Universidad del Valle de Guatemala
Facultad de Ingeniería
Departamento de Ciencias de la Computación
CC3067 Redes

Laboratorio 3 - Primera Parte

Algoritmos de Enrutamiento

1 Antecedentes

Conociendo a dónde enviar los mensajes para cualquier router se vuelve trivial el envío de mensajes. Únicamente es necesario conocer el destino final y se reenvía al vecino que puede proveer la mejor ruta al destino. Toda esa información es almacenada en las tablas de enrutamiento.

No obstante, con el dinamismo con el que se espera que pueda funcionar el Internet es necesario que dichas tablas puedan actualizarse y acomodarse a cambios en la infraestructura. Los algoritmos con los que se actualizan estas tablas son conocidos como algoritmos de enrutamiento.

2 Objetivos

- Conocer los algoritmos de enrutamiento utilizados en las implementaciones actuales de Internet.
- Comprender cómo funcionan las tablas de enrutamiento.
- Implementar los algoritmos de enrutamiento y probarlos.

3 Desarrollo

Los algoritmos de enrutamiento funcionan sobre nodos interconectados entre sí, donde cada nodo conoce únicamente cuáles son los vecinos que tiene. Dicha información inicial será proporcionada para cada nodo. A partir de ello se simulará de forma manual el envío y recepción de los mensajes que los nodos realizan entre sí para conformar sus tablas de enrutamiento.

Por ahora, nos enfocaremos en los algoritmos de ruteo de forma “offline” (simulando, hard-coding, o alimentandole manualmente los inputs requeridos).

Para tales efectos se conformarán grupos de un máximo de tres integrantes (o 4 en caso sea necesario) donde se implementarán los algoritmos según descrito más adelante.

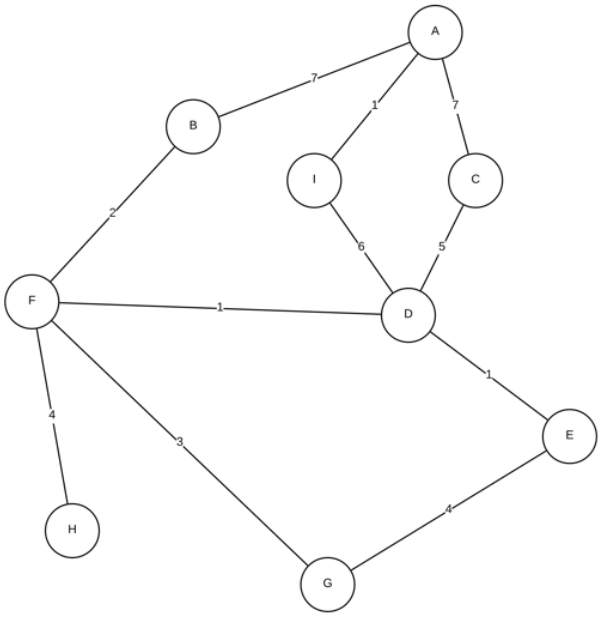


Imagen 1: Mapa de conexiones entre nodos.

En esta propuesta cada uno de los nodos corresponde a un cliente el cuál puede enviar o recibir mensajes para conformar su tabla de enrutamiento y posteriormente enviar un paquete de un nodo origen a un nodo destino utilizando dicha tabla.

3.1 Implementación de algoritmos

Cada uno de los integrantes del grupo debe de implementar al menos un algoritmo (el lenguaje de programación es libre, y puede ser el mismo para los tres algoritmos, pero es **altamente sugerido** que sea uno donde se tenga experiencia trabajando con el protocolo XMPP: revisar enunciado segunda parte). Los algoritmos para implementarse son (en negrita los obligatorios):

1. **Dijkstra**
2. **Flooding**
3. Distance vector routing
4. Link state routing

Las implementaciones deben de estar debidamente identificadas y publicadas en un repositorio privado (el día de la entrega lo hacen público), junto con sus requerimientos para su uso e instalación. Las implementaciones idealmente deben de poder ejecutarse en distintas plataformas de forma sencilla (makefiles, python venv, .jar).

Dependiendo del algoritmo, se deberá simular/proveer los inputs requeridos por cada uno para funcionar adecuadamente. En este laboratorio estaremos simulando o hard-coding tales valores; en la Parte 2 es que procederemos a recibir tales inputs de forma automática de los demás nodos. Lo que requiere cada algoritmo incluye:

- Dijkstra: Topología (nodos, aristas)
- Flooding: ninguno (a lo sumo la Topología).
- Distance Vector: Las Tablas de los Vecinos, Topología
- Link State Routing: Las Tablas de los demás Nodos (de ella se extrae la Topología)

Los nodos deben indicar cuándo están listos para transmitir.

3.2 Formato de Paquetes (y consideraciones para el Lab 3.2)

Todos los grupos deberán seguir el protocolo/formato estándar siguiente, pudiendo agregar elementos si así lo consideran (ojo, deben ponerse de acuerdo entre todos los grupos). La estructura será tipo JSON y será de la siguiente forma:

```
{
        “type” : ”message|echo|info”,
        “headers” : [“from” : ”foo”, “to” : ”bar”, “hop_count” : 3,  …],
        “payload” : “loremipsum\{lo que sea aca\}”
}
```

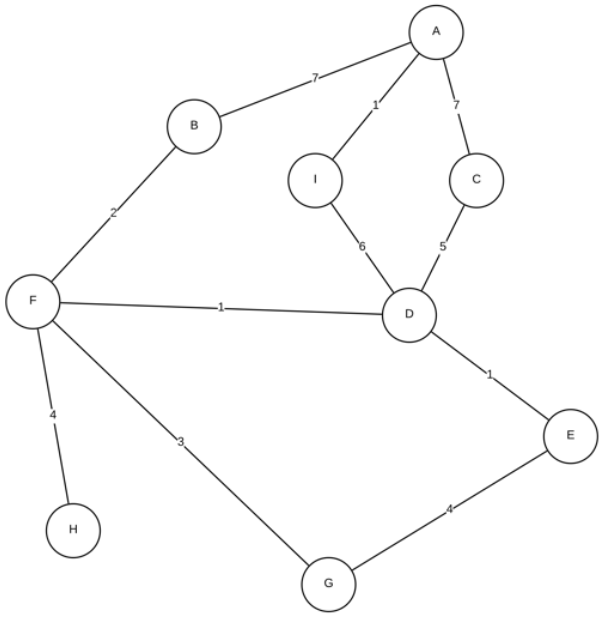
El elemento “type” indica el tipo de paquete (mensaje de usuario, paquete de info de tablas, paquete echo para descubrimiento, etc.). Los 3 tipos mínimos a definir e implementar son los indicados en el ejemplo. En “headers” van los encabezados del paquete.

Los encabezados (headers) mínimos a usar y definir son: from (nodo origen), to (nodo destino), hop count (conteo de saltos). Pueden definir otros como listado de nodos visitados, ‘distancia’ recorrida en milisegundos, etc.., siempre con el objetivo de facilitar la comunicación.

3.3 Ejemplo

Fase 1: Tabla de enrutamiento

Supongamos una implementación de Distance Vector. Inicialmente cada nodo deberá enviar a sus vecinos su vector de distancia.



El cliente en el nodo **A** devolverá como output los siguientes json (**la estructura de datos del payload queda a discreción y debe consensuarse entre toda la clase**):

```
{
        “type” : ”info”,
        “headers” : [“from” : ”A”, “to” : ”B”, “hop_count” : 3,  …],
        “payload”      :      “[0,1,1,0,0,0,0,0,1],      [0,0,0,0,0,0,0,0,0],      [0,0,0,0,0,0,0,0,0],
                              [0,0,0,0,0,0,0,0,0],                                  [0,0,0,0,0,0,0,0,0],
                              [0,0,0,0,0,0,0,0,0],[0,0,0,0,0,0,0,0,0],[0,0,0,0,0,0,0,0,0],[0,0,0,0,0,0,0
                              ,0,0]”
}
```

```
{
    “type” : ”info”,
    “headers” : [“from” : ”A”, “to” : ”C”, “hop_count” : 3,  …],
    “payload” : “[0,1,1,0,0,0,0,0,1], [0,0,0,0,0,0,0,0,0], [0,0,0,0,0,0,0,0,0],
                 [0,0,0,0,0,0,0,0,0], [0,0,0,0,0,0,0,0,0],
                 [0,0,0,0,0,0,0,0,0],[0,0,0,0,0,0,0,0,0],[0,0,0,0,0,0,0,0,0],[0,0,0,0,0,0,0,0
                 ,0,0]”
}
```

```
{
    “type” : ”info”,
    “headers” : [“from” : ”A”, “to” : ”I”, “hop_count” : 3,  …],
    “payload” : “[0,1,1,0,0,0,0,0,1], [0,0,0,0,0,0,0,0,0], [0,0,0,0,0,0,0,0,0],
                 [0,0,0,0,0,0,0,0,0], [0,0,0,0,0,0,0,0,0],
                 [0,0,0,0,0,0,0,0,0],[0,0,0,0,0,0,0,0,0],[0,0,0,0,0,0,0,0,0],[0,0,0,0,0,0,0,0
                 ,0,0]”
}
```

Proporcionaremos esos JSON devueltos a los nodos indicados como destino, quienes a su vez proporcionaran nuevos vectores de distancia al recibir actualizaciones. Este proceso manual continuará hasta que haya convergencia en todos los nodos (los vectores ya no cambian). En ese momento los nodos deben indicar que ya están listos para transmitir pues ya tienen toda la información necesaria. (Para DV se sugiere probar inicialmente con una topología pequeña)

Fase 2: envio de “paquetes”

En esta fase cada nodo puede transmitir hacia cualquier nodo. Supongamos un envío de A hacia F. La ruta más corta es A-B-F. El cliente en A consultará su tabla de enrutamiento y devolverá este output:

```
{
    “type” : ”message”,
    “headers” : [“from” : ”A”, “to” : ”F”, “hop_count” : 0,  …],
    “payload” : “Hola mundo”
}
```

Este JSON será proporcionado manualmente a **B** (ya que es la ruta óptima hacia F según la tabla) el cuál devolverá como output:

```
{
        “type” : ”message”,
        “headers” : [“from” : ”A”, “to” : ”F”, “hop_count” : 1,  …],
        “payload” : “Hola mundo”
}
```

Y este JSON será proporcionado a **F** el cuál deberá mostrar el mensaje “Hola mundo.”

4 Reporte

Al finalizar la actividad debe de realizarse un reporte grupal donde se incluyan las siguientes secciones:

- Nombres y carnés
- Título de la práctica
- Descripción de la práctica
    - Incluir explicación de los algoritmos utilizados
- Resultados
- Discusión
- Comentario grupal
- Conclusiones

4.1 Rúbrica de evaluación

Elemento		Ponderación	
Código		**75%**	
	Documentación, orden, comentarios, limpieza, legibilidad/funcionalidad balanceada, etc..		5%
	Implementación de los Algoritmos, de forma eficiente y optimizada. (Parejas: 15 c/u, Tríos: 10 c/u)		70%
Reporte Escrito		**25%**	

	Encabezado, Ortografía, Formato Adecuado, Descripción de la Práctica		2.5%
	Descripción de los Algoritmos Utilizados y su Implementación		10%
	Resultados		5%
	Discusión		5%
	Conclusiones + Comentarios + Referencias		2.5%

** Una inasistencia injustificada anula la nota del laboratorio.

Entregar en Canvas:

1. Archivo .pdf con su reporte en grupo
2. Código utilizado para el Laboratorio, y demás, en un zip/rar + link a repositorio