Universidad del Valle de Guatemala
Facultad de Ingeniería
Departamento de Ciencias de la Computación
Redes CC3067
Ing. Jorge Yass

Laboratorio # 3 (Parte 2)
Algoritmos de Enrutamiento

Christopher García 20541
José Rodrigo Barrera 20807
Maria Isabel Solano 20504
Semestre II
Septiembre 2023

Descripción de la práctica

La práctica realizada para el presente laboratorio consistió en la implementación de los algoritmos de enrutamiento antes practicados en la tercera parte en una simulación sobre el protocolo XMPP. Los algoritmos utilizados fueron Flooding, Linked State, y Distance Vector Routing. La implementación fue realizada en el lenguaje de programación Python, y basándose en la librería Slixmpp, la cual permitió la exitosa implementación del protocolo XMPP.

El programa consiste en un menú principal en el que se puede iniciar sesión, crear cuentas o salirse del programa. Las primeras 2 opciones se basan en los archivos de topología y los nombres de los nodos, por lo que solamente permiten iniciar sesión en nodos pertenecientes a la topología. Cuando crea o inicia sesión, este automáticamente asume que la contraseña es “redes2023”. Luego de iniciar sesión se hacen las configuraciones necesarias para que el programa funcione correctamente, tanto para los algoritmos como para la configuración de la conexión de Slixmpp. A partir de ello al usuario ya se le presentan las opciones de revisar a los contactos y el estado en el que se encuentran.

La segunda opción es enviar un mensaje. Este solicitará el algoritmo que se desea usar para enviar el mensaje y luego requerirá el nombre del nodo al cuál enviar el mensaje y cuál es el mensaje en sí. Y la última opción antes de salir del programa es ver la tabla de enrutamiento. del nodo actual, en donde la columna izquierda representa al nodo al que se quiere llegar, la columna de en medio, representa el peso neto y más corto encontrado por el momento para llegar a ese nodo, y la última columna es el nodo siguiente por el que se debe pasar para llegar al nodo destino.

Para poder re-configurarse y recibir mensajes, el cliente posee un manejador de eventos para cuando le llegan mensajes. Cuando esto ocurre, primero distingue a qué algoritmo le pertenece el mensaje y cómo lo debe tratar después.

Algoritmos utilizados

Flooding

En este programa, flooding se desempeña de la misma manera que en la parte 1 de este laboratorio con la excepción de que es todo “automático”. Cuando se escoge enviar un mensaje con este algoritmo se realiza la solicitud de dos cosas: El nodo destino y el mensaje. Dado que no conocemos el grafo completo sino que cada nodo/terminal conoce su entorno se comienza el “flooding” del mensaje. Luego de ingresar lo solicitado se crea el mensaje y se envía a todos los vecinos del nodo actual, si el nodo que recibe no es el nodo destino se procede a reenviar el mensaje a todos los vecinos de este nuevo nodo (se aplica una llamada recursiva de flooding para hacer este proceso hasta llegar al nodo destino). Para evitar

enviar/recibir el mensaje dos o más veces dentro del json que se envía existe una lista de nodos visitados. Al llegar al nodo destino se muestra el payload y se termina el algoritmo.

Linked State

En este programa se toma en consideración muchas cosas con respecto a su funcionamiento, primero, esto consiste en enviarle a cada uno de los nodos la info de la topología del mensaje que se quiere enviar, entonces, una vez se tenga la información enviada en cada uno de los nodos se procedería a calcular mediante el algoritmo de Link State Routing el camino más corto utilizando Dijkstra, esto considerando como tal la propia definición de lo que consiste dicho algoritmo, el cual, es un algoritmo de enrutamiento dinámico que utiliza el concepto de estado de enlace para construir una visión global de la topología de la red. Cada router en una red LSR mantiene una base de datos de estado de enlace que contiene información sobre todos los enlaces en la red. Esta información incluye el costo de cada enlace, el router vecino al que está conectado el enlace y el estado del enlace.

Y para que se envíe toda esa información a cada uno de los vecinos, se consideró utilizar qué, cuando es de tipo “info” esto significa que se debe de pasar la toda la topología a cada uno de los nodos antes de realizar el cálculo del camino más corto, una vez se realizó dicho cálculo, este tipo cambia a “message” lo cual significa que es el estado en donde se llegó al nodo destino para que este muestre el mensaje.

Distance Vector Routing

En este programa, distance vector routing tiene dos funciones muy importantes. La primera es generar y compartir las tablas de enrutamiento. Justo al inicializar el cliente, al momento de hacer las configuraciones necesarias, este genera su propia tabla de enrutamiento solamente con sus vecinos directos. Debido a que no se están trabajando pesos, la topología automáticamente le asigna a todos los vecinos un peso de 1. Luego de generar la tabla, la comparte con estos vecinos.

Cuando el programa detecta un mensaje perteneciente al algoritmo, este determina si es un paquete de tipo “mensaje” o de tipo “información”. En el primer caso, se revisa si el nodo actual es el nodo destino. Si es el caso, se imprime el mensaje para notificarle al usuario que le ha llegado un mensaje. Si es el segundo caso, se revisa la tabla de enrutamiento y se reenvía el mensaje por medio del salto para el nodo destino. Por el contrario, cuando se encuentra un mensaje de tipo “información”, este realiza el proceso necesario para actualizar la tabla de enrutamiento. Solamente si ha ocurrido un cambio gracias a esta actualización, se vuelve a compartir la tabla de enrutamiento a todos los vecinos, en el caso contrario, no es necesario compartir la tabla.

Resultados y Discusión de Resultados

Flooding

```
{'type': 'message', 'headers': {'algorithm': '
flooding', 'from': 'A', 'to': 'I', 'receivers'
: ['A', 'B', 'C']}, 'payload': 'Hola desde A '
}
```


Imagen #. Formato de paquete de tipo mensaje

```
Mensaje entrante de: A
Hola desde A
```

Imagen #. Impresión de payload

Linked State


```
{
    "type": "info",
    "headers": {
        "from": "A",
        "to": "H",
        "algorithm": "Link State Routing"
    },
    "payload": [
        [
            "A",
            0,
            "A"
        ],
        [
            "A",
            [
                "A"
            ],
            "A"
        ]
    ]
}
menu

_________________________

Ingrese el número de la opción que desea realizar:
```



```
{
    "type": "message",
    "headers": {
        "from": "A",
        "to": "H",
        "hop": " ",
        "algorithm": "Link State Routing"
    },
    "payload": "hola desde Link State Routing"
}

_____________________
```


```
3) Consultar tabla de enrutamiento

4) SALIR
No. de opción: 1
{'type': 'message', 'headers': {'from': 'A', 'to': 'H', 'hop': ' '
, 'algorithm': 'Link State Routing'}, 'payload': 'hola desde Link
State Routing'}

===================================
Nuevo mensaje de A!
hola desde Link State Routing
===================================
```

Distance Vector Routing


```
{'type': 'info', 'headers': {'from': 'C', 'to': 'G', 'algorith
m': 'DVR'}, 'payload': [['C', 0, 'C'], ['A', 1, 'A'], ['B', 1,
 'B'], ['E', 1, 'E'], ['G', 1, 'G'], ['H', 1, 'H'], ['F', 2, '
E'], ['D', 2, 'G']]}
```

Imagen #. Formato de paquete de tipo información

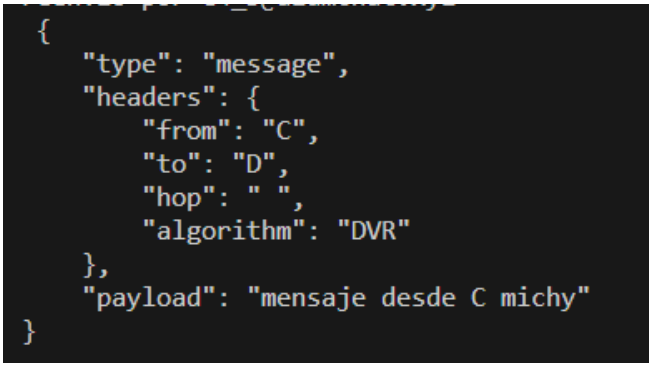

Imagen #. Formato de paquete de tipo mensaje

Conclusiones

- El objetivo de este laboratorio fue cumplido ya que se logró implementar exitosamente tres distintos algoritmos de enrutamiento con el protocolo XMPP.
- Los algoritmos de enrutamiento presentan diferentes características, ventajas y desventajas para el envío y recepción de mensajes, el desempeño de los mismos dependerá de las topologías, de la dimensión del grafo y también de la velocidad del servidor o routers.
- En cuanto a dificultad, flooding se podría decir que fue el más sencillo pero esto representó cierto tráfico ya que debía enviar el mensaje a todos los vecinos hasta llegar al nodo final, por otro lado tanto DVR y LSR son buenos algoritmos de enrutamiento pero dependerá del contexto para evaluar su desempeño completo.

Comentarios

- Usar python, puede facilitar muchas cosas, pero al mismo tiempo puede que traiga otros problemas. Tal como en el caso de este laboratorio el cual gracias a que el programa espera a que una persona haga un input del teclado para continuar, los mensajes no aparecen justo al momento de ser recibidos sino hasta que el usuario ingrese un input y lo siguiente del programa se imprima, haciendo que el programa no parezca automático.
- Durante el trabajo de laboratorio hubo varios errores con el servidor. Entre estos, muchas veces simplemente no permitía iniciar sesión para un nodo. Por lo que realizar pruebas fue difícil. Pero estos problemas fueron resueltos poco antes de la hora de entrega por lo que sí funcionó bien.

Link del Repositorio

https://github.com/MaIsabelSolano/UVG_Redes_Lab-3_P2