Universidad del Valle de Guatemala
Facultad de Ingeniería
Departamento de Ciencias de la Computación
CC3067 Redes
Catedrático: Jorge Yass
Ciclo 2 de 2022

Laboratorio 3
Algoritmos de enrutamiento

Link del Repositorio:
https://github.com/bryannalfaro/Redes-Lab3

Bryann Alfaro 19372
Diego Arredondo 19422
Julio Herrera 19402

Guatemala, 1 de septiembre de 2022

Universidad del Valle de Guatemala
CC3067 Redes
Catedrático: Jorge Yass
Ciclo 2 de 2022

Descripción

Esta práctica consistió en la implementación de tres distintos algoritmos de enrutamiento con el fin de poder comprender el funcionamiento característico de cada uno de ellos y poder tener un mejor conocimiento de estos en un ambiente real.

De igual manera, la forma de implementar este ecosistema fue por medio del servidor alumchat.fun , en donde se tendrá una topología y cada nodo será representado por un estudiante con usuarios de la forma carnetusuario@alumchat.fun. Y se enviará un mensaje con el objetivo de que le llegue correctamente al usuario destino mostrando cierta información como los nodos recorridos, el usuario destino, los saltos, etc.

Los algoritmos implementados fueron:

- **Flooding**

  El algoritmo flooding también se le conoce como “Inundación de red” , es un algoritmo relativamente sencillo de implementar. Su funcionamiento se basa en enviar el paquete de información por cada nodo de salida que tenga cada nodo. Una restricción importante es que no se puede recibir si el paquete le llega al nodo emisor.

  Este algoritmo tiene la ventaja que se asegura la entrega del paquete, sin embargo, puede resultar poco eficiente ya que se estaría provocando en algún caso un reenvío constante de un paquete al existir un bucle dentro de la topología.

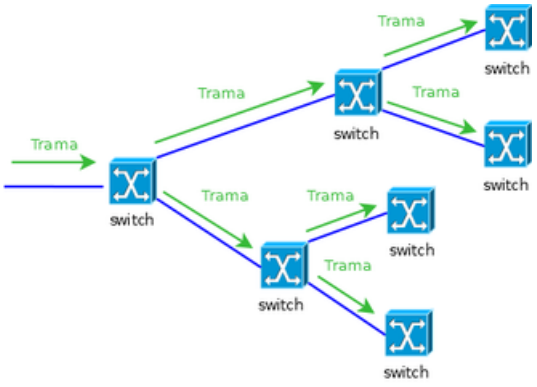

Imagen extraída de: Wikipedia, s.f

Otras ventajas que presenta este algoritmo es que no necesita cálculos complejos para obtener la ruta a seguir por el paquete por lo que la complejidad de la aplicación podría verse reducida, de igual manera, en cuanto a espacio de almacenamiento, tiene cierta eficiencia ya que los nodos conectados en la red no guardan en ningún lado la información del enlace. (Wikipedia, s.f)

- **Linkstate**

  El algoritmo de Linkstate es un algoritmo con el que todos los nodos o routers comparten la información sobre toda la topología. El algoritmo se basa en que todos los routers envían su tabla de enrutamiento hacía sus vecinos. Esta tabla de enrutamiento con la mejor ruta hacia los demás nodos es porque utiliza el algoritmo de Dijkstra del camino más corto, y luego de encontrar el mejor camino para todos los nodos, esta tabla se comparte entre estos. (Gaurav, 2022)

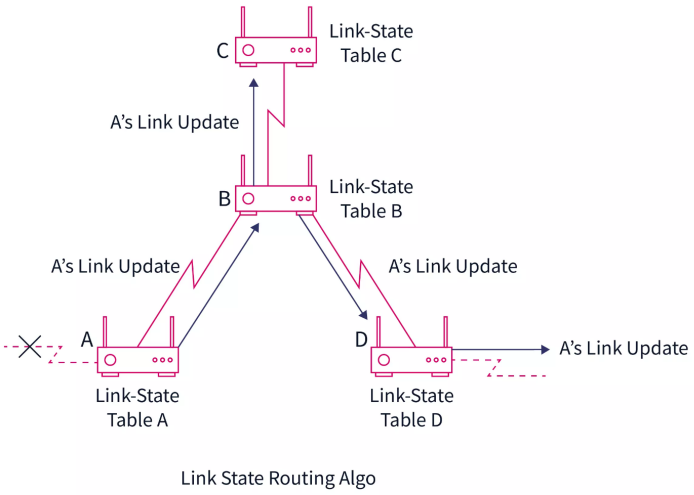

Imagen extraída de: Scaler Topics, 2022

La ventaja de este algoritmo es que es muy eficiente, por que en todas las tablas de enrutamiento de todos los nodos están sincronizadas, pero esto requiere de bastante bastante memoria, ya que se deben almacenar todas estas tablas para el correcto funcionamiento del algoritmo. (Gaurav, 2022)

- **Distance Vector Routing**

  El algoritmo Distance Vector Routing (DVR) es usado en los protocolos de enrutamiento de manera que los nodos conocen la red en forma de vectores, es decir que conocen la distancia y la dirección para llegar a otro nodo, sin conocer la topología completa. La distancia se da en términos de una métrica que representa el peso entre cada par de nodos y la dirección es uno de los nodos vecinos (conexión directa). (Cisco Press, 2001)

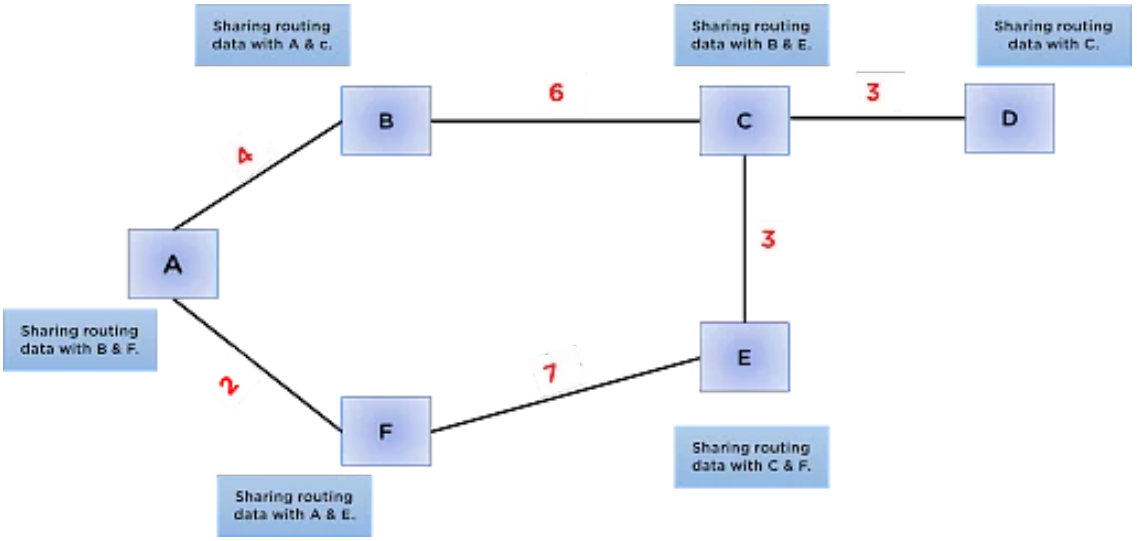

Imagen extraída de: simplilearn, 2022.

El DVR se caracteriza por utilizar el algoritmo Bellman-Ford para calcular la distancia más corta a cada nodo de la red por medio de las tablas que son compartidas periódica y únicamente entre vecinos. Bellman-Ford es un algoritmo más costoso que Dijkstra pero más simple que puede ser implementado en redes distribuidas y que necesitan trabajar pesos (incluso negativos) (Geeks4Geeks, 2022). Este también es un proceso iterativo ya que cada nodo envía su información en intervalos de tiempo definidos y es asíncrono ya que no requiere de un orden de ejecución, pero sí de varias iteraciones para que el broadcast se disperse en la red con la información actualizada de cada nodo. (JavaTpoint, s.f)

Resultados

Flooding

Para el análisis de los algoritmos, se realizaron pruebas con ciertas topologías y pruebas enviando mensajes desde distintos nodos para observar el comportamiento. En primer lugar para el algoritmo flooding se utilizaron 2 topologías distintas y se enviaron 2 mensajes en cada una de ellas para prueba.

Universidad del Valle de Guatemala
CC3067 Redes
Catedrático: Jorge Yass
Ciclo 2 de 2022

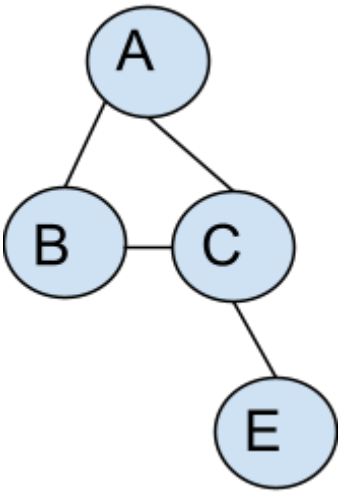

Imagen 1. Topología 1 para flooding.

Para esta primera topología se envió un mensaje desde el nodo A hacia el nodo E. Y el segundo mensaje se realizó desde el nodo B hacia el nodo E.

```
Bienvenido al programa
1. Iniciar sesion
2. Registrarse
3. Salir
Ingrese una opcion: 1
Ingresa tu username (sin @alumchat.fun): test3
Ingresa tu password:

Choose an option:
1 - Mandar un mensaje
2 - Salir
Option:

PS C:\Users\Bryann\Desktop\RedesLab3\flooding> py mainf.py -q
Bienvenido al programa
1. Iniciar sesion
2. Registrarse
3. Salir
Ingrese una opcion: 1
Ingresa tu username (sin @alumchat.fun): testcomp
Ingresa tu password:

Choose an option:
1 - Mandar un mensaje
2 - Salir
Option:
```

```
Bienvenido al programa
1. Iniciar sesion
2. Registrarse
3. Salir
Ingrese una opcion: 1
Ingresa tu username (sin @alumchat.fun): pacotest
Ingresa tu password:

Choose an option:
1 - Mandar un mensaje
2 - Salir
Option:
```

```
2. Registrarse
3. Salir
Ingrese una opcion: 1
Ingresa tu username (sin @alumchat.fun): testflood
Ingresa tu password:
WARNING  jid property deprecated. Use boundjid.bare
INFO     JID set to: testflood@alumchat.fun/6k9c7cdb56

Choose an option:
1 - Mandar un mensaje
2 - Salir
Option:
```

Imagen 2. Inicialización de nodos

```
Option: 1
Ingrese el usuario al que desea enviar el mensaje (sin @alumchat.fun):
Usuario: testflood
Ingrese el mensaje que desea enviar
Mensaje: hola desde A
Getting key:  A
send message to nodes:  ['B', 'C']

Choose an option:
1 - Mandar un mensaje
2 - Salir
Option:
```

```
2 - Salir
Option:
Mensaje recibido de:  test3@alumchat.fun
Mensaje para:  testflood@alumchat.fun
Mensaje:  hola desde A
Saltos:  1
Distancia:  1
Nodos:  ['A']


Getting key:  B
send message to:  ['A', 'C']
```

```
Option:
Mensaje recibido de:  test3@alumchat.fun
Mensaje para:  testflood@alumchat.fun
Mensaje:  hola desde A
Saltos:  1
Distancia:  1
Nodos:  ['A']


Getting key:  C
send message to:  ['A', 'E', 'B']
```

```
Option: WARNING  jid property deprecated. Use boundjid.bare
LLEGUE A MI DESTINO

Mensaje recibido de:  test3@alumchat.fun
Mensaje para:  testflood@alumchat.fun
Mensaje:  hola desde A
Saltos:  2
Distancia:  2
Nodos:  ['A', 'C']
```

Imagen 3. Mensaje de A hacia E

Imagen 4. Mensaje de B hacia E

Para la segunda topología se enviaron mensajes del nodo C hacia el nodo B y del nodo B hacia el nodo E

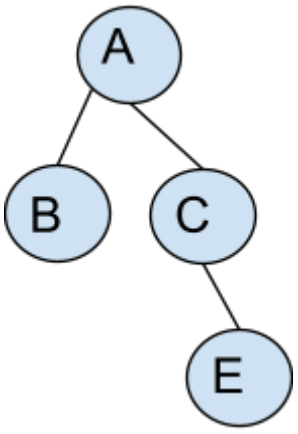

Imagen 5. Topología 2 para flooding

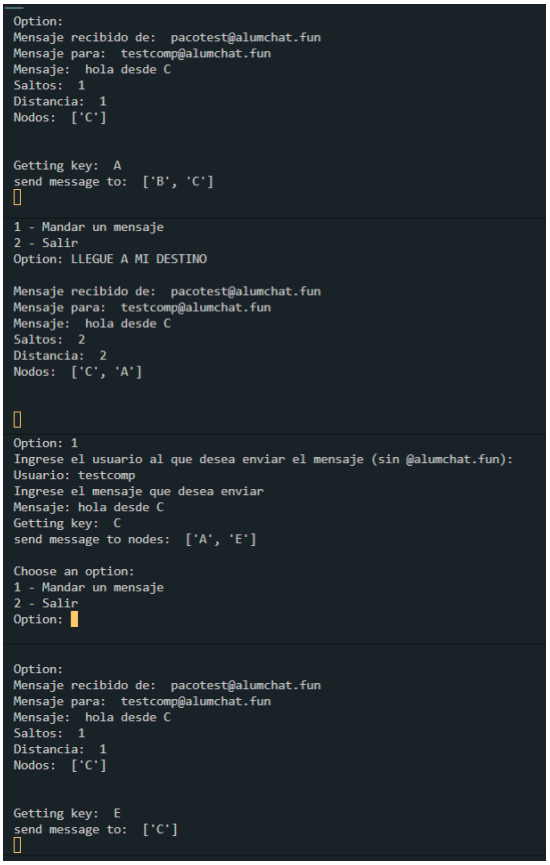

Imagen 6. Mensaje de C hacia B

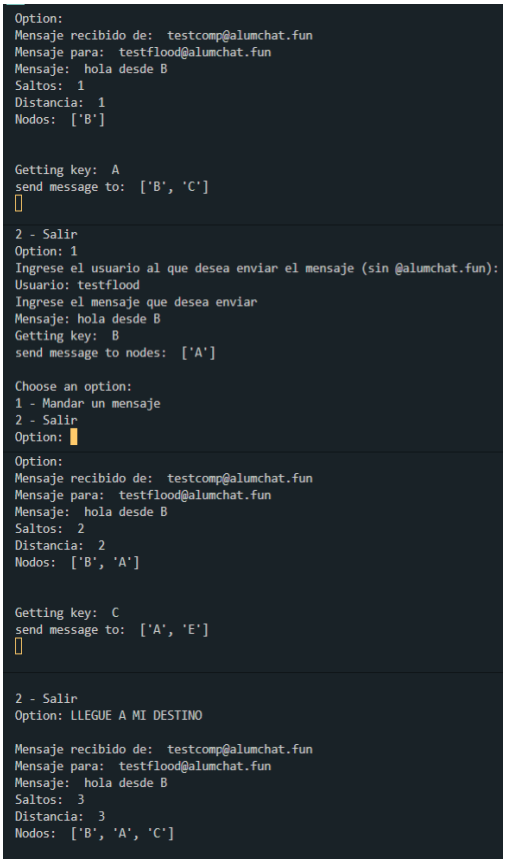

Imagen 7. Mensaje de B hacia E

Linkstate

Para el análisis del algoritmo Linkstate se realizaron pruebas con dos topologías, y de la misma manera que con el algoritmo de Flooding, se enviaron 2 mensajes en cada una.

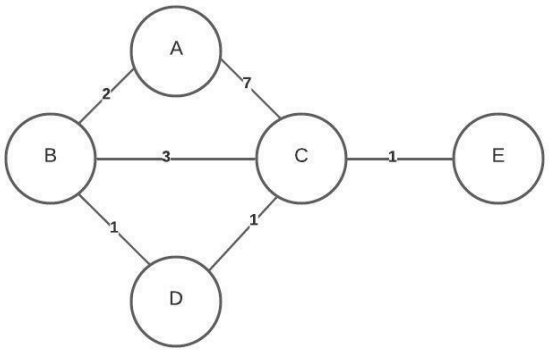

Imagen 8. Topología 1 para Linkstate.

Se enviarán 2 mensajes, uno de A a D y uno de B a E.

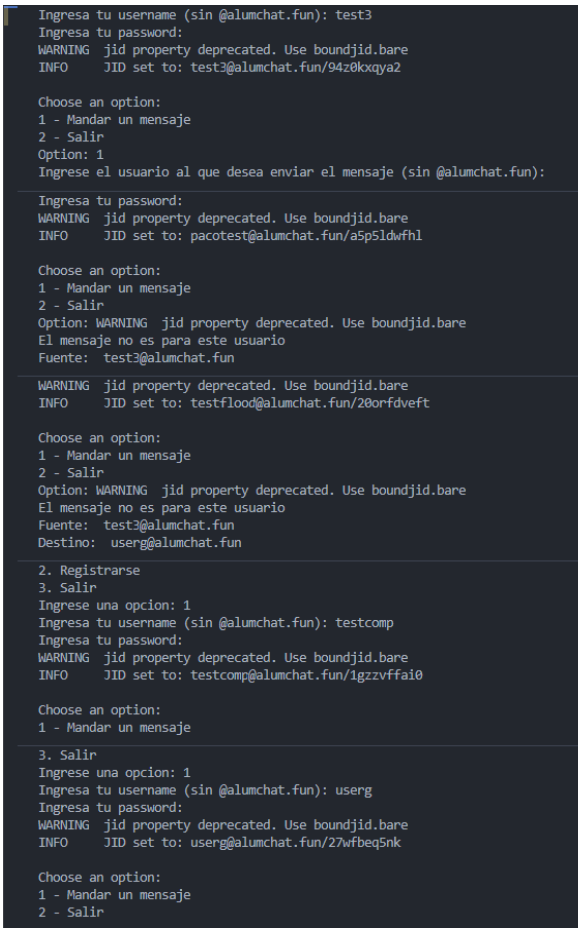

Imagen 9. Inicialización de nodos

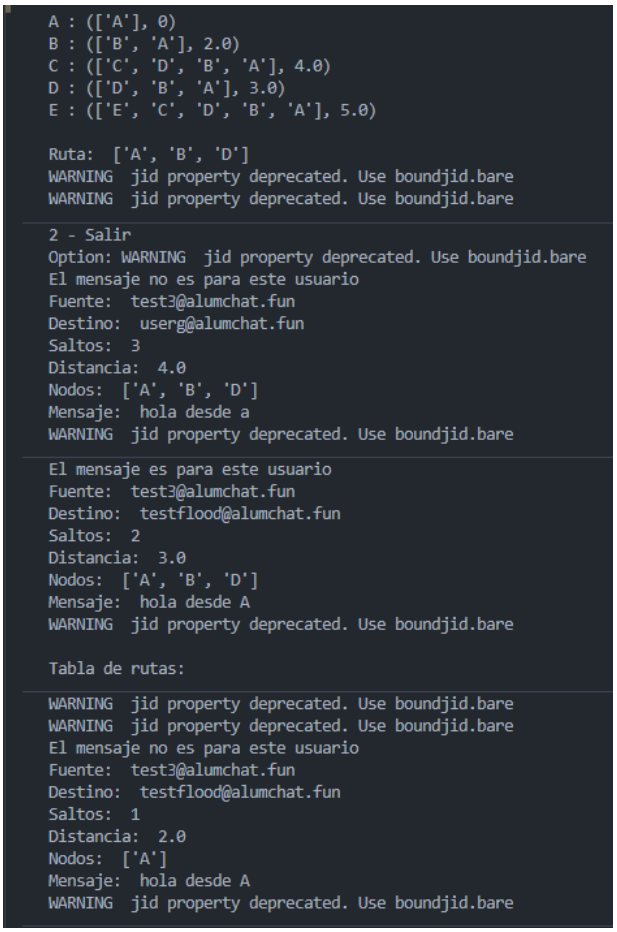

Imagen 10. Mensaje de A hacia D

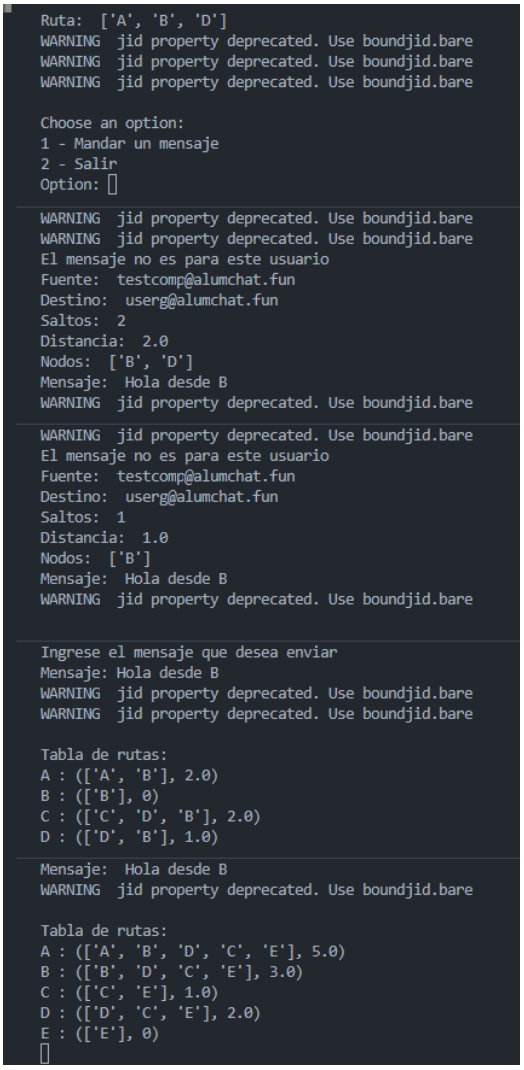

Imagen 11. Mensaje de B hacia E

Para la segunda prueba, se utilizará la siguiente topología:

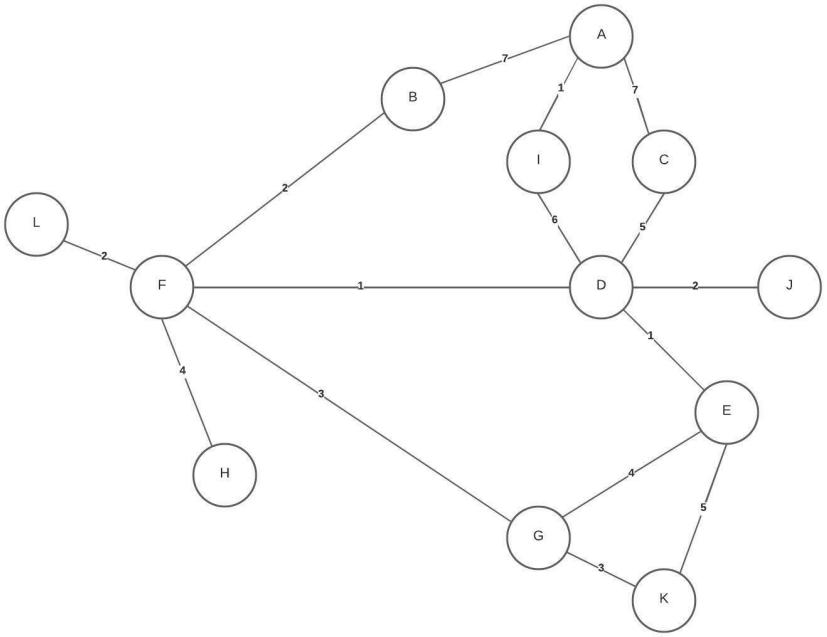

Imagen 12. Topología 2 para Linkstate.

Se enviarán 2 mensajes, uno de A a G y uno de A a H.

```
Topologia 1
A B 2
B C 3
B D 1
D C 1
C E 1
```

```
Topologia 2
A B 7
A I 1
A C 7
B F 2
C D 5
I D 6
D F 1
D E 1
F H 4
F G 3
E G 4
F L 2
D J 2
E K 5
G K 3
```

Imagen 13. Peso para las aristas de los nodos del Algoritmo Linkstate.

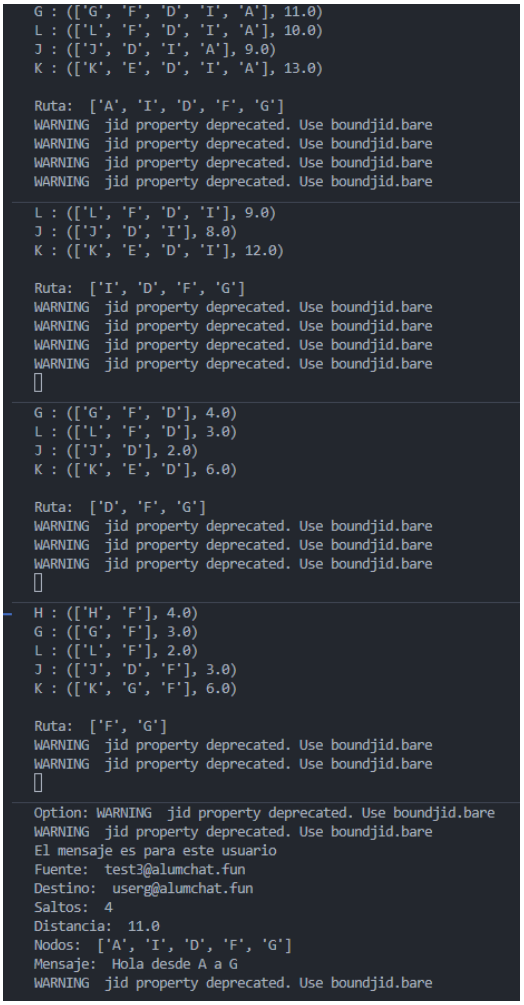

Imagen 14. Mensaje de A hacia G

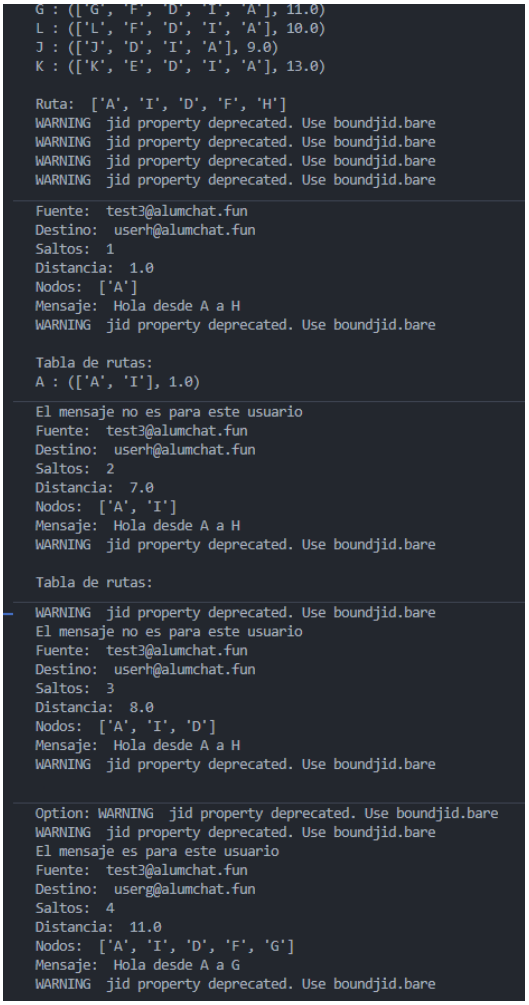

Imagen 15. Mensaje de A hacia H

Distance Vector Routing

Para este protocolo se realizaron pruebas con 2 topologías, una con pesos de valor 1 entre todos los nodos, y otra que aplica pesos distintos entre los nodos. Para cada una de ellas se ejecuta una instancia del programa individual para cada nodo de la red.

Para ejecutar un nodo se especifica el identificador de dicho nodo, es decir el nodo **A**, el nodo **B**, el nodo **C**, etc. Al momento de conectarse, se crea la tabla inicial con los nodos vecinos y empieza a compartir con estos, dicha tabla en un loop y a recibir las tablas de los otros, actualizando la tabla actual.

Las opciones del menú permiten ver la tabla actual, ver los vecinos, enviar un mensaje a un nodo y desconectarse.

```
PS D:\Projects\Redes-Lab3\DVR> py .\main.py
Ingrese el identificador del nodo (ej. A, B, C, etc): A
Nodo conectado

1. Ver tabla
2. ver vecinos
3. enviar mensaje
4. salir

Ingrese una opcion:
```

Imagen 16: Menú de opciones DVR.

La primera topología usada es la siguiente:

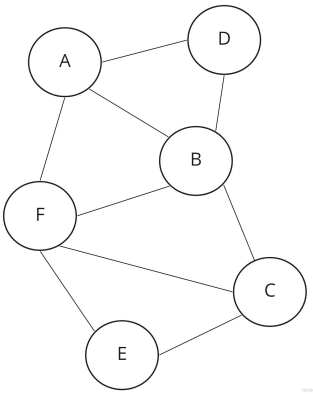

Imagen 17: Topología 1 para DVR.

Primero ejecutamos el nodo A y revisamos su tabla inicial, como podemos ver en la Imagen 18, este solo tiene a sus vecinos.

```
Ingrese una opcion: 1

group5_dvr_B - Peso acumulado: 1 - Salto Para llegar a él: group5_dvr_B
group5_dvr_D - Peso acumulado: 1 - Salto Para llegar a él: group5_dvr_D
group5_dvr_F - Peso acumulado: 1 - Salto Para llegar a él: group5_dvr_F
```

Imagen 18: Tabla inicial del nodo A para topología 1 DVR.

Luego ejecutamos el nodo B y para que comparta su tabla con A, si volvemos a ver la tabla de A, esta ya está actualizada con conocimiento del nodo C por medio del nodo B.

```
Ingrese una opcion: 1

group5_dvr_B - Peso acumulado: 1 - Salto Para llegar a él: group5_dvr_B
group5_dvr_D - Peso acumulado: 1 - Salto Para llegar a él: group5_dvr_D
group5_dvr_F - Peso acumulado: 1 - Salto Para llegar a él: group5_dvr_F
group5_dvr_C - Peso acumulado: 2 - Salto Para llegar a él: group5_dvr_b
```

Imagen 19: Tabla actualizada del nodo A para topología 1 DVR.

Luego de conectar todos los nodos, vemos que A tiene su tabla completa con las distancias más cortas a cada nodo y el vecino al que se tiene que dirigir para llegar a ellos.

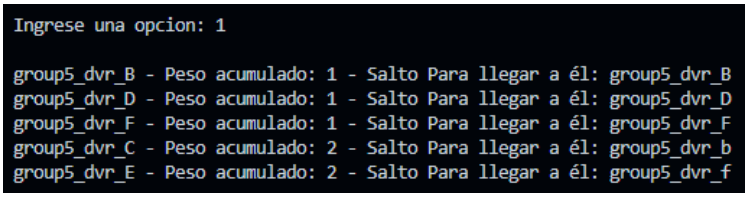

Imagen 20: Tabla completa del nodo A para topología 1 DVR.

Ahora si intentamos enviar un mensaje del nodo A al nodo E, sabemos bien que tiene que pasar por el nodo F, por lo tanto hacemos la prueba, especificando sólo el identificador del nodo al que va el mensaje y dicho mensaje.

Si revisamos el nodo F, este muestra una notificación de que está reenviando un mensaje desde A para E, así como los saltos que lleva, la distancia recorrida y los nodos por los que va pasando, en este caso, solo A.

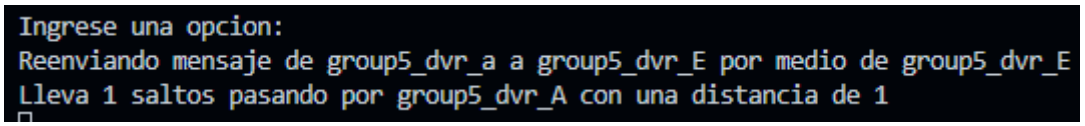

Imagen 21: Reenvío desde nodo F, topología 1 DVR.

Ahora revisando el nodo destino E, este muestra la notificación que el nodo A le envió un mensaje, mostrando el mensaje y el recorrido de este.

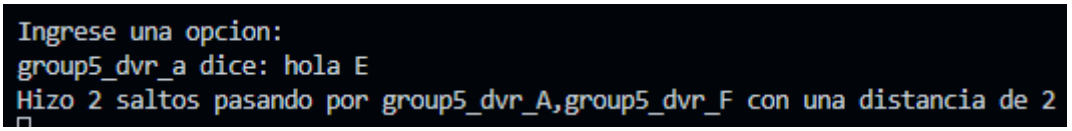

Imagen 22: Recorrido de mensaje de A para E, topología 1 DVR.

La segunda topología utiliza pesos distintos entre cada par de nodos:

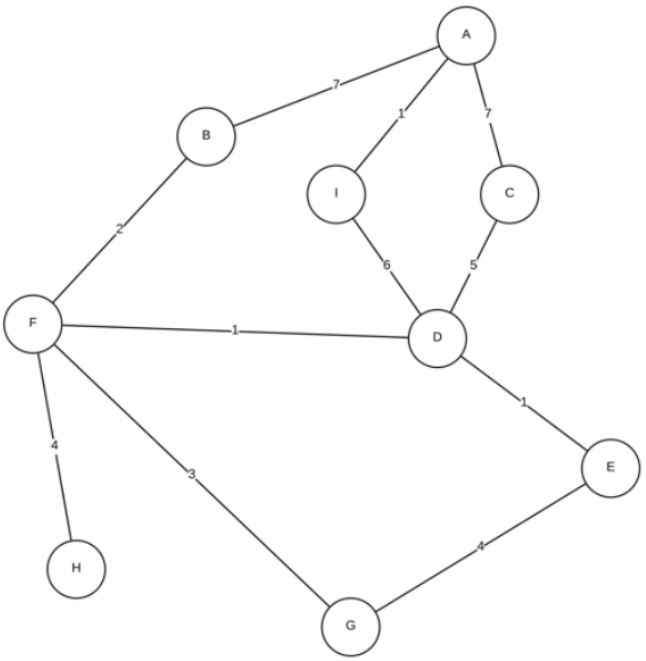

Imagen 23: Topología 2, con pesos para DVR.

De la misma manera que para la topología 1, ejecutamos el primer nodo y revisamos su tabla inicial.

```
Ingrese una opcion: 1

group5_dvr_B - Peso acumulado: 7 - Salto Para llegar a él: group5_dvr_B
group5_dvr_C - Peso acumulado: 7 - Salto Para llegar a él: group5_dvr_C
group5_dvr_I - Peso acumulado: 1 - Salto Para llegar a él: group5_dvr_I
```

Imagen 24: Tabla inicial nodo A, topología 2 DVR.

Al ejecutar los nodos B, F, I y C, vemos cómo se actualiza la tabla con la mejor ruta para llegar a D y el nodo al cuál hay que saltar para cumplir con ello, que es por medio del nodo I, con un peso de 7 (1 para el nodo I + 6 para el nodo D).

```
Ingrese una opcion: 1

group5_dvr_B - Peso acumulado: 7 - Salto Para llegar a él: group5_dvr_B
group5_dvr_C - Peso acumulado: 7 - Salto Para llegar a él: group5_dvr_C
group5_dvr_I - Peso acumulado: 1 - Salto Para llegar a él: group5_dvr_I
group5_dvr_F - Peso acumulado: 9 - Salto Para llegar a él: group5_dvr_b
group5_dvr_D - Peso acumulado: 7 - Salto Para llegar a él: group5_dvr_i
group5_dvr_G - Peso acumulado: 12 - Salto Para llegar a él: group5_dvr_b
group5_dvr_H - Peso acumulado: 13 - Salto Para llegar a él: group5_dvr_b
```

Imagen 25: Tabla actualizada nodo A, topología 2 DVR.

Conectando el resto de los nodos, vemos como la tabla se actualiza poco a poco cambiando la distancia y nodo de salto para algunos, como el nodo F, el nodo G y el nodo H.

```
Ingrese una opcion: 1

group5_dvr_B - Peso acumulado: 7 - Salto Para llegar a él: group5_dvr_B
group5_dvr_C - Peso acumulado: 7 - Salto Para llegar a él: group5_dvr_C
group5_dvr_I - Peso acumulado: 1 - Salto Para llegar a él: group5_dvr_I
group5_dvr_F - Peso acumulado: 8 - Salto Para llegar a él: group5_dvr_i
group5_dvr_D - Peso acumulado: 7 - Salto Para llegar a él: group5_dvr_i
group5_dvr_G - Peso acumulado: 11 - Salto Para llegar a él: group5_dvr_i
group5_dvr_H - Peso acumulado: 12 - Salto Para llegar a él: group5_dvr_i
group5_dvr_E - Peso acumulado: 8 - Salto Para llegar a él: group5_dvr_i
```

Imagen 26: Tabla final nodo A, topología 2 DVR.

Ahora con la prueba de envío de mensajes, de igual manera que con la topología 1, se van mostrando las notificaciones entre nodos que reenvían los mensajes así como el recorrido final al llegar al destino. Enviando un mensaje del nodo A al nodo E, este es su recorrido.

```
Ingrese una opcion: 3
Ingrese el id del nodo destinatario: E
Ingrese el mensaje: hola de A para E!!!
```

```
Ingrese una opcion:
Reenviando mensaje de group5_dvr_a a group5_dvr_E por medio de group5_dvr_D
Lleva 1 saltos pasando por group5_dvr_A con una distancia de 1
```

```
Ingrese una opcion:
Reenviando mensaje de group5_dvr_a a group5_dvr_E por medio de group5_dvr_E
Lleva 2 saltos pasando por group5_dvr_A,group5_dvr_I con una distancia de 7
```

```
Ingrese una opcion:
group5_dvr_a dice: hola de A para E!!!
Hizo 3 saltos pasando por group5_dvr_A,group5_dvr_I,group5_dvr_D con una distancia de 8
```

Imagen 27: Recorrido de mensaje desde A para E, topología 2 DVR.

Haciendo otra prueba con un mensaje desde el nodo H para el nodo C:

```
Ingrese una opcion: 3
Ingrese el id del nodo destinatario: C
Ingrese el mensaje: hola desde H para C!!!
```

```
Ingrese una opcion:
Reenviando mensaje de group5_dvr_h a group5_dvr_C por medio de group5_dvr_D
Lleva 1 saltos pasando por group5_dvr_H con una distancia de 4
```

```
Reenviando mensaje de group5_dvr_h a group5_dvr_C por medio de group5_dvr_C
Lleva 2 saltos pasando por group5_dvr_H,group5_dvr_F con una distancia de 5
```

```
Ingrese una opcion:
group5_dvr_h dice: hola desde H para C!!!
Hizo 3 saltos pasando por group5_dvr_H,group5_dvr_F,group5_dvr_D con una distancia de 10
```

Imagen 28: Recorrido de mensaje desde H para C, topología 2 DVR.

Discusión

Al momento de llevar a cabo la ejecución de los distintos algoritmos, se pudo comprender de mejor manera la forma en que funcionan los algoritmos de enrutamiento y la importancia que estos tienen al momento de enviar paquetes de información dentro de la red, ya que se busca que estos lleguen de la mejor manera posible y además en el menor tiempo posible determinando la mejor ruta.

Como primer algoritmo se utilizó flooding, el cual se basa en reenviar el mismo paquete hacia todos los nodos vecinos de salida. Como se puede observar en la imagen 1, la cual fue la primer topología utilizada para pruebas se tiene una conexión entre el nodo B y C lo cual podría provocar un loop y se debe hacer cierta evaluación para evitar este comportamiento.

La primer prueba fue enviar un mensaje desde el nodo A (test3@alumchat.fun) hacia el nodo E (testflood@alumchat.fun), como se puede observar en la imagen 2 se inicializan los nodos en distintas terminales y en la imagen 3 se observa el nodo A que envia el mensaje “hola desde A” el cual se lo envía al nodo B y C y se puede ver en el despliegue del mensaje , en donde los nodos recorridos se encuentra solamente “A” . En la última consola se observa que el mensaje llegó a su destino y los nodos recorridos fueron A y C con 2 saltos, lo cual es un comportamiento adecuado.

Para la segunda prueba de la primera topología se envió un mensaje del nodo B hacia el nodo E , de igual manera, en la imagen 4 se puede observar que B se lo envío al nodo A y C y en la consola de E se despliega el mensaje “Llegué a mi destino” y pasó por los nodos B y C con 2 saltos, lo cual nuevamente es un comportamiento correcto.

Por último, para la segunda topología, como se observa en la imagen 5, presenta un comportamiento lineal , sin embargo, hay un tope en el nodo B por lo que podría servir para hacer pruebas.

En este caso, se envió un mensaje del nodo C hacia el nodo B y como se observa en la imagen 6, el nodo C le envía el mensaje al nodo A y al E y el nodo A se lo envía al nodo B y la ruta queda que pasó por los nodos C y A.

En la última prueba, se envió un mensaje del nodo B al E, y se observa que B se lo envía al nodo A y este al nodo C y llega finalmente al nodo E con 3 saltos pasando por B, A , C.

Como segundo algoritmo se utilizó el algoritmo de Linkstate, el cual se basa es que todos los nodos tienen toda la información de la topología para tener la mejor ruta posible a los otros nodos. Se puede ver en la imagen 8 y 12, que las topologías tienen ciertas distribuciones para poder evaluar bien el algoritmo.

Para la primera prueba, se envió un mensaje desde el nodo A, al nodo D. Como se observa en la imagen 10, la ruta más corta para enviar este mensaje es a través de los nodos A, B y D, lo cual da un peso de 3, lo cual es la ruta más corta.

Luego, la segunda prueba fue enviar un mensaje desde el nodo B al nodo E. Para este envío, como se observa en la imagen 11, el mensaje se envió a través de 4 nodos, los cuales fueron los nodos B, C, D y E. Las aristas para esta ruta suman un peso de 3, lo cual indica que es la ruta más corta para este envío.

Seguidamente para la segunda topología, se puede observar en la imagen 14 la primera prueba, en la cual se envió un mensaje desde el nodo A, al nodo G. Como se muestra en dicha imagen, el mensaje se envió por la ruta más corta, la cual es ir por los nodos A, I, D, F y G, con un peso total de 11.

Para la siguiente prueba, se envió un mensaje desde el nodo A, al nodo H, como se observa en la imagen 15, los nodos que que se requieren para el envío de este mensaje son el A, I, D, F, H, con un peso total de 12, la ruta es parecida para el nodo G, pero esto debido a que la arista que conecta el nodo A y el nodo B, tiene un peso de 7, haciendo que esta arista no se encuentre en la tabla de rutas para los nodos de la topología.

El tercer algoritmo implementado fue Distance Vector Routing, el cual recibe como input la topología y crea cada nodo solamente con el conocimiento de sus vecinos, luego con el uso de hilos se implementó el *loop* para enviar constantemente la tabla a los vecinos. El algoritmo mostró resultados satisfactorios para las dos topologías implementadas las cuales tienen pesos positivos. Para este algoritmo se necesita que todos los nodos estén conectados para obtener la mejor ruta a cada uno y la frecuencia de actualización es un valor importante para mantener las tablas actualizadas lo más rápido posible pero sin saturar la red de estos mensajes de actualización.

Como se muestra en la Imagen 27 e Imagen 28 la ruta que toma cada mensaje se da por medio de los saltos a los vecinos, lo cual permite que la información de cada nodo individual sea muy simple. Como recomendación de mejora para este algoritmo se puede implementar la detección de desconexión de nodos, ya que actualmente si un nodo se desconecta las tablas lo siguen considerando como opción, esto puede llegar a bloquear la red si no se maneja.

Comentario grupal

En general, esta práctica ayudó a comprender de mejor manera el paso de mensajes y lo complejo que puede resultar el cálculo de la ruta más óptima en un ambiente real de red.

De igual manera, se pudo entender la importancia de llevar un orden y control en los nodos que están conectados para poder detectar cualquier error en el proceso de enviar mensajes y saber por los nodos que pasó dicho paquete.

Conclusiones:

1. Tomando en cuenta el funcionamiento de Flooding, se puede concluir que a pesar de ser confiable, puede llegar a ser muy redundante al momento de enviar paquetes si

no se configura adecuadamente y puede saturar la red representando un problema.

2. Debido a los resultados obtenidos con el algoritmo de Linkstate, se concluye que el algoritmo es eficiente por la manera en que aplica el algoritmo de Dijkstra, pero que requiere bastante memoria para que toda la topología tenga toda la información de esta misma.
3. A partir de la implementación de Distance Vector Routing se concluye que cada nodo es bastante simple pero el costo de cálculo y congestión de la red depende mucho del intervalo de envío constante de la tabla a los nodos vecinos.

Literatura citada:

Cisco Press. (2001). Dynamic Routing Protocols. sample chapter from CCIE: Routing TCP/IP Volume I. Extraído de: https://www.ciscopress.com/articles/article.asp?p=24090&seqNum=3

Wikipedia. (s.f). Inundación de red. Extraído de: https://es.wikipedia.org/wiki/Inundaci%C3%B3n_de_red

Gaurav S., (2022). Link State Routing Algorithm. Extraído de: https://www.scaler.com/topics/link-state-routing-algorithm/

Geeks4Geeks. (2022). Bellman–Ford Algorithm | DP-23. Extraído de: https://www.geeksforgeeks.org/bellman-ford-algorithm-dp-23/

JavaTPoint. (s.f.). Distance Vector Routing Algorithm. Extraído de: https://www.javatpoint.com/distance-vector-routing-algorithm