Laboratorio 3 - Parte 1: Algoritmos de ruteo

Brandon Reyes - 22992
Santiago Pereira - 22318
Nancy Mazariegos - 22513

Reporte de la Práctica

En esta práctica se implementaron y compararon distintos algoritmos de enrutamiento en redes de computadoras. Se utilizó una topología definida en un archivo JSON y se simularon nodos que se comunican entre sí a través de sockets en Python. El objetivo fue observar cómo los algoritmos Flooding y Distance Vector transmiten mensajes, gestionan la información de rutas y reaccionan ante la propagación de datos en la red.

Se implementaron dos algoritmos principales:

1. ***Flooding:***
   Cada nodo envía los mensajes recibidos a todos sus vecinos excepto al que se los envió, repitiendo este proceso hasta que el TTL se agota o el mensaje llega a su destino. En el código se manejó la detección de duplicados mediante un identificador único por mensaje y un conjunto de control. Se añadieron cabeceras con información como ID, timestamp y origen, y se aplicó un TTL decreciente en cada reenvío.

2. ***Distance Vector Routing (DVR):***
   Cada nodo mantiene una tabla de enrutamiento con la distancia estimada a todos los demás nodos y el siguiente salto correspondiente. Las tablas se actualizan intercambiando periódicamente vectores de distancias con los vecinos. Si un nodo detecta una mejor ruta, actualiza su tabla. Se implementó la actualización periódica, la propagación de tablas, la detección de destinos inalcanzables y el reenvío inteligente de mensajes solo hacia el siguiente salto.

Ambos algoritmos se desarrollaron utilizando Python, sockets para la comunicación entre nodos, hilos para concurrencia y estructuras de datos como diccionarios y conjuntos para el control de mensajes y tablas.

Resultados

Durante las pruebas, el algoritmo Flooding logró entregar los mensajes pero generó un número alto de transmisiones redundantes, ya que todos los nodos reenviaban los paquetes hasta agotar el TTL. En contraste, el algoritmo Distance Vector, después de un tiempo de convergencia, permitió el envío de mensajes de manera más eficiente, con menos redundancia y con rutas definidas hacia el destino.
Se observaron tablas de enrutamiento dinámicas que se ajustaban conforme se recibían vectores de los vecinos, mostrando el aprendizaje gradual de la red.

Discusión

El algoritmo Flooding garantiza que un mensaje llegue al destino si existe alguna ruta, pero no es eficiente debido a la duplicación y al consumo de ancho de banda. Es útil en redes pequeñas o cuando no se conoce la topología.

El algoritmo Distance Vector requiere más tiempo para estabilizarse, pero una vez alcanzada la convergencia es más eficiente, ya que permite la transmisión de mensajes siguiendo rutas específicas y evitando reenvíos innecesarios. Sin embargo, DVR puede sufrir del problema de conteo hasta el infinito si no se aplican técnicas adicionales.

La comparación muestra la diferencia entre un enfoque ingenuo (Flooding) y uno más inteligente basado en tablas (DVR), destacando la importancia de los algoritmos de enrutamiento en redes modernas.

Conclusiones y comentarios

Se concluye que Flooding asegura la entrega pero con un alto costo en redundancia, mientras que Distance Vector proporciona mayor eficiencia y escalabilidad en redes de tamaño medio y grande. Ambos algoritmos ilustran conceptos clave en el enrutamiento y la propagación de información en redes distribuidas.

La práctica permitió comprender de forma aplicada el funcionamiento de protocolos de enrutamiento y la importancia de estructuras como tablas de enrutamiento y control de duplicados.

Referencias

- Kurose, J., & Ross, K. (2021). Computer Networking: A Top-Down Approach. Pearson.
- Tanenbaum, A., & Wetherall, D. (2011). Computer Networks. Pearson.
- Documentación oficial de Python (https://docs.python.org/3/library/socket.html)