

UNIVERSIDAD DEL VALLE DE GUATEMALA

Redes

Sección 20



“Laboratorio 3 Parte 1”

ESTEBAN ALDANA GUERRA 20591

JESSICA PAMELA ORTIZ IXCOT 20192

GUATEMALA, Septiembre 2023

Descripción de la Práctica

Este laboratorio tiene como objetivo principal entender y aplicar los conceptos básicos relacionados con los algoritmos de enrutamiento del Internet. Dentro de este laboratorio se implementan los diferentes algoritmos haciendo uso de la herramienta de Python. Dentro de este ambiente se ejecutan y simulan los comportamientos de los algoritmos.

Los algoritmos implementados durante este laboratorio son Flooding, Distance Vector y Link State Routing.

El algoritmo de Flooding consiste en un algoritmo simple de enrutamiento en el cual se envían todos los paquetes entrantes por cada interfaz de salida, excepto por la que se ha recibido. Debido a como funciona el algoritmo de enrutamiento se garantiza que un paquete es entregado (si este puede ser entregado). Este algoritmo de enrutamiento es muy fácil de implementar, aunque con un enfoque bruto e ineficiente.

El algoritmo de Distance Vector se basa en que cada nodo intercambia tablas de enrutamiento con sus vecinos y actualiza sus tablas basándose en la información recibida. Se trata de uno de los más importantes junto con el de estado de enlace. Utiliza el algoritmo de Bellman-Ford para calcular las rutas.

Dentro del algoritmo de Link State cada nodo conoce el estado de sus propios enlaces y comparte esta información con todos los demás nodos en la red. Los protocolos link-state utilizan el algoritmo SPF (Shortest Path First) creado por Edsger Dijkstra, el cual calcula la mejor ruta en base al menor costo acumulado a lo largo de una ruta.

Implementación / Resultados

Para la implementación de los tres algoritmos se utilizó la herramienta de python mencionada anteriormente.

Flooding

Se adoptó un enfoque simulado para el algoritmo de flooding en lugar de una implementación directa. En esta simulación, cada instancia de la terminal se comporta como un nodo en la topología previamente definida (ver Imagen 1). Al ejecutar la simulación, se emula el funcionamiento del algoritmo. Por ejemplo, una de las funciones implica enviar mensajes. Si elegimos el nodo A y deseamos enviar un mensaje a D según la topología, el mensaje se

transmitirá a través de B, C y finalmente D. En las otras terminales, se simula la recepción del mensaje. Cada nodo recibe la información y la reenvía a sus nodos vecinos, continuando este proceso hasta que el destinatario final, en este caso D, reciba el mensaje. Si el Nodo A intenta enviar el mensaje nuevamente, se rechaza y no se permite su reenvío. Puedes visualizar el funcionamiento en el video adjunto. La dinámica de esta simulación se ilustra en la Imagen 2.

Imagen 1

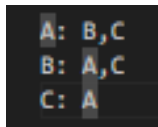
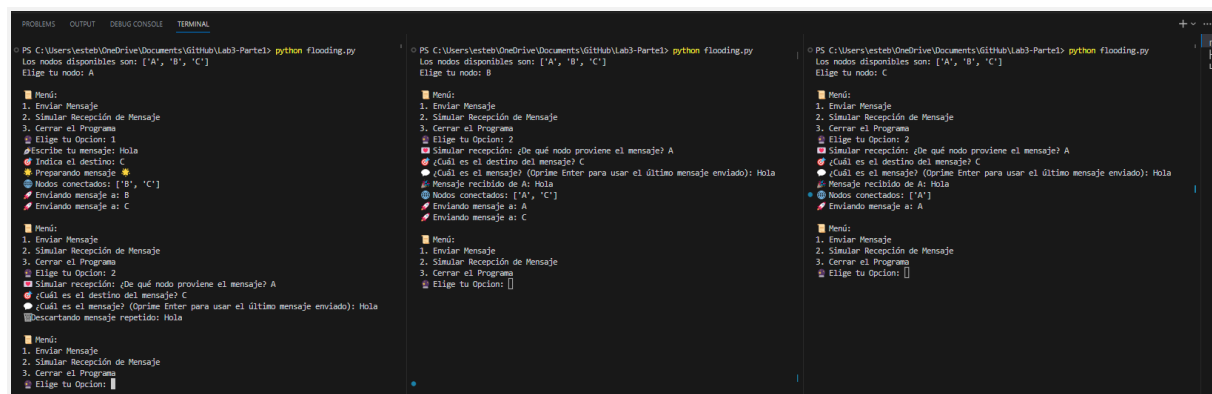


Imagen 2



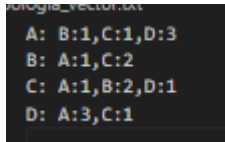
Distance Vector

Para la implementación de Distance Vector se siguió un enfoque similar al del algoritmo de flooding, optando por una simulación del proceso. En el caso de Distance Vector, se utilizó la topología proporcionada (ver Imagen 3), en la cual cada nodo está conectado a sus vecinos con distancias asociadas. Al iniciar el programa, el usuario debe seleccionar su propio nodo, donde cada nodo representa una instancia. Por ejemplo, si el usuario desea actualizar su tabla de enrutamiento, procede a enviar la tabla hacia los nodos vecinos. Estos nodos receptores, en esencia, reciben la tabla de enrutamiento del nodo emisor. Sin embargo, dado que aún no hay clientes implementados en esta fase del laboratorio, se emplea la topología inicial para simular la recepción de tablas de enrutamiento.

Un ejemplo de este escenario es cuando el usuario se encuentra en el nodo B y recibe la tabla de enrutamiento del nodo A. En este caso, el usuario selecciona la opción de recepción de tabla, lo que lo lleva a consultar la topología inicial en busca de la tabla de enrutamiento del

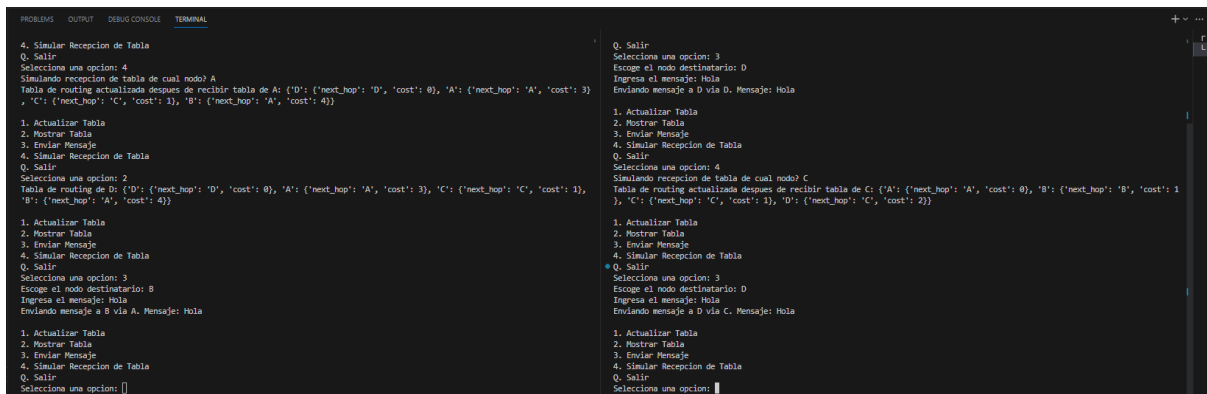
nodo A. Una vez que encuentra la tabla, aplica el algoritmo para actualizar su propia tabla y refrescar los valores iniciales. Este proceso se repite para los demás nodos en la tabla de enrutamiento. Para una comprensión más detallada, se recomienda consultar el video explicativo sobre el funcionamiento de Distance Vector. Además, el funcionamiento de este proceso se ilustra en la Imagen 4.

Imagen 3



```
logia_vector.txt
A: B:1,C:1,D:3
B: A:1,C:2
C: A:1,B:2,D:1
D: A:3,C:1
```

Imagen 4



```
PROBLEMAS  OUTPUT  SERVIDOR CONSOLE  TERMINAL

4. Simular Recepcion de Tabla
Q. Salir
Selecciona una opcion: 4
Simulando recepcion de tabla de cual nodo? A
Tabla de routing actualizada despues de recibir tabla de A: ('D': ('next_hop': 'D', 'cost': 0), 'A': ('next_hop': 'A', 'cost': 3), 'C': ('next_hop': 'C', 'cost': 1), 'B': ('next_hop': 'A', 'cost': 4))

1. Actualizar Tabla
2. Mostrar Tabla
3. Enviar Mensaje
4. Simular Recepcion de Tabla
Q. Salir
Selecciona una opcion: 2
Tabla de routing de D: ('D': ('next_hop': 'D', 'cost': 0), 'A': ('next_hop': 'A', 'cost': 3), 'C': ('next_hop': 'C', 'cost': 1), 'B': ('next_hop': 'A', 'cost': 4))

1. Actualizar Tabla
2. Mostrar Tabla
3. Enviar Mensaje
4. Simular Recepcion de Tabla
Q. Salir
Selecciona una opcion: 3
Escoge el nodo destinatario: B
Ingresa el mensaje: Hola
Enviando mensaje a B via A. Mensaje: Hola

1. Actualizar Tabla
2. Mostrar Tabla
3. Enviar Mensaje
4. Simular Recepcion de Tabla
Q. Salir
Selecciona una opcion: 1

Q. Salir
Selecciona una opcion: 3
Escoge el nodo destinatario: D
Ingresa el mensaje: Hola
Enviando mensaje a D via C. Mensaje: Hola

1. Actualizar Tabla
2. Mostrar Tabla
3. Enviar Mensaje
4. Simular Recepcion de Tabla
Q. Salir
Selecciona una opcion: 3
Escoge el nodo destinatario: D
Ingresa el mensaje: Hola
Enviando mensaje a D via C. Mensaje: Hola

1. Actualizar Tabla
2. Mostrar Tabla
3. Enviar Mensaje
4. Simular Recepcion de Tabla
Q. Salir
Selecciona una opcion: 1
```

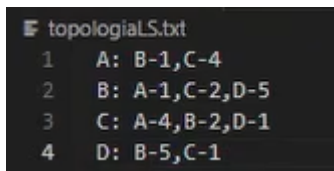
Link State

Para la implementación del algoritmo de Link-State, se optó por un modelo basado en la topología de la red proporcionada en un archivo de texto. En este archivo (ver imagen 5), cada nodo está descrito junto con sus vecinos y los costos asociados a cada enlace. Al iniciar el programa, se cargan estos datos para simular la topología completa de la red.

Al principio de la ejecución, el programa genera las tablas de enrutamiento para cada nodo utilizando el algoritmo de Dijkstra, y se pueden ver al darle a la opción 1 del menú (ver imagen 6). Estas tablas se almacenan en memoria y son accesibles durante toda la simulación. Esta fase simula el proceso en el que cada nodo calcula las rutas más cortas a todos los otros nodos basándose en la información global de la topología, simulando cómo cada tabla hace sus cálculos y se los comparten entre ellos para que todos tengan conocimiento de la topología completa.

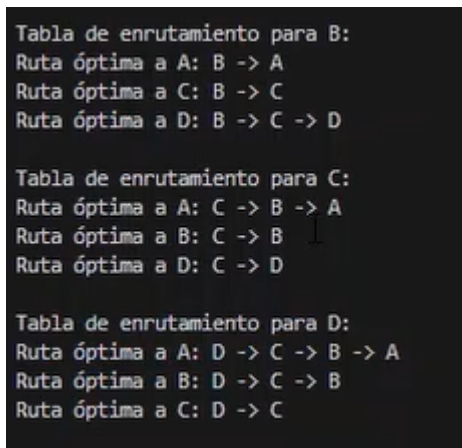
El usuario interactúa con el programa mediante un menú interactivo. A través de este menú, se pueden realizar varias acciones como visualizar las tablas de enrutamiento y simular el envío de un mensaje de un nodo a otro. Cuando se opta por enviar un mensaje, el programa busca en la tabla de enrutamiento del nodo de origen para encontrar la ruta más corta al nodo de destino. Luego, simula el envío del mensaje a lo largo de esta ruta óptima.

Imagen 5



```
topologiaLS.txt
1  A: B-1,C-4
2  B: A-1,C-2,D-5
3  C: A-4,B-2,D-1
4  D: B-5,C-1
```

Imagen 6



```
Tabla de enrutamiento para B:
Ruta óptima a A: B -> A
Ruta óptima a C: B -> C
Ruta óptima a D: B -> C -> D

Tabla de enrutamiento para C:
Ruta óptima a A: C -> B -> A
Ruta óptima a B: C -> B
Ruta óptima a D: C -> D

Tabla de enrutamiento para D:
Ruta óptima a A: D -> C -> B -> A
Ruta óptima a B: D -> C -> B
Ruta óptima a C: D -> C
```

Discusión

Flooding

Lo más destacado del algoritmo de Flooding es su simplicidad. El mecanismo para reenviar paquetes/mensajes a todos los nodos vecinos garantiza la entrega del mensaje, siempre y cuando haya un camino viable hacia el destinatario. Sin embargo, esta simplicidad viene a un alto costo en términos de eficiencia y escalabilidad. En nuestras pruebas, pudimos observar que el tráfico de la red aumenta exponencialmente cuando se envían múltiples mensajes, lo cual es una clara indicación de su ineficiencia. Al igual que el algoritmo requiere que cada nodo tenga un alto nivel de procesamiento y una buena cantidad de memoria para gestionar la tabla de mensajes ya recibidos, con el objetivo de evitar reenvíos redundantes. Esto convierte al algoritmo en un candidato poco práctico para redes con dispositivos de recursos limitados o para entornos donde la eficiencia en el uso de los recursos es crítica.

Si bien la confiabilidad de la entrega del mensaje está prácticamente garantizada (a menos que haya un fallo en la red), esta confiabilidad se compra a expensas de la complejidad añadida en cada nodo y el gasto general de recursos de red. En situaciones donde se requiere alta confiabilidad, pero los recursos son escasos, el algoritmo de Flooding podría no ser la mejor elección.

Distance Vector

Una de las ventajas más notables del algoritmo Distance Vector es su relativa eficiencia en comparación con el Flooding. La eficiencia se debe en parte a que cada nodo solo necesita comunicarse con sus vecinos inmediatos, en lugar de transmitir información a través de toda la red. Sin embargo, esta eficiencia tiene un costo: la actualización de las tablas de enrutamiento puede ser más lenta, especialmente en redes grandes y dinámicas. Esta lentitud puede llevar a rutas subóptimas o incluso a bucles de enrutamiento si no se maneja adecuadamente.

El algoritmo también puede ser bastante robusto en redes donde la topología no cambia con frecuencia. El uso del algoritmo de Bellman-Ford ayuda en la convergencia hacia una tabla de enrutamiento óptima. Sin embargo, este algoritmo es susceptible a "count-to-infinity" problems y necesita mecanismos adicionales como "route poisoning" o "split horizon" para mejorar su confiabilidad.

Link State

Nuestra implementación del algoritmo de enrutamiento Link-State demuestra no solo eficiencia en el cálculo de rutas óptimas, sino también una notable flexibilidad y adaptabilidad. Una de las fortalezas más notables es que el programa puede adaptarse a cambios en la topología de la red sin requerir modificaciones en el código. Esto se debe a que la topología se carga desde un archivo de texto externo, lo que permite a los usuarios ajustar la configuración de la red simplemente actualizando este archivo. Aunque nuestra implementación está diseñada para una simulación 'offline', la arquitectura es lo suficientemente robusta como para adaptarse a escenarios más complejos en futuras iteraciones del proyecto."

Conclusiones

Una de las principales conclusiones acerca del algoritmo de Flooding es que, aunque garantiza que el mensaje llegará a su destino si hay una ruta disponible, su enfoque de "difusión total" lo hace tremendamente ineficiente. Este algoritmo podría ser útil en escenarios donde la entrega garantizada es más crítica que la eficiencia, pero es poco práctico para redes grandes y ocupadas debido al riesgo de sobrecarga. Su mayor ventaja es la implementación fácil que provee.

El algoritmo de Distance Vector ofrece un compromiso entre la eficiencia y la complejidad, lo que lo hace adecuado para redes de tamaño mediano que no experimentan cambios frecuentes en la topología. La eficiencia relativa en el uso de recursos y el tiempo de convergencia moderado lo hacen aplicable en una variedad más amplia de escenarios en comparación con Flooding.

El algoritmo de enrutamiento Link-State se ha mostrado como una solución efectiva y precisa para determinar rutas óptimas en una red de nodos interconectados. A diferencia de otros algoritmos como Distance Vector, Link-State ofrece una visión más global de la red, ya que cada nodo mantiene una tabla completa de rutas óptimas a todos los demás nodos en la red. Esto resulta en un enrutamiento más fiable y menos susceptible a bucles de enrutamiento y otros problemas comunes en algoritmos más simplistas.

Comentarios

Fue bastante interesante observar cómo cada algoritmo tiene sus propias fortalezas y debilidades, y cómo cada uno podría adaptarse a diferentes escenarios en el mundo real. En este sentido, el laboratorio fue más allá del simple aprendizaje académico; nos hizo pensar más críticamente sobre cómo se podrían aplicar estos conceptos en situaciones prácticas.

Si tuviéramos que mejorar algo, quizás sería genial incorporar algún elemento de interacción en tiempo real o alguna forma de simular cambios en la red para ver cómo los algoritmos responden. Pero en general, consideramos que fue una excelente manera de profundizar en un tema que es crucial para cualquiera que quiera entender cómo funcionan las redes.

Referencias

Wikipedia. (2023, 2 de septiembre). Inundación de red. En Wikipedia.

https://es.wikipedia.org/wiki/Inundaci%C3%B3n_de_red#:~:text=La%20inundaci%C3%B3n%20

W0lff4ng.org. (n.d.). Protocolos de enrutamiento Link-State. W0lff4ng.org. Obtenido el 2 de septiembre de 2023, de

<https://www.w0lff4ng.org/protocolos-de-enrutamiento-link-state/#:~:text=Los%20protocolos%20link%2Dstate%20utilizan,saltos>)

Pluralsight. (n.d.). What are Distance Vector routing protocols?. Pluralsight Blog. Obtenido el 2 de septiembre de 2023, de

<https://www.pluralsight.com/blog/it-ops/dynamic-routing-protocol#:~:text=What%20are%20Distance%20Vector%20routing,packets%20lost%2C%20or%20something%20similar>.

GeeksforGeeks. (n.d.). Fixed and Flooding Routing Algorithms. GeeksforGeeks. Obtenido el 2 de septiembre de 2023, de

<https://www.geeksforgeeks.org/fixed-and-flooding-routing-algorithms/>