UNIVERSIDAD DEL VALLE DE GUATEMALA

CC 3067 Redes

Sección 10

“Laboratorio 03”

Algoritmos de enrutamiento

ANDREA DE LOURDES LAM PELAEZ 20102
GABRIELA PAOLA CONTRERAS GUERRA 20213
ANA PAOLA DE LEON MOLINA 20361

Guatemala, 03 de Septiembre de 2023

Introducción

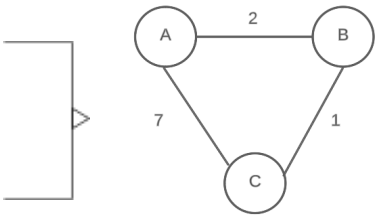
El fin de este laboratorio fue conocer y comprender el funcionamiento de las tablas de enrutamiento por medio de la implementación de tres algoritmos diferentes, siendo estos Flooding, Distance vector routing y Linked state, cabe mencionar que estos algoritmos son actualmente utilizados por el internet lo cual permite tener un acercamiento más profundo.

Esta actividad fue realizada en trios y quedó a discreción de estas que algoritmo desarrolló cada una y el lenguaje de programación a utilizar. Para la realización de este se debía establecer una topología con la cual se iba a trabajar, seguidamente por medio del hard coding se debía simular/proveer los inputs requeridos por cada uno de los algoritmos para así lograr que estos funcionaran adecuadamente. Cabe destacar que cada uno de estos debe de generar un paquete en el cual se establezca el tipo, el header el cual contendrá el nodo origen, destino y la cantidad de saltos; y por último el payload el cual contendrá el mensaje a enviar. Al finalizar se discutieron diferentes aspectos por medio de los cuales se puede concluir que los algoritmos de enrutamiento son componentes esenciales para la gestión de redes puesto a que desempeñan un papel crucial en la entrega eficiente de datos.

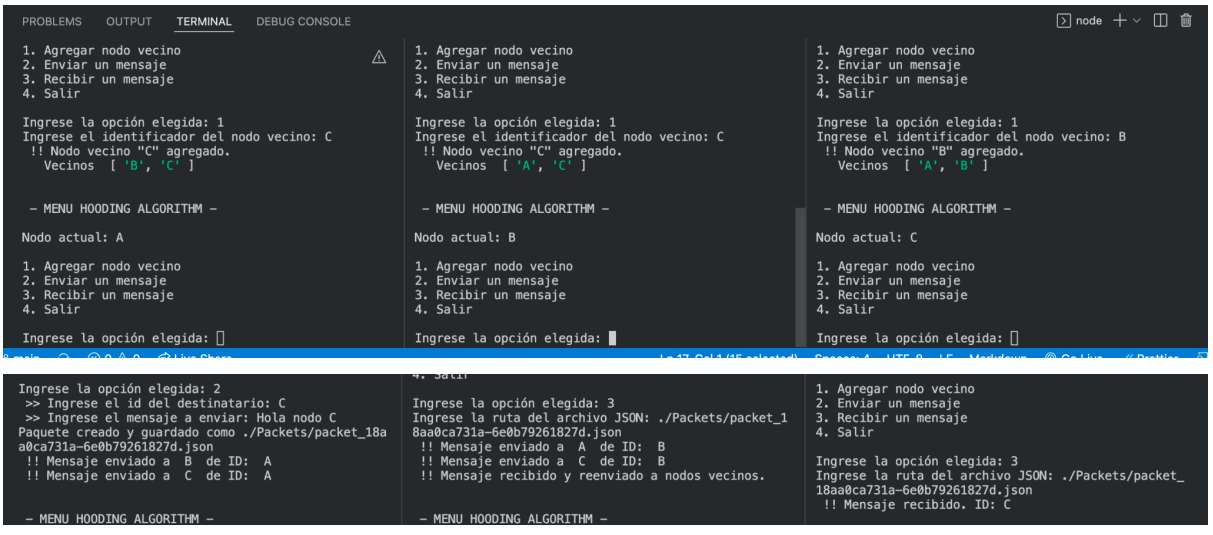
Resultados

- **Flooding**
  - Topología utilizada

A: (B, C)
B: (A, C)
C: (A, B)

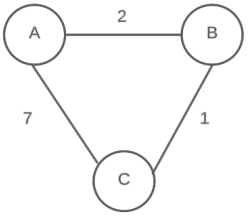


  - Comunicación Final



- **Distance Vector Routing**
  - Topología utilizada

A: (B,2) , (C,7)
B: (A,2) , (C,1)
C: (A,7) , (B,1)



  - Tablas de los vecinos

```
Que nodo eres: A

--- MENU PARA CONOCER VECINOS ---
1. Ingresar vecinos
2. Salir
Ingresa tu elección: 1

 --- INGRESAR NODO VECINO ---.
Ingresa Nodo vecino: B
Ingresa costo: 2
Vecino ingresado: B, Costo: 2


--- MENU PARA CONOCER VECINOS ---
1. Ingresar vecinos
2. Salir
Ingresa tu elección: 1

 --- INGRESAR NODO VECINO ---.
Ingresa Nodo vecino: C
Ingresa costo: 7
Vecino ingresado: C, Costo: 7


--- MENU PARA CONOCER VECINOS ---
1. Ingresar vecinos
2. Salir
Ingresa tu elección: 2

 *MATRIZ INICIAL*.
[ [ 0, 2, 7 ], [ 'null', 'null', 'null' ],
  [ 'null', 'null', 'null' ] ]
```

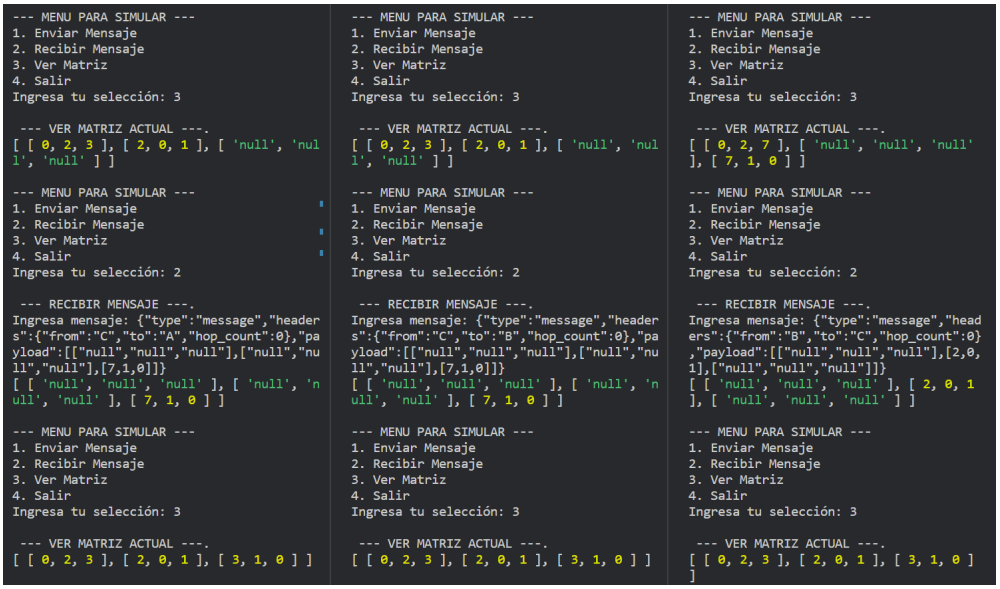
```
Que nodo eres: B

--- MENU PARA CONOCER VECINOS ---
1. Ingresar vecinos
2. Salir
Ingresa tu elección: B
Opción inválida. Por favor, selecciona una
 opción válida.

--- MENU PARA CONOCER VECINOS ---
1. Ingresar vecinos
2. Salir
Ingresa tu elección: 1

 --- INGRESAR NODO VECINO ---.
Ingresa Nodo vecino: A
Ingresa costo: 2
Vecino ingresado: A, Costo: 2


--- MENU PARA CONOCER VECINOS ---
1. Ingresar vecinos
2. Salir
Ingresa tu elección: 1

 --- INGRESAR NODO VECINO ---.
Ingresa Nodo vecino: C
Ingresa costo: 1
Vecino ingresado: C, Costo: 1


--- MENU PARA CONOCER VECINOS ---
```

```
Que nodo eres: C

--- MENU PARA CONOCER VECINOS ---
1. Ingresar vecinos
2. Salir
Ingresa tu elección: 1

 --- INGRESAR NODO VECINO ---.
Ingresa Nodo vecino: A
Ingresa costo: 7
Vecino ingresado: A, Costo: 7


--- MENU PARA CONOCER VECINOS ---
1. Ingresar vecinos
2. Salir
Ingresa tu elección: 1

 --- INGRESAR NODO VECINO ---.
Ingresa Nodo vecino: B
Ingresa costo: 1
Vecino ingresado: B, Costo: 1


--- MENU PARA CONOCER VECINOS ---
1. Ingresar vecinos
2. Salir
Ingresa tu elección: 2

 *MATRIZ INICIAL*.
[ [ 'null', 'null', 'null' ], [ 'null',
'null', 'null' ], [ 7, 1, 0 ] ]
```

```
B : {"type":"message","headers":{"from":"A
","to":"B","hop_count":0},"payload":[[0,2,
7],["null","null","null"],["null","null","
null"]]}
C : {"type":"message","headers":{"from":"A
","to":"C","hop_count":0},"payload":[[0,2,
7],["null","null","null"],["null","null","
null"]]}

--- MENU PARA SIMULAR ---
1. Enviar Mensaje
2. Recibir Mensaje
3. Ver Matriz
4. Salir
Ingresa tu selección: 2

 --- RECIBIR MENSAJE ---.
Ingresa mensaje: {"type":"message","header
s":{"from":"B","to":"A","hop_count":0},"pa
yload":[["null","null","null"],[2,0,1],["n
ull","null","null"]]}
[ [ 'null', 'null', 'null' ], [ 2, 0, 1 ],
  [ 'null', 'null', 'null' ] ]

--- MENU PARA SIMULAR ---
1. Enviar Mensaje
```

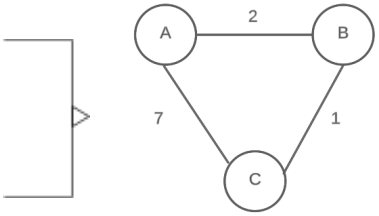
```
 *MATRIZ INICIAL*.
[ [ 'null', 'null', 'null' ], [ 2, 0, 1 ],
  [ 'null', 'null', 'null' ] ]
A : {"type":"message","headers":{"from":"B
","to":"A","hop_count":0},"payload":[["nul
l","null","null"],[2,0,1],["null","null","
null"]]}
C : {"type":"message","headers":{"from":"B
","to":"C","hop_count":0},"payload":[["nul
l","null","null"],[2,0,1],["null","null","
null"]]}

--- MENU PARA SIMULAR ---
1. Enviar Mensaje
2. Recibir Mensaje
3. Ver Matriz
4. Salir
Ingresa tu selección: 2

 --- RECIBIR MENSAJE ---.
Ingresa mensaje: {"type":"message","header
s":{"from":"A","to":"B","hop_count":0},"pa
yload":[[0,2,7],["null","null","null"],["n
ull","null","null"]]}
[ [ 0, 2, 7 ], [ 'null', 'null', 'null' ],
  [ 'null', 'null', 'null' ] ]
```

```
A : {"type":"message","headers":{"from":
"C","to":"A","hop_count":0},"payload":[[
"null","null","null"],["null","null","nu
ll"],[7,1,0]]}
B : {"type":"message","headers":{"from":
"C","to":"B","hop_count":0},"payload":[[
"null","null","null"],["null","null","nu
ll"],[7,1,0]]}

--- MENU PARA SIMULAR ---
1. Enviar Mensaje
2. Recibir Mensaje
3. Ver Matriz
4. Salir
Ingresa tu selección: 2

 --- RECIBIR MENSAJE ---.
Ingresa mensaje: {"type":"message","head
ers":{"from":"A","to":"C","hop_count":0}
,"payload":[[0,2,7],["null","null","null
"],["null","null","null"]]}
[ [ 0, 2, 7 ], [ 'null', 'null', 'null'
], [ 'null', 'null', 'null' ] ]

--- MENU PARA SIMULAR ---
1. Enviar Mensaje
```



- **Link State Routing**
  - Topología utilizada

A: (B,2) , (C,7)
B: (A,2) , (C,1)
C: (A,7) , (B,1)



  - Tablas de los Nodos

```
1. Enviar mensaje
2. Recibir mensaje
3. Agregar relación a la topología
4. Salir
>> 3
Primer nodo:
>> A
Segundo nodo:
>> C
Peso de la relación:
>> 7
Relación agregada: A -> C (Peso: 7)

1. Enviar mensaje
2. Recibir mensaje
3. Agregar relación a la topología
4. Salir
>>
```

Imagen: Ejemplo de cómo se agrega una relación

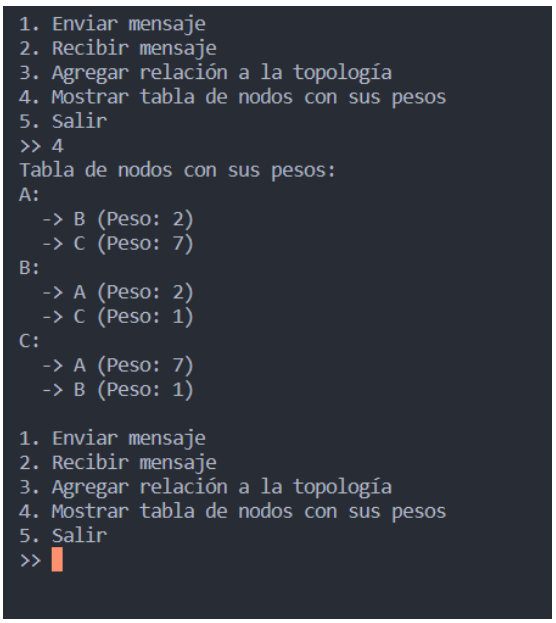


Imagen: Tabla de los nodos

Discusión

El objetivo del laboratorio fue conocer las tablas de enrutamiento mediante la implementación de algoritmos de enrutamiento actuales, para cumplir dicho objetivo se elaboraron los algoritmos de Flooding, Distance vector routing y Linked state los cuales fueron elaborados siguiendo cada una de sus reglas y siendo empaquetados de la manera solicitada.

En la realización del algoritmo de Flooding se realizó una simulación en donde se logra comprender a grandes rasgos el funcionamiento del mismo, pues al realizar el envío de un mensaje este es transmitido a todos los nodos vecinos hasta llegar al destinatario deseado. Se logró identificar que, a pesar de no ser un algoritmo eficiente, logra darnos una idea de cómo funciona el empaquetamiento y evnío de información utilizando nodos y un grafo de referencia.

El protocolo de enrutamiento distance vector se utiliza para calcular la distancia más contra entre nodos este se basa en obtener un vector de distancia el cual calcula la distancia y la dirección del vector del siguiente salto a partir de la información obtenida por el nodo vecino (Jadhav, 2022b). Mediante el envío de estos vectores se va formando una tabla de vectores la cual se va actualizando en cada envío. Dicha actualización hace uso del algoritmo de Bellman-Ford la cual se es el encargado de calcular la distancia más corta entre nodos, esta se define como $d_x(y) = min\{c(x,v) + d_v(y)\}$ en donde c (x ,v) es el costo del nodo x de cada uno de sus vecinos v y d_v (y) es la distancia a cada nodo desde el nodo inicial (GeeksforGeeks, 2019).

En base a los resultados obtenidos en el apartado de distance vector, se puede observar que el algoritmo está realizado correctamente, pues este es capaz de realizar una

matriz con distancias infinitas hacia los nodos desconocidos y en base a la información obtenida saca el vector distancia y luego al recibir mensajes logra editar la tabla previamente realizada con el nuevo vector y empaqueta está para ser enviada. Cabe resaltar que al momento de recibir un mensaje este es capaz de desempaquetar el mensaje y en base a la información recibida actualiza la tabla actual, seguidamente realiza la ecuación Bellman Ford para calcular la distancia más corta; lo anterior se repite n veces hasta lograr que la tabla converga, lo cual se puede observar en las figuras adjuntas, de esa manera sabemos que ya se ha obtenido la ruta más corta. Es importante destacar que para el funcionamiento de este debe de tener conocimiento de todos los nodos de la topología y se le debe de indicar por medio de un input todos los nodos vecinos del nodo que se está simulando, es decir nodo actual a simular, para así poder realizar lo anteriormente mencionado.

El Link State Routing (enrutamiento por estado de enlace) es un enfoque sofisticado para el enrutamiento en redes, que utiliza la información detallada sobre el estado de los enlaces en la red para calcular rutas eficientes. El código es una implementación simplificada de este concepto en un programa interactivo. El código implementa un programa de enrutamiento básico utilizando el algoritmo de Link State Routing. El programa permite a los usuarios interactuar a través de la consola para realizar diversas acciones. Pueden enviar y recibir mensajes entre nodos de una red simulada, agregar nuevas conexiones entre nodos a la topología de la red y mostrar una tabla que lista los nodos y sus pesos de conexión. La lógica central del programa se basa en mantener una topología de red que representa las conexiones entre los nodos, calcular las rutas más cortas utilizando el algoritmo de Dijkstra y permitir que los usuarios interactúen con la red para enviar mensajes y administrar las conexiones. Además, el programa guarda y carga la topología desde un archivo JSON para persistencia de datos.

Comentario grupal

Como grupo consideramos que este laboratorio ha sido uno de los más complejos no en cuanto a su realización sino que en relación a entender que es lo que se estaba solicitando. Pues desde un inicio surgieron muchas dudas las cuales nos impiden poder avanzar de una forma fluida, estas con el paso de la semana fueron solucionadas y nos permitió finalizar el laboratorio. Por otra parte, al momento de estar programando todas concordamos en que logramos ver un poco acerca de las limitaciones que cada algoritmo tiene en cuanto a comunicación y conocimiento de la red con la cual se trabajó. Al concluir esta práctica pudimos entender la importancia que tiene configurar bien cada uno de estos pues son parte esencial para garantizar una comunicación efectiva a bajo costo y un funcionamiento eficiente de la red.

Conclusiones

- En conclusión, el algoritmo de vector distancia, a pesar de su simplicidad, tiende a ser ineficiente en términos de velocidad debido a las frecuentes actualizaciones de rutas en cada nodo, lo que puede complicar su convergencia, lo cual lo vuelve menos eficiente para redes que requieran comunicación rápida.

- El Link State Routing se basa en la recopilación de información detallada sobre las conexiones de red y utiliza el algoritmo de Dijkstra para calcular las rutas más cortas entre los nodos. De igual manera la implementación brinda una comprensión básica de cómo funcionan los enrutadores en una red real y cómo se mantienen y actualizan las tablas de enrutamiento para tomar decisiones inteligentes sobre la transmisión de datos en una red de comunicación.

- El algoritmo de Flooding demostró ser bastante ineficiente tanto a nivel computacional como a nivel teórico. Sin embargo, nos permitió comprender la base del empaquetamiento de información y enrutamiento utilizando grafos.

Cita y referencias

- Jadhav, S. (2022, 7 octubre). Distance Vector Routing Algorithm | Scaler Topics. *Scaler Topics*. https://www.scaler.com/topics/computer-network/distance-vector-routing-algorithm/

- GeeksforGeeks. (2019). Distance Vector Routing DVR Protocol. *GeeksforGeeks*. https://www.geeksforgeeks.org/distance-vector-routing-dvr-protocol/