UNIVERSIDAD DEL VALLE DE GUATEMALA
Redes
Sección 30

Laboratorio3

Abby Donis
Edwin de Leon - 22809

GUATEMALA, 11 de septiembre de 2025

Descripción de la Práctica

La práctica tuvo como objetivo principal comprender el funcionamiento de diferentes algoritmos de enrutamiento mediante la implementación en Python y la simulación en una red definida a través de archivos JSON.

El archivo de configuración (topology_sec30.json) describe una topología con nodos y enlaces, donde cada enlace tiene un costo de transmisión (edge_cost). Dichos costos representan métricas como el ancho de banda, la latencia o la carga del enlace.

Durante la simulación, los nodos intercambian mensajes mediante dos mecanismos fundamentales:

- **HELLO**: utilizado para descubrir vecinos directos, cada nodo envía mensajes periódicos a sus enlaces inmediatos, confirmando la conectividad activa, esto asegura la detección de fallos locales y permite conocer el costo directo hacia cada vecino.
- **Flooding**: mecanismo que propaga mensajes a través de toda la red, cuando un nodo recibe un paquete, lo reenvía a todos sus vecinos excepto al remitente, esto garantiza que la información llegue a todos los nodos, incluso a aquellos que no son vecinos directos.

El análisis consistió en observar la construcción de la tabla de enrutamiento, midiendo:

- **edge_cost**: costo del enlace inmediato.
- **hops**: número de saltos entre origen y destino.
- **total_cost**: costo acumulado de la ruta.
- **time_left** y **up**: tiempo restante de validez de la información y tiempo de actividad del nodo.

La práctica permitió conectar teoría con experimentación, reforzando conceptos de protocolos como RIP (Vector de Distancia) y OSPF (Link State).

Descripción de los Algoritmos Utilizados y su Implementación

1. Flooding

- **Descripción**: cada nodo que recibe un mensaje lo reenvía a todos sus vecinos, excepto al remitente.
- **Ventajas**: robustez, alta probabilidad de entrega en redes pequeñas.
- **Desventajas**: tráfico redundante, riesgo de tormentas de broadcast en redes grandes.
- **Implementación**: en el código, este algoritmo está en src/algorithms/flooding.py. Se controla mediante identificadores únicos para evitar reenvíos infinitos.

2. Vector de Distancia (Distance Vector)

- **Descripción**: cada nodo mantiene una tabla con la distancia a todos los demás nodos y la actualiza intercambiando información periódica con sus vecinos.
- **Base teórica**: implementa el algoritmo de **Bellman-Ford**.
- **Ventajas**: simplicidad, bajo uso de memoria.
- **Desventajas**: convergencia lenta, problemas de bucles de conteo infinito.
- **Implementación**: en src/algorithms/distance_vector.py. Cada nodo recibe tablas de vecinos y actualiza la suya sumando el costo del enlace.

3. Estado de Enlace (Link State)

- **Descripción**: cada nodo descubre sus vecinos y costos, y difunde esta información a toda la red. Con ella, cada nodo construye un mapa completo de la topología.

- **Base teórica**: protocolo **OSPF (Open Shortest Path First)**.
- **Ventajas**: convergencia rápida, rutas más eficientes.
- **Desventajas**: requiere mayor memoria y procesamiento.
- **Implementación**: en src/algorithms/link_state.py. Cada nodo inunda la red con sus estados de enlace y mantiene una base de datos de topología.

4. Dijkstra

- **Descripción**: algoritmo matemático que, a partir de la información completa de la topología (obtenida con Link State), calcula la ruta de menor costo desde un nodo a todos los demás.
- **Ventajas**: siempre encuentra rutas óptimas.
- **Desventajas**: mayor costo computacional en redes grandes.
- **Implementación**: en src/algorithms/dijkstra.py. El nodo inicializa costos infinitos, establece costo cero hacia sí mismo y, mediante una cola de prioridad, actualiza rutas mínimas.

Resultados

Durante la ejecución en la topología del archivo topology_sec30.json, los resultados observados fueron los siguientes:

Vecinos detectados con HELLO

==== VECINOS (hello) ====

sec30.grupo7.nodo7 edge_cost= 4 hops= 1 total_cost= 4 time_left= 6 up=00:15

sec30.grupo11.nodo11 edge_cost= 12 hops= 1 total_cost= 12 time_left= 6 up=00:15

No vecinos descubiertos vía Flooding

==== NO VECINOS (flooding) ====

sec30.grupo8.nodo8 edge_cost= - hops= 2 total_cost= 11 up=00:08

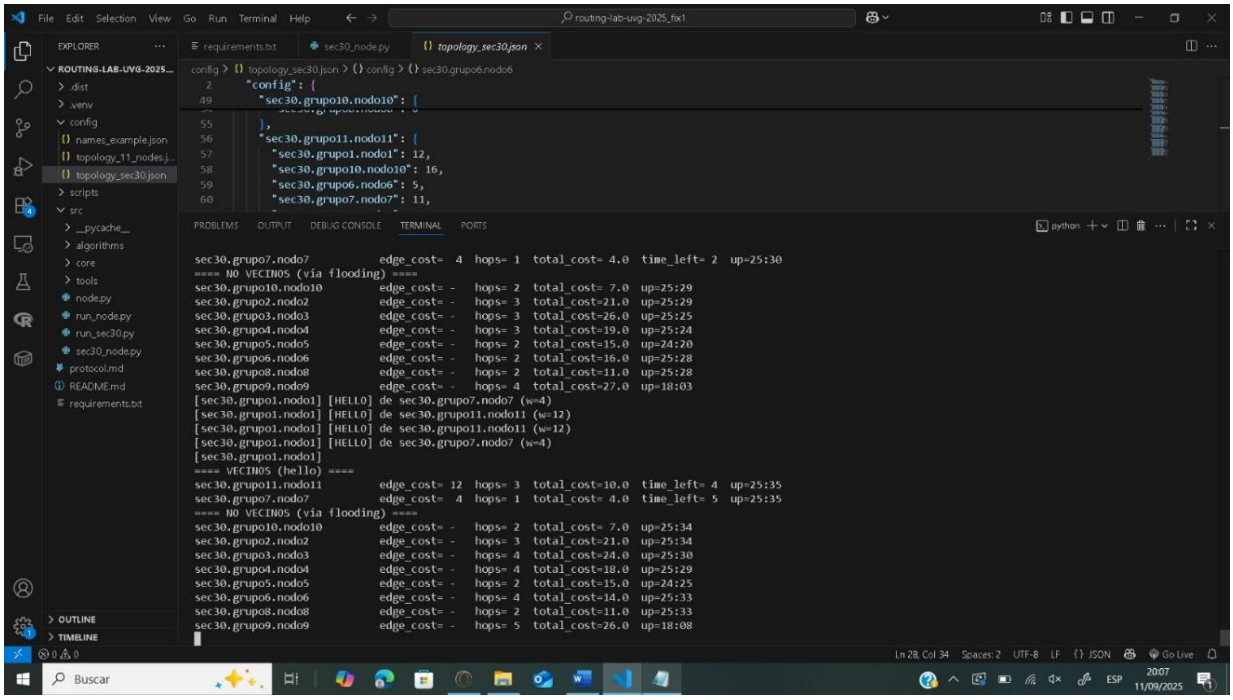

Imagen1: Resultado en clase probando con los nodos activos

Observaciones

- Los vecinos inmediatos tienen costos bajos (4 y 12), confirmando enlaces directos.
- Los nodos alcanzados por flooding requieren dos saltos y presentan un costo acumulado mayor (11).
- La métrica hops diferencia caminos directos de caminos indirectos, validando la utilidad del flooding para descubrir topologías más amplias.

Discusión

El análisis muestra la importancia de combinar diferentes algoritmos en protocolos de enrutamiento reales:

- **HELLO** proporciona precisión en la detección de enlaces activos, pero no ofrece información más allá de los vecinos.
- **Flooding** garantiza que todos los nodos conozcan la existencia de otros, aunque con un costo en tráfico.
- **Vector de Distancia** permite que la red converja hacia un estado estable compartiendo tablas periódicamente, aunque con riesgo de lentitud.
- **Link State + Dijkstra** aportan rutas más óptimas y rápidas, siendo los algoritmos preferidos en redes modernas.

Esto explica por qué protocolos como **RIP** (basado en Vector de Distancia) son usados en redes pequeñas, mientras que **OSPF** (basado en Link State y Dijkstra) se emplea en redes empresariales de gran escala.

Conclusiones

1. La práctica permitió diferenciar entre métodos de descubrimiento local (HELLO) y global (Flooding).
2. Los algoritmos Vector de Distancia y Link State representan enfoques distintos para lograr tablas de enrutamiento estables.
3. Dijkstra asegura rutas óptimas, lo cual es crucial en redes con múltiples alternativas.
4. Las métricas observadas (costo, hops, tiempo) muestran la importancia de elegir correctamente el algoritmo según el tamaño de la red.

Comentarios

Abby: El uso de Python y topologías definidas en JSON facilitó la simulación y experimentación, permitiendo modificar escenarios rápidamente, esta práctica prepara al estudiante para entender protocolos de enrutamiento reales y su aplicación en ambientes de ingeniería de redes.

Edwin: La implementación del algoritmo Link State en la práctica presentó una dificultad mayor en comparación con Flooding o Vector de Distancia, ya que requiere que todos los nodos mantengan una base de datos de estado de enlaces sincronizada y consistente, esto implica que cualquier cambio en la topología (caída de un enlace, incorporación de un nodo, variación en los costos) debe ser propagado correctamente a todos los nodos de la red.

Enlace de Repositorio
Github: https://github.com/EJGDLG/Lab3Redes.git

Referencias

Biradar, A. G. (2020). A Comparative Study on Routing Protocols: RIP, OSPF and EIGRP and Their Analysis Using GNS-3. https://doi.org/10.1109/icraie51050.2020.9358327

Rathi, B., & Singh, E. (2015). Performance Analysis of Distance Vector and Link State Routing Protocols. International Journal of Computer Science Trends and Technology, 3. https://www.ijcstjournal.org/volume-3/issue-4/IJCST-V3I4P4.pdf?utm_source=chatgpt.com

Verma, A., & Bhardwaj, N. (2016). A Review on Routing Information Protocol (RIP) and Open Shortest Path First (OSPF) Routing Protocol. International Journal of Future Generation Communication and Networking, 9(4), 161–170. https://doi.org/10.14257/ijfgcn.2016.9.4.13

Kabir, M. H., Kabir, M. A., Islam, M. S., Mortuza, M. G., & Mohiuddin, M. (2021). Performance Analysis of Mesh Based Enterprise Network Using RIP, EIGRP and OSPF Routing Protocols. 47–47. https://doi.org/10.3390/ecsa-8-11285