Universidad del Valle de Guatemala

Laboratorio 3 — Algoritmos de Enrutamiento

Avances en Dijkstra e infraestructura de red

Integrantes:
Diederich Solis
Jorge López
Angel Herrante
jose gramajo

Repositorio:
`https://github.com/DiederichSolis/Redes_Lab3.git`

15 de agosto de 2025

Resumen

En esta entrega se presenta un avance funcional del Laboratorio 3, consistente en la implementación de:

- Algoritmo de Dijkstra para cálculo de rutas óptimas.
- Infraestructura básica de comunicación entre nodos mediante sockets y manejo de hilos.
- Proceso de ruteo y reenvío (*forwarding*) de paquetes.

El prototipo desarrollado permite levantar múltiples nodos en una topología predefinida, intercambiar información de estado de enlace (LSPs), calcular tablas de ruteo con Dijkstra y reenviar mensajes de datos (`DATA`) de origen a destino siguiendo el *next hop* calculado.

1. Descripción de lo realizado

1.1. Implementación de Dijkstra

Se implementó el algoritmo de Dijkstra en el módulo `router/dijkstra.py`, el cual recibe como entrada:

- Un grafo representado como diccionario de diccionarios (`dict[str, dict[str, float]]`).
- Un nodo origen.

El algoritmo produce:

- **dist**: distancia mínima desde el origen a cada nodo.
- **prev**: predecesores para reconstruir el camino más corto.

A partir de **prev** se determina el vecino inmediato (*next hop*) hacia cada destino.

1.2. Infraestructura de sockets e hilos

El módulo `router/node.py` implementa la clase `Node`, que encapsula:

- **Socket UDP** para envío y recepción de mensajes.
- **Hilos de ejecución**:

1. `_listener`: escucha paquetes entrantes y los coloca en una cola.
2. `_forwarding_loop`: procesa paquetes y decide reenvío o entrega local.
3. `_routing_loop`: recalcula la tabla de rutas periódicamente usando Dijkstra.
4. `_hello_loop`: envía `HELLO` y `LSP` periódicamente para medir RTT y propagar topología.

1.3. Formato de mensajes

Se definió la clase `Message` (`router/message.py`) con serialización/deserialización JSON. Campos principales:

- `proto`, `type`, `from`, `to`, `ttl`
- `headers`, `payload`

1.4. Demostración

Se implementó un script `run_demo.py` que:

1. Levanta cuatro nodos (`A`, `B`, `C`, `D`) en localhost con puertos distintos.
2. Simula la topología definida en `topo-sample.json`.
3. Intercambia LSPs hasta que cada nodo tiene la topología completa.
4. Calcula tablas de ruteo con Dijkstra.
5. Envía un mensaje `DATA` de A a D siguiendo el *next hop* óptimo.

2. Conclusiones

El avance presentado cumple con los objetivos intermedios del laboratorio:

- Se cuenta con un módulo de Dijkstra funcional e integrado en el ciclo de ruteo.
- La infraestructura de comunicación es capaz de manejar varios hilos para separar funciones de *forwarding* y *routing*.
- La simulación muestra la entrega correcta de un mensaje entre nodos no directamente conectados.

3. Trabajo futuro

- Implementar flooding con control de duplicados.
- Agregar soporte para *Distance Vector Routing*.
- Incluir manejo de fallos y cambios dinámicos en la topología.
- Medición de métricas y registro de estadísticas.