Universidad del Valle de Guatemala

Facultad de Ingeniería

Redes

Laboratorio 3. Algoritmos de enrutamiento

Integrantes

Francis Aguilar 22243

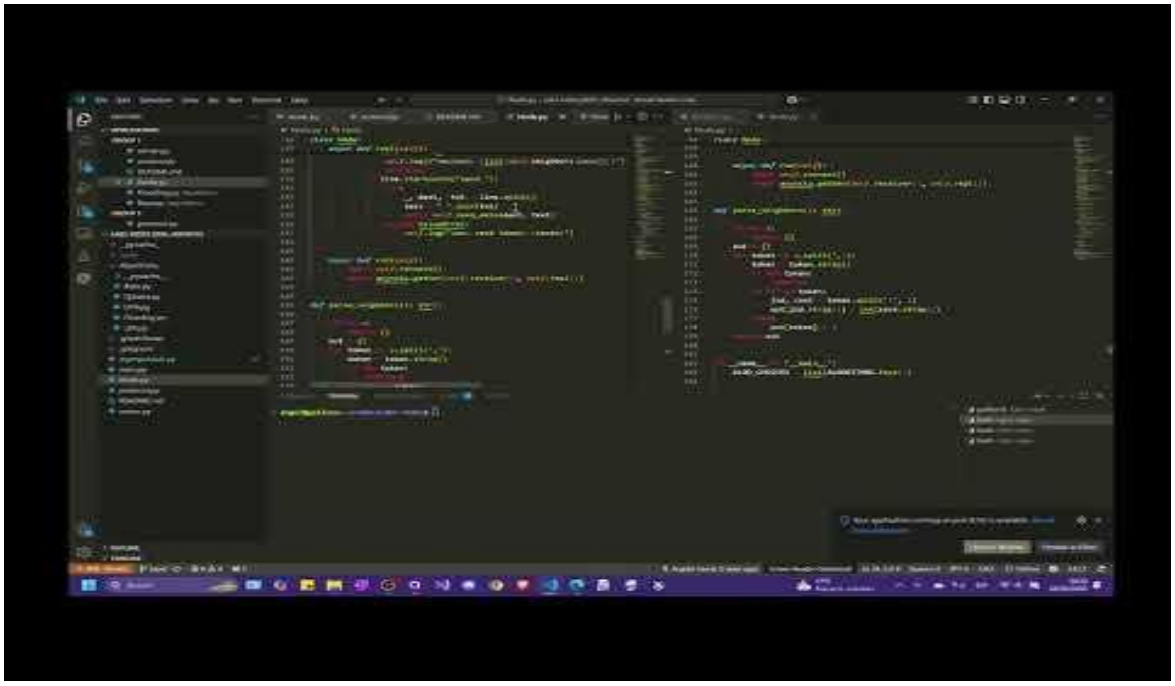
Angela García 22869

Cesar Lopez 22535

Gerardo Pineda 22880

Enlace del repositorio: https://github.com/angelargd8/Lab3-redes

Enlace del video: https://youtu.be/-70DT0xZ5ss



Objetivos

- Conocer los algoritmos de enrutameinto utilizados en las implementaciones actuales de Internet.
- Comprender cómo funcionan las tablas de enrutamiento.
- Implementar los algoritmos de enrutamiento y probarlos en una red simulada sobre un protocolo de capa superior (XMPP o similar).
- Analizar el funcionamiento de los algoritmos de enrutamiento.

Introducción

En el ámbito de las redes de computadoras, los algoritmos de enrutamiento juegan un papel importante, ya que permiten determinar el camino que seguirá la información desde un origen hasta un destino dentro de la red. Que funcionen correctamente asegura la eficiencia, confiabilidad y robustez en la transmisión de datos. Existen distintos enfoques para resolver este problema, cada uno con características y aplicaciones específicas. El algoritmo de Dijkstra, que calcula las rutas más cortas con base en la topología conocida, Flooding, que reenvía paquetes a todos los vecinos garantizando la entrega, aunque con sobrecarga de tráfico, Link State Routing, que combina el intercambio de información de estado de enlace con Dijkstra para obtener rutas óptimas, y Distance Vector Routing, que utiliza el intercambio periódico de vectores de distancia entre vecinos para converger en una tabla de enrutamiento global. La implementación y comparación de estos algoritmos en una red simulada permite comprender mejor sus ventajas, limitaciones y aplicaciones prácticas.

Descripción de los algoritmos

Dijkstra

El algoritmo de Dijkstra es un método para encontrar la ruta más corta desde un nodo origen hacia todos los demás nodos en una red o grafo ponderado. Selecciona en cada paso el nodo con la menor distancia acumulada y actualizando las distancias de sus vecinos hasta que se procesen todos. Es muy usado en enrutamiento de redes porque garantiza caminos óptimos siempre que los costos de los enlaces sean positivos.

Para la implementación en este laboratorio, el algoritmo adapta el algoritmo clásico de rutas más cortas a un entorno de enrutamiento distribuido: cada nodo mantiene una topología de la red con sus vecinos y costos, y mediante una cola de prioridad calcula las distancias mínimas desde sí mismo hacia todos los destinos. Además de registrar las distancias, guarda los primeros saltos posibles hacia cada destino, lo que permite decidir por dónde reenviar paquetes.

Flooding

El flooding es un método de enrutamiento en el que cada paquete que llega a un nodo se reenvía a todos sus vecinos, excepto al que lo envió originalmente. Aunque garantiza que el mensaje llegue a todos los nodos de la red, puede generar gran cantidad de tráfico redundante y congestión. Para controlarlo, se usan mecanismos como límites de tiempo de vida (TTL) o el marcado de paquetes ya transmitidos.

Esta implementación de Flooding es muy simple y directa, cada nodo solo necesita conocer a sus vecinos inmediatos y, cuando recibe un mensaje de datos, lo reenvía a todos ellos excepto al que lo envió originalmente, evitando así bucles inmediatos. A diferencia de Dijkstra u otros algoritmos, no requiere de un control plane ni del cálculo de rutas, ya que no construye ni mantiene tablas de encaminamiento, sino que se basa en la replicación masiva de paquetes.

Link State Routing

El enrutamiento por estado de enlace se basa en que cada router conoce el estado y costo de sus enlaces directos y comparte esa información con todos los demás routers de la red. Con esta visión global, cada router puede construir un mapa completo de la topología y calcular las mejores rutas usando algoritmos como Dijkstra. Protocolos como OSPF se basan en este enfoque, logrando rutas rápidas y confiables.

Esta implementación de Link State Routing combina dos mecanismos: usa Flooding para el plano de control y Dijkstra para el plano de datos. Cada nodo anuncia periódicamente sus vecinos y costos en forma de mensajes LSA con número de secuencia creciente, los cuales se propagan a toda la red, al recibir un LSA nuevo, el nodo actualiza su topología local y vuelve a ejecutar Dijkstra para recalcular los caminos más cortos. Cuando debe enrutar datos, si

conoce la ruta al destino usa los next hops de Dijkstra, y si no, puede recurrir a flooding como respaldo.

Distance Vector Routing

El enrutamiento vector-distancia hace que cada router mantenga una tabla con la mejor distancia conocida a cada destino y el vecino a través del cual llegar. Periódicamente, cada router comparte su tabla con sus vecinos, y a partir de esa información, las tablas se van actualizando. Aunque es más simple que el estado de enlace, puede tener problemas como convergencia lenta y bucles, por lo que se utilizan técnicas como split horizon y poison reverse.

Esta implementación de Distance Vector Routing (DVR) sigue el principio del algoritmo de Bellman-Ford: cada nodo mantiene una tabla de enrutamiento con el costo estimado a cada destino y el siguiente salto para alcanzarlo, además de un vector de distancias que resume esa información. Al iniciar, el nodo llena su tabla con costo 0 hacia sí mismo y los costos directos a sus vecinos, luego envía su vector a todos ellos. Cuando recibe un vector de un vecino, calcula los costos hacia cada destino a través de ese vecino y actualiza su tabla si encuentra una ruta más barata; si hubo cambios, vuelve a anunciar su vector. Para enrutar un mensaje de datos, consulta la tabla y reenvía el paquete al siguiente salto más conveniente.

Conexion y pruebas de los algoritmos

Infraestructura de conexion

Para la interconexion real entre nodos utilizamos Redis Pub/Sub provisto por el curso, donde cada nodo publica en los canales de sus vecinos y se suscbibe unicamente a su propio canal. La nomenclatura de canales sigue el patron:

channel:<seccion>.<topologia>.<nodo>(.prueba)

El nodo redis se conecta a lab3.redesuvg.coud:5379 y abre un bucle asincronico que escucha su canal y processa mensajes segun su tipo (hello, info, message).

Protocolo de mensajes

Se mantuvo un formato JSON:

{ proto, type, from, to, ttl, headers, payload }

En la version TCP local ademas normalizamos headers a diccionario y agregamos headers.msg_id para de-duplicacion; el servidor inyecta headers.via al reenviar hacia el destino.

En la version Redis usamos elm ismo esquema logico con tipos:

- Hello: desscubrimiento / arranque.

- Info: anuncios de control.
- Message: datos de usuario.

Pruebas por algoritmo

- Flooding
  Se lanzaron de 3 – 5 nodos con vecinos directos definidos y desde un nodo origen se ejecutó send <dest> “hola” para observar la propagacion.

  Criterios de exito: El destino imprime el payload; los routers intermedios registran reenvios y el ttl decrece en cada salto.

- Link State Routing (LSR)
  Con 4 – 6 nodos, iniciar LSR: cada nodo anuncia LSA (vecinos, costos) y ejecuta Dijkstra local.
  Enviar message entre dos nodos no adyacentes; verificar que el next hop coincide con el camino minimo.
  Fallback: si no hay ruta conocida aun, permitir flooding temporal de datos hasta converger.

  Criterios de exito: Convergencia tras recibir LSA’s y rutas optimas en DATA.

- Distance Vector (DVR)
  Iniciar nodos con DVR, al arrancar se publica el vector local.
  Al recibir un vector, actualizar tabla si se mejora el costo, si cambia, volver a anunciar.
  Enviar DATA y verificar que se reenvia por next_hop de la tabla.

  Criterios de exito: convergencia en pocos ciclos y rutas consistentes con costos.

Resultados

Durante el desarrollo del proyecto logramos implementar distintos algoritmos de enrutamiento con el objetivo de probarlos en una red simulada. Inicialmente, el planteamiento consistía en realizar estas pruebas sobre un protocolo de capa superior como XMPP, aprovechando sus mecanismos de mensajería y comunicación entre nodos. A pesar de que las instrucciones cambiaron para la siguiente fase, de igual manera se probó la implementación de XMPP y se logró validar su funcionamiento básico en la comunicación entre nodos, aunque de manera limitada. Esto permitió comprender cómo los nodos podrían intercambiar mensajes y actualizar información de enrutamiento en un entorno real de mensajería.

Tras estas pruebas iniciales con XMPP, se decidió implementar un servidor y protocolo propios para la comunicación entre nodos. Esta solución permitió un mayor control sobre la simulación, facilitando la propagación de mensajes de control y el envío de datos de acuerdo con cada algoritmo de enrutamiento. Al contar con un entorno adaptado, fue posible observar directamente cómo cada algoritmo gestionaba la topología de la red, calcular rutas, reenviar paquetes y manejar cambios dinámicos en la conectividad de los nodos.

Los resultados obtenidos demostraron claramente las diferencias entre los algoritmos: Flooding garantiza la entrega de paquetes pero con gran sobrecarga de tráfico, Dijkstra y LSR proporcionan rutas óptimas basadas en la visión global de la red, y DVR permite construir rutas de manera distribuida aunque con posibles retrasos en la convergencia. La simulación con el servidor propio facilitó el análisis de estos comportamientos y permitió validar tanto la lógica de los algoritmos como la eficacia de la infraestructura de comunicación desarrollada para el proyecto.

Conclusiones

- Cada algoritmo mostró ventajas y limitaciones que se pudieron analizar en la simulación.
- El proyecto permitió comprender mejor la teoría y enfrentar retos prácticos de implementación.
- El proyecto permitió comprender la teoría, enfrentar retos prácticos de implementación y analizar el desempeño de distintos algoritmos en un entorno controlado.
- Flooding asegura entrega pero genera sobrecsarga. Es util como respaldo en LSR durante convergencia.
- LSR ofrece rutas optimas y reconverge rapido al recibir nuevos LSA's, el costo es mantener la LSDB y ejecutar Dijkstra.
- DVR es sencillo y distribuido, puede tardar mas en converger y requiere cuidados contra loops.

Referencias

W3Schools.com. (s. f.). https://www.w3schools.com/dsa/dsa_algo_graphs_dijkstra.php

GeeksforGeeks. (2023, 22 abril). Fixed and Flooding Routing algorithms. GeeksforGeeks. https://www.geeksforgeeks.org/computer-networks/fixed-and-flooding-routing algorithms/

GeeksforGeeks. (2025, 11 julio). Unicast routing Link State routing. GeeksforGeeks. https://www.geeksforgeeks.org/computer-networks/unicast-routing-link-state-routing/

GeeksforGeeks. (2024b, diciembre 27). Distance Vector Routing (DVR) protocol. GeeksforGeeks.

https://www.geeksforgeeks.org/computer-networks/distance-vector-routing-dvr-protocol/