Universidad del Valle de Guatemala 01/09/2022
Redes Sección 20
Walter Danilo Saldaña Salguero 19897
Jose Abraham Gutierrez Corado 19111
Javier Alejandro Cotto Argueta 19324

LAB 3:
Algoritmos de Enrutamiento

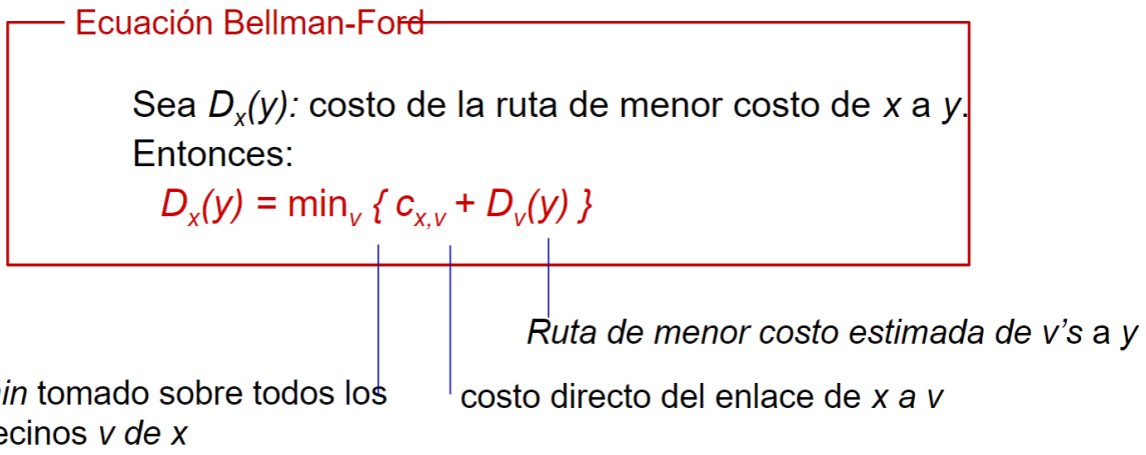
Descripción de la práctica

La práctica consiste en desarrollar una implementación de algoritmos de enrutamiento, con ayuda del chat desarrollado en el proyecto 1. Como objetivos de aprendizaje se reconoce el; Entender y conocer los algoritmos de enrutamiento que se utilizan actualmente en internet y saber el cómo funcionan las tablas de enrutamiento.

Los algoritmos de enrutamiento utilizados serán:

- Flooding
  Es un algoritmo similar a hacer un *broadcast* en una topología de red solo que se trata de enviar un mensaje/paquete por cada nodo hasta llegar al final. Cabe recalcar que se tienen que agregar ciertas decisiones para que el algoritmo no se quede trabado en un ciclo de transmisión-recepción pues cuando llega al tope, puede que este mande de nuevo el paquete al nodo que lo envió primeramente.

- Distance Vector Routing
  Primordialmente, este es un protocolo de cambio de rutas, no constante que informa al router sus vecinos directos e indirectos en la topología. Conocido históricamente como el algoritmo de ARPANET. Este algoritmo se basa en la tabla de enrutamiento de Bellman Ford. Esta tabla de enrutamiento se establece para cada nodo, y solo se sabrán los pesos de sus vecinos directos, mientras que los demas vecinos de la topología se irán calculando, conforme se necesite enviar un mensaje hasta un punto donde cada nodo conozca las rutas y pesos hacia todos los nodos. Se apoya de esta ecuación para determinar las rutas:



- Link State Routing
  Este algoritmo se utiliza en el caso en que routers conocen las tablas de enrutamiento usando la misma lógica que Dijkstra, el cual es encontrar la ruta más corta entre dos nodos. En este caso como los pesos son 1, entonces lo que será la ruta corta se basará en el número de "saltos" posibles.

Resultados

topologia.txt

```
{
"type":"topo",
"config":{
    "A": ["B", "D", "F"],
    "B": ["A", "C", "D", "F"],
    "C": ["B", "E", "F"],
    "D": ["A", "B"],
    "E": ["C", "F"],
    "F": ["A", "B", "C", "E"]}
}
```

Imagen 1: Topología para pruebas

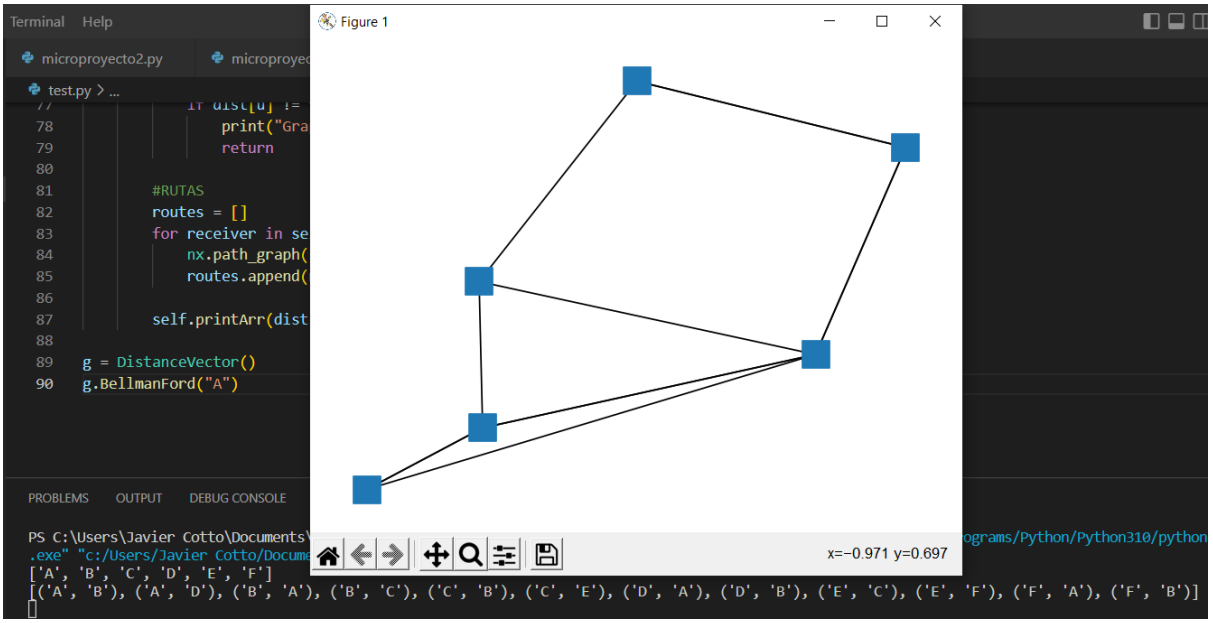


Imagen 2: Grafo mostrato a partir de la topología de prueba

Distance Vector

VERTICES	PESO	RUTA
A	0	['A']
B	1	['A', 'B']
C	2	['A', 'B', 'C']
D	1	['A', 'D']
E	3	['A', 'B', 'C', 'E']
F	4	['A', 'B', 'C', 'E', 'F']

Imagen 3: Tabla de enrutamiento para el nodo A, con sus respectivas rutas más cortas hacia todos los nodos.

Flooding

```
---------------------------
|     MENU DE OPCIONES     |
---------------------------
|1. Iniciar Sesión         |
|2. Salir                  |
---------------------------
Ingrese opción: 1
Ingrese usuario(usuario@alumchat.fun): walt.gfe@alumchat.fun
Ingrese contraseña:
 1) Flooding
 2) Distance Vector
 3) Dijkstra
Ingrese algoritmo: 1
['A', 'B', 'C', 'D', 'E', 'F']
[('A', 'B'), ('A', 'D'), ('B', 'A'), ('B', 'C'), ('C', 'B'), ('C', 'E'), ('D', 'A'), ('D', 'B'), ('E', 'C'), ('E', 'F'), ('F', 'A'), ('F', 'B')]
---------------------------
|     MENU DE FUNCIONES     |
---------------------------
|1.  Mostrar Contactos      |
|2.  Agregar Contacto       |
|3.  Detalles Contacto      |
|4.  Mensaje Directo        |
|5.  Conversación Grupal    |
|6.  Mensaje de Presencia   |
|7.  Archivos               |
|8.  Eliminar               |
|9.  Salir                  |
|10. Obtener ruta           |
---------------------------
Ingrese opción: 10
Nodo inicial: A
Nodo receptor: B
{'A': ['B', 'D'], 'B': ['C'], 'D': [], 'C': ['E'], 'E': ['F'], 'F': []}
---------------------------
```

Imagen 4: Tabla de enrutamiento para el nodo A para enviar a todos los nodos con flooding.

Link State Routing

```
['A']  with number of steps -> 0
['A', 'B']  with number of steps -> 1
['A', 'C']  with number of steps -> 1
['A', 'D']  with number of steps -> 1
['A', 'B', 'F', 'E']  with number of steps -> 3
['A', 'B', 'F']  with number of steps -> 2
```

Imagen 5: Tabla de enrutamiento del Nodo A hacia otros nodos con la ruta más corta

Discusión

Para nuestro laboratorio se implementaron los algoritmos para calcular todas las rutas posibles, dependiendo del tipo de enrutamiento que seguiría el conjunto de routers donde cada uno sería representado por un usuario. Y por cada usuario se crearía una tabla de enrutamiento para saber cómo llegar a los otros nodos con sus posibles rutas (excepto en el caso de *Distance Vector* que cada vez que un usuario ingresara se deben actualizar las tablas).

Para *Flooding Routing* solo era necesario conocer los nodos vecinos y recorrerlos hasta que el mensaje haya pasado por todos los nodos. Este algoritmo no presentó mayor dificultad más que evitar un loop de envío de mensajes, en otras palabras, que el mensaje recorriera todo el grafo varias veces.

Para *Link State Routing* se mostraba las tablas de enrutamiento de cada nodo y los caminos más cortos hacia el resto. Esto se logró gracias al famoso algoritmo de Dijkstra el cual se encarga precisamente de esto: encontrar la ruta más corta entre nodos con diferentes pesos entre cada uno de ellos, ya sea en un grafo dirigido o no. Y dado a que existe una extensa lista de diferentes implementaciones de este, no fue mayor reto su implementación.

La convergencia entre nodos en *Distance Vector* fue un problema. Pues trabajamos con la topología completa en vez de que cada nodo enviará sus pesos estipulados. El problema surge a partir de que al buscar la implementación en internet para tener como referencia, todas estas contaban con la topología completa y no fue hasta el día de la presentación que nos dimos cuenta de ello. Entonces se estaba aplicando bellman-ford a la topología completa, en vez de que se calculará cada vez que un nuevo router (en este caso usuario) se uniera a la red.

Durante la elaboración del proyecto se nos presentaron dificultades con respecto al recibimiento y reenvío de mensajes, pues por alguna extraña razón que no sabemos la causa, estábamos recibiendo mensajes(echo) de otro cliente el cual nos mandaba mensajes en un formato que nosotros no teníamos estipulado/presente en el programa, por lo tanto el programa los leía como errores externos a este y a partir de ahí cortaba la conexión con el servidor y la cuenta. Se arregló utilizando otras cuentas y volviendo a hacer las revisiones en el programa.

Comentario Grupal

Una práctica interesante, se pudo ver en realidad cómo interactúa el internet con los dispositivos actualmente y como las propias tipologías tienen que estar propensas a errores para que no suceda ningún imprevisto. El haber implementado y actualizado las topologías de internet con el pasar del tiempo, tuvo que llevar a un uso de bastantes recursos, para lograr lo que se usa comúnmente hoy en día. Luego de esto como camino a futuro, se puede ver que las topologías seguramente se reducirán a cada nodo ubicado en la nube y ya no físico.

Conclusiones

- *Link State Routing* y *Distance Vector* son algoritmos similares en su funcionamiento, pues los dos calculan la ruta más corta. Donde difieren es que en LSR cada nodo sabe la topología completa y procede a calcular todas las rutas desde un inicio, mientras que DV calcula nuevas rutas cada vez que un router ingresa a la red o ya no es existente, actualizando las tablas de ruteo constantemente para mostrar los caminos más cortos entre nodos.
- *Flooding Routing* es el más sencillo de todos dado a que solo es pasar el mensaje por cada nodo presente en el grafo. El único reto que presenta es que no resulte en un ciclo infinito de envío de mensajes, sino que el algoritmo debe de saber cuando ya envió el mensaje a un nodo y no volverlo a hacer.
- Para la elaboración de una tarea similar, se debe de tener en cuenta los clientes externos a este, pues muchas veces los reenvíos de mensajes no se realizaban debido a la existencia de otra instancia (cliente externo como Gajim) al momento de la ejecución.