

Figura 2: Tabla de Distance Vector final del nodo A (ram18099-A).

```
========== ram18099-B Session ==========
1. Display User/s information
2. Manage contacts
3. Manage Account
4. Chat
5. Routing Options
6. Close session
> Enter option: 5

========== ROUTING OPTIONS ==========
1. Read JSON file
2. Write new Message
3. Show Distance Vector Table
4. Cancel
> Enter option: 3
======= DISTANCE VECTOR =======
Node: ram18099-B, Distance: 0, Hop: ram18099-B
Node: ram18099-A, Distance: 1, Hop: ram18099-A
Node: ram18099-C, Distance: 2, Hop: ram18099-C
```

Figura 3: Tabla de Distance Vector final del nodo B (ram18099-B).

```
========== ram18099-C Session ==========
1. Display User/s information
2. Manage contacts
3. Manage Account
4. Chat
5. Routing Options
6. Close session
> Enter option: 5

========== ROUTING OPTIONS ==========
1. Read JSON file
2. Write new Message
3. Show Distance Vector Table
4. Cancel
> Enter option: 3
======= DISTANCE VECTOR =======
Node: ram18099-C, Distance: 0, Hop: ram18099-C
Node: ram18099-B, Distance: 2, Hop: ram18099-B
Node: ram18099-A, Distance: 3, Hop: ram18099-A,ram18099-B
```

Figura 4: Tabla de Distance Vector final del nodo C (ram18099-C).

Ejemplo de transmisión de un mensaje desde el nodo A al nodo C y con el payload de “Hola ram18099-C, att. ram18099-A”.

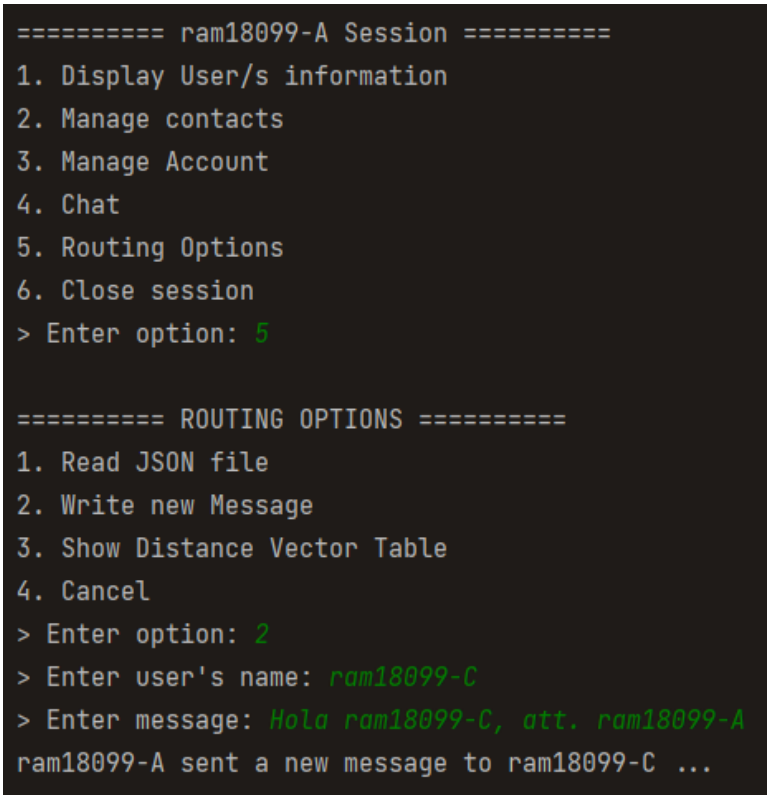

Figura 5: Mensaje enviado desde el Nodo A al nodo C con un mensaje personalizado.

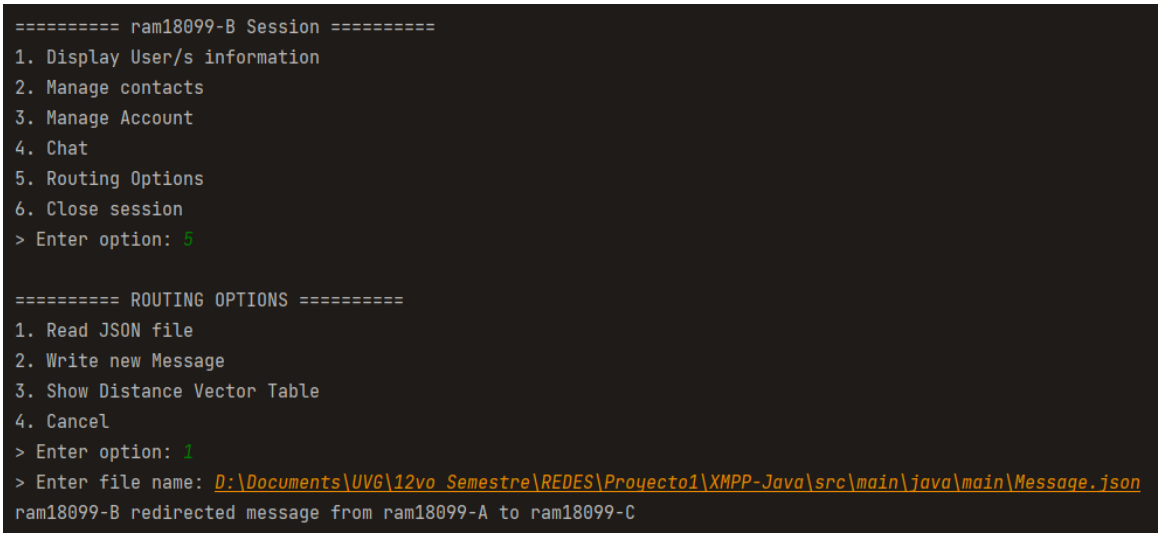

Figura 6: Mensaje recibido por el Nodo B y redirigido al nodo C.

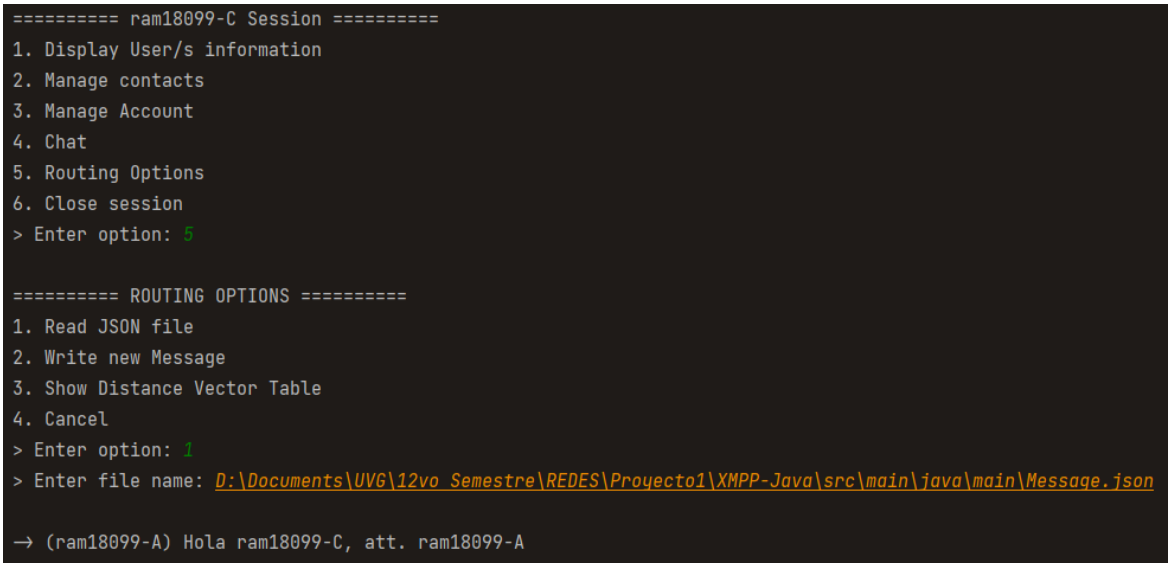

Figura 7: Mensaje recibido por el nodo C con origen en el nodo A.

Discusión

Para el algoritmo de Distance Vector, la tabla inicial de cada nodo se calcula al momento en que un usuario hace login con el cliente, luego de verificar que el nombre del nodo que ingrese el usuario se encuentre dentro de la topología definida previamente. Una vez que la tabla inicial ha sido calculada, el programa procede a recibir mensajes de tipo “message” (mensaje de un nodo A a un nodo B), “info” (Mensaje de actualización por parte de otro nodo) o “echo” (Mensaje de reconocimiento).

En caso que se pase un mensaje de tipo “info”, el programa procede a parsear un archivo de tipo.json con la información de las tablas calculadas por el nodo emisor, para después comparar las nuevas distancias calculadas con el algoritmo de Bellman-Ford. En caso que la nueva distancia calculada sea igual a la que se encuentra en la tabla, el programa procede a añadir al nodo emisor como canal alternativo de enrutamiento de mensajes. En caso que la distancia calculada sea menor a la que se encuentra en la tabla, se procede a actualizar la tabla con el nodo emisor como único canal de transmisión de mensajes.

En caso que el mensaje sea de tipo “message”, el usuario revisa el destinatario del mensaje. Si el destinatario coincide con su propio nombre de usuario, el cliente procede a mostrar el mensaje recibido. En caso que el destinatario sea distinto, el cliente procede a redirigir dicho mensaje con ayuda de su tabla de enrutamiento, con el mismo mensaje, nodo destino y destinatario, agregando un salto más a la cantidad de saltos realizados.

Para la transmisión de un mensaje, se puede observar que las Figuras 5 - 7 muestran el correcto funcionamiento del enrutamiento de un mensaje. La Figura 5 muestra el Nodo A enviando el mensaje *“Hola ram18099-C, att. ram18099-A”* al Nodo C. Debido a que la tabla de enrutamiento en la Figura 2 incluye al nodo B como ruta para transmisión de mensajes, el mensaje pasa a través del Nodo B antes de llegar al Nodo C. La Figura 6 muestra que el Nodo

B recibe el mensaje de A, pero detecta que el mensaje no le corresponde, por lo que redirige el mensaje al Nodo C, agregando un salto más a los saltos registrados para el mensaje. Finalmente el mensaje llega a C, el cual lo recibe de parte del nodo B. El Nodo C al leer el mensaje detecta que dicho mensaje le corresponde, por lo que el cliente decide mostrar dicho mensaje al usuario, como se muestra en la Figura 7.

Comentario grupal

Para este laboratorio, la implementación de los algoritmos de enrutamiento de mensajes se vio afectada por la complejidad que tiene el incluir esta funcionalidad dentro del cliente previamente desarrollado. Para futuras implementaciones se sugiere agregar como comentario al desarrollo del proyecto (Cliente de protocolo XMPP) la consideración de funcionalidades para transmitir mensajes por medio de archivos JSON, además de permitir que el servidor permita enviar dichos archivos a otros usuarios.

Conclusiones

En base a lo realizado en este laboratorio, se llegaron a las siguientes conclusiones:

- El uso de protocolos para intercambio de mensajes a través de redes con diferentes topologías es de gran importancia, debido a que se establece una única forma de comunicación y esto a su vez permite rastrear los mensajes a través de la red y detectar posibles errores producidos durante su transmisión.
- El intercambio y recepción de mensajes a través de nodos en una red es compleja y requiere de una comunicación constante e iterativa, de modo que todos los nodos tengan la mayor información posible. Para este laboratorio se asume que todos los nodos están disponibles en todo momento, pero en casos reales es necesario evaluar constantemente si los nodos se encuentran disponibles dentro de una red.

Referencias

- https://www.geeksforgeeks.org/dijkstras-shortest-path-algorithm-greedy-algo-7/