

Universidad del Valle de Guatemala
Facultad de Ingeniería



Redes
Laboratorio #3 Parte 2

Algoritmos de Enrutamiento

JUAN ANGEL CARRERA SOTO 20593

JOSE MARIANO REYES HERNANDEZ 20074

JUAN CARLOS BAJAN CASTRO 20109

Descripción de la Práctica

En la primera parte del laboratorio se desarrollaron los algoritmos de enrutamiento y se probó su funcionamiento de forma hard-coded, es decir, se proporcionó de forma manual la información que cada algoritmo necesitaba tanto para la creación de las tablas de enrutamiento como para el envío de un paquete de un nodo origen a un nodo destino.

En la segunda parte se estará simulando este tipo de infraestructura utilizando como base el chat `alumchat.xyz`, para el envío de los mensajes necesarios para construir las tablas de enrutamiento y la funcionalidad de enviar un paquete de un nodo origen a un nodo destino utilizando dicha tabla.

En este laboratorio se implementaron los siguientes algoritmos:

Flooding: Este algoritmo difunde cada paquete que recibe a todos sus vecinos excepto al nodo del cual lo recibió. Eventualmente el paquete llegará a su destino. Es simple pero genera mucho tráfico innecesario.

Distance Vector: Cada nodo mantiene una tabla con las distancias a cada destino a través de cada vecino. Periódicamente los nodos intercambian sus tablas con sus vecinos para actualizar sus rutas. Es adaptativo pero lento ante cambios.

Link State Routing: Cada nodo conoce toda la topología de red. Utiliza el algoritmo de Dijkstra para calcular la ruta más corta a cada destino y construye una tabla de forwarding. Es complejo pero converge rápido. Requiere flooding de topología.

Resultados

Distance Vector

Con el algoritmo Distance Vector, cada nodo construyó inicialmente una tabla de enrutamiento con las distancias directas a sus vecinos adyacentes. Una vez que todos los nodos conocieron sus vecinos directos, se intercambiaron sus tablas y utilizaron el algoritmo de Bellman-Ford para recalcular las rutas más óptimas a los demás nodos no adyacentes. De esta manera, nodos que no tenían conexión directa pudieron determinar los pesos y la mejor ruta a seguir para llegar a otros nodos destino en la red. El intercambio periódico de tablas permitió que el algoritmo Distance Vector convergiera hacia rutas óptimas entre todos los pares de nodos de la red.

Al probar el envío de mensajes entre dos nodos no adyacentes, se obtuvieron resultados satisfactorios. Los mensajes pudieron ser enrutados exitosamente a través de nodos intermedios que actuaron como puente entre el nodo origen y el nodo destino. Inicialmente, al no ser vecinos directos, estos nodos no tenían conexión entre sí. Sin embargo, gracias a la convergencia de los algoritmos de enrutamiento, se determinaron rutas óptimas utilizando uno o más saltos a través de otros nodos intermedios. De esta manera, el mensaje del nodo origen pudo alcanzar el nodo destino remotamente ubicado en la topología de red,

PROBLEMS	OUTPUT	DEBUG CONSOLE	TERMINAL	PORTS
<p>un nodo</p> <p>6. Enviar presencia a vecinos</p> <p>7. Imprimir tabla de ruteo del nodo</p> <p>8. Salir</p> <p>>> 5</p> <p>Ingresar el Nodo destino</p> <p>Ingresar el Mensaje C</p> <p>>> Hola C</p> <p>Qué desea hacer? (I</p>	<p>pción)</p> <p>1. Setear vecinos</p> <p>2. Agregar vecino</p> <p>3. Eliminar vecino</p> <p>4. Mostrar detalles de vecinos</p> <p>5. Enviar mensaje a un nodo</p> <p>6. Enviar presencia a vecinos</p> <p>7. Imprimir tabla de ruteo del nodo</p> <p>8. Salir</p> <p>>> Tienes una comunicación de F> Hola C de: A</p> <p>v</p> <p>>> []</p>	<p>pción)</p> <p>1. Setear vecinos</p> <p>2. Agregar vecino</p> <p>3. Eliminar vecino</p> <p>4. Mostrar detalles de vecinos</p> <p>5. Enviar mensaje a un nodo</p> <p>6. Enviar presencia a vecinos</p> <p>7. Imprimir tabla de ruteo del nodo</p> <p>8. Salir</p> <p>>> []</p>	<p>pción)</p> <p>1. Setear vecinos</p> <p>2. Agregar vecino</p> <p>3. Eliminar vecino</p> <p>4. Mostrar detalles de vecinos</p> <p>5. Enviar mensaje a un nodo</p> <p>6. Enviar presencia a vecinos</p> <p>7. Imprimir tabla de ruteo del nodo</p> <p>8. Salir</p> <p>>> Tienes una comunicación de A> Hola C / >> A >> C</p> <p>v</p> <p>>> []</p>	<p>pción)</p> <p>1. Setear vecinos</p> <p>2. Agregar vecino</p> <p>3. Eliminar vecino</p> <p>4. Mostrar detalles de vecinos</p> <p>5. Enviar mensaje a un nodo</p> <p>6. Enviar presencia a vecinos</p> <p>7. Imprimir tabla de ruteo del nodo</p> <p>8. Salir</p> <p>>> []</p>

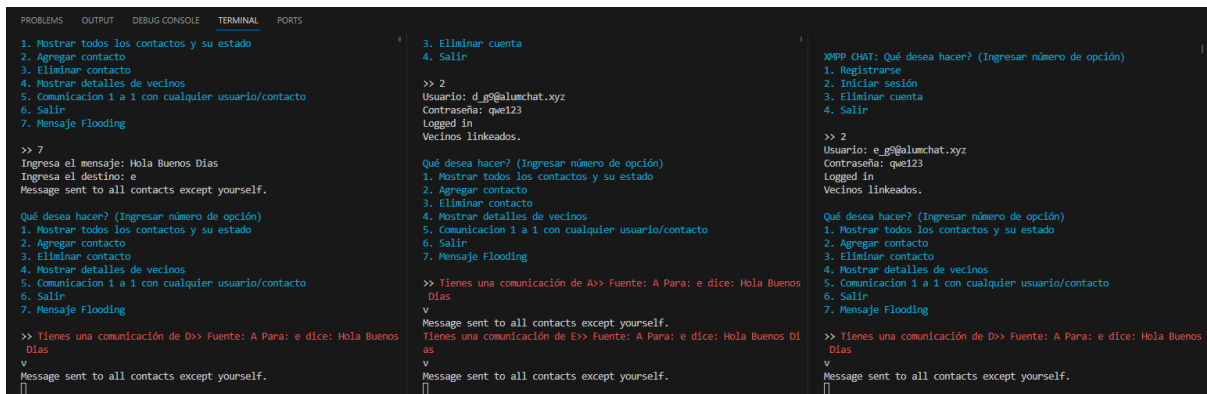
Con el algoritmo de LinkState, cada nodo se inicializa su topología con solo sus vecinos. Una vez todos los vecinos de la red están conectados se realiza una sincronización de topología para descubrir toda la topología. De esta forma el método permite que, aunque dos nodos no estén conectados directamente, puedan comunicarse a través de uno o varios nodos intermedios que funcionan como puentes, estableciendo así un camino eficiente para la transmisión del mensaje, siempre que exista una ruta viable en la topología de la red actual. Es así cómo se garantiza que el mensaje alcanzará su destino de la manera más óptima posible, utilizando el estado actualizado y sincronizado de la tabla de enrutamiento.

[illegible]

El algoritmo Flooding es el algoritmo más sencillo de entre todos los demás. Dado que se trata de un mensaje que se envía por toda la red inundando a todos, como dice su nombre, es un algoritmo perfecto para asegurarnos de que el mensaje llegará eventualmente, sin embargo, es poco eficiente dado que no busca el mejor camino para llegar al destino, claro, al recorrer todos los posibles caminos al mismo tiempo, toma el mismo tiempo en llegar al destino. Sin embargo, aplica demasiado estrés sobre el sistema.

En nuestras pruebas hallamos distintas complicaciones, principalmente al momento de diferenciar los mensajes, los resultados muestran un esfuerzo exitoso para enviar el mensaje y enviarlo cada vez que es recibido, de esta manera nos aseguramos de replicar el funcionamiento exacto del algoritmo. Como podemos observar, el nodo de en medio ocurre algo interesante, el mensaje es recibido dos veces, esto se da porque la fuente del mensaje es distinto y eso causa que el mensaje sea recibido más de una vez, sin embargo el mensaje no será recibido más de la misma cantidad de hermanos que posee dicho nodo y

eso se permitió porque, así también podemos diferenciar entre mensajes enviados por distintos nodos y no perdemos información en la red.



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
1. Mostrar todos los contactos y su estado
2. Agregar contacto
3. Eliminar contacto
4. Mostrar detalles de vecinos
5. Comunicación 1 a 1 con cualquier usuario/contacto
6. Salir
7. Mensaje Flooding

>> 7
Ingresa el mensaje: Hola Buenos Días
Ingresa el destino: e
Message sent to all contacts except yourself.

Qué desea hacer? (Ingresar número de opción)
1. Mostrar todos los contactos y su estado
2. Agregar contacto
3. Eliminar contacto
4. Mostrar detalles de vecinos
5. Comunicación 1 a 1 con cualquier usuario/contacto
6. Salir
7. Mensaje Flooding

>> Tienes una comunicación de D>> Fuente: A Para: e dice: Hola Buenos Días
V
Message sent to all contacts except yourself.
[]

3. Eliminar cuenta
4. Salir

>> 2
Usuario: d_g@alumchat.xyz
Contraseña: que123
Logged in
Vecinos linkeados.

Qué desea hacer? (Ingresar número de opción)
1. Mostrar todos los contactos y su estado
2. Agregar contacto
3. Eliminar contacto
4. Mostrar detalles de vecinos
5. Comunicación 1 a 1 con cualquier usuario/contacto
6. Salir
7. Mensaje Flooding

>> Tienes una comunicación de A>> Fuente: A Para: e dice: Hola Buenos Días
V
Message sent to all contacts except yourself.
Tienes una comunicación de E>> Fuente: A Para: e dice: Hola Buenos Días
V
Message sent to all contacts except yourself.
[]

WEP GWT. Qué desea hacer? (Ingresar número de opción)
1. Registrarse
2. Iniciar sesión
3. Eliminar cuenta
4. Salir

>> 2
Usuario: e_g@alumchat.xyz
Contraseña: que123
Logged in
Vecinos linkeados.

Qué desea hacer? (Ingresar número de opción)
1. Mostrar todos los contactos y su estado
2. Agregar contacto
3. Eliminar contacto
4. Mostrar detalles de vecinos
5. Comunicación 1 a 1 con cualquier usuario/contacto
6. Salir
7. Mensaje Flooding

>> Tienes una comunicación de D>> Fuente: A Para: e dice: Hola Buenos Días
V
Message sent to all contacts except yourself.
[]
```

Discusión

Por parte de Distance Vector, este demostró ser efectivo para determinar rutas óptimas entre todos los nodos de la red simulada. Una ventaja clave es que solo requiere que cada nodo conozca sus vecinos directos e intercambie tablas con ellos, sin necesidad de tener visibilidad completa de toda la topología como en Link State. Esto lo hace escalable a redes grandes.

Sin embargo, presenta la desventaja de ser lento para converger ante cambios en la red, ya que la información se propaga de nodo en nodo. Además, es susceptible a problemas de routing looping si no se implementan mecanismos como split horizon.

En nuestra implementación, la ejecución periódica de Bellman-Ford permitió recalcular rutas e incorporar cambios en los pesos o topología. El intercambio de tablas converge a un estado estable con rutas óptimas incluso entre nodos no adyacentes.

Un aspecto positivo es que los mensajes pudieron ser enrutados correctamente incluso entre nodos no vecinos, demostrando que los nodos intermedios actuaron efectivamente como puentes. Esto verifica el correcto funcionamiento del algoritmo para determinar saltos múltiples en la ruta más corta.

Tomando en consideración el Link_state, lo que podemos decir de este algoritmo es que este algoritmo se caracteriza por su rápida convergencia hacia un estado estable tras los cambios en la red, superando así a Distance Vector en este aspecto. Además, proporciona a cada nodo una visión completa de la topología, permitiendo rutas más óptimas y una predicción más exacta del camino que tomará un mensaje. No obstante, también presenta desventajas significativas como su complejidad elevada de implementación y mantenimiento, además de un mayor requerimiento de recursos computacionales. A medida que la red se amplía, la tabla de enrutamiento puede incrementar considerablemente en tamaño, lo que se traduce en un mayor consumo de memoria y CPU.

Dentro de las ventajas de usar LinState esta la convergencia rapsodia ya que a diferencia de Distance Vector, tiene a converger rápido a un estado estable tras un cambio en la red. Además que tiene la característica de la visibilidad de la topología cada nodo tiene una

visión completa de la topología lo que permite saber la ruta más óptima y predicciones más exactas sobre la ruta de un mensaje.

Dentro de las desventajas del LinkState está que el algoritmo es poco complejo de implementar y mantener comparado con Distance vector o flooding requiriendo un poco más de recurso computacional. Otra desventaja es que a medida de que la red crezca la tabla de enrutamiento puede volverse muy grande lo que implicaría un mayor consumo computacional.

Por otro lado, en el algoritmo de flooding, como se mencionó anteriormente, se presentaron distintos retos. Poder sincronizar los mensajes dado la gran cantidad de mensajes que se estaban enviando podía ser un punto de complicaciones para el sistema. Sin embargo, llevando un registro interno de qué mensajes son recibidos y dicho mensaje, podemos realizar grandes diferenciaciones de los mensajes que se envían por la red. De esta manera mantenemos la integridad de la data.

Como se puede observar en la imagen del algoritmo de flooding, todos los mensajes son desplegados en pantalla, esto para poder mantener un buen registro de todos los mensajes y poder observar a detalle y con exactitud el camino que toman los mensajes.

Como recomendación se podría decir que es importante mantener ese registro de mensajes siempre activo en la red, pues en nuestra implementación, dado que son mensajes locales y se pierden cuando el sistema deja de funcionar, pueden haber problemas de bucles y mensajes enviados muchas veces dado que se pierde esa información.

Comentario grupal

El algoritmo Distance Vector efectivamente funciona bien cuando los nodos tienen conocimiento de la topología de red completa inicialmente. El poder construir las tablas de enrutamiento con visibilidad total de la red permite lograr convergencia rápidamente hacia un estado óptimo.

Sin embargo, una desventaja clave es que Distance Vector depende de la propagación paso a paso de cualquier cambio en la topología, lo cual puede ser lento. Si ocurre un cambio como la caída de un enlace, el algoritmo demorará en reconverger, durante lo cual podría enviar tráfico por rutas subóptimas o erróneas.

Para lidiar con cambios frecuentes en la topología, Distance Vector debe complementarse con actualizaciones incrementales frecuentes. Cada nodo debe comunicar a sus vecinos cambios tan pronto sucedan para acelerar la propagación de dicha información. Adicionalmente, se podrían implementar optimizaciones como trigger updates y hold-down timers para lidiar con fluctuaciones.

Flooding parece ser uno de las primeras estrategias para envío de mensajes, es muy útil pero requiere muchos recursos, además que compromete en gran medida la seguridad de la información. La mayoría de algoritmos inician con un pseudo-flooding para poder exponer la topología del sistema a los demás nodos. En este punto se puede observar que dicho algoritmo es fundamental para la comunicación en la red.

Conclusiones

- Distance Vector presenta ventajas de simplicidad y bajo overhead de mensajes de control, pero sacrifica velocidad de convergencia. Para nuestra topología simulada, los resultados fueron satisfactorios. En futuros trabajos se podría analizar su desempeño en redes más grandes y dinámicas.
- El algoritmo de flooding presenta desafíos significativos debido a la enorme cantidad de mensajes que se envían, lo que puede complicar la sincronización. Para abordar esto, es esencial llevar un registro preciso de los mensajes recibidos, lo que permite diferenciar claramente los mensajes transmitidos en la red y garantizar la integridad de la data. Esta práctica se refleja visualmente en la forma en que todos los mensajes se muestran en la pantalla, permitiendo un seguimiento detallado de la trayectoria de cada mensaje. Es vital mantener un registro persistente de estos mensajes en la red, ya que en implementaciones donde los mensajes son locales y desaparecen cuando el sistema se apaga, pueden surgir problemas como bucles o reenvíos excesivos debido a la pérdida de información.
- El algoritmo Link-State se destaca por ofrecer una visión más completa de la topología de una red, comparado con el método Distance Vector. Esto permite una convergencia más rápida hacia rutas óptimas tras cualquier modificación en la red, garantizando así una mayor estabilidad y fiabilidad en el enrutamiento de los mensajes. Sin embargo, esta técnica tiene el costo de una mayor complejidad tanto en su implementación como en su mantenimiento, demandando más recursos computacionales y haciendo que la gestión de redes grandes sea potencialmente más desafiante. Además, la necesidad de que cada nodo tenga una comprensión completa de la red puede resultar en una sobrecarga significativa, particularmente en escenarios de redes extensas y dinámicas. A pesar de estos desafíos, el algoritmo Link-State sigue siendo una herramienta poderosa y eficaz para el enrutamiento de redes, siendo capaz de encontrar el camino más corto y optimizado para la transmisión de datos, lo que lo convierte en una opción sólida para una variedad de aplicaciones en el campo de las ciencias de la computación y las redes.