

Universidad del Valle de Guatemala  
Facultad de Ingeniería



Redes - Algoritmos de enrutamiento  
Interconexión de Nodos  
Informe  
Laboratorio 3.2

Mariana David 201055

Fredy Velasquez 201011

Angel Higueros 20460

Guatemala 12 de septiembre del 2023

## **Descripción de la práctica**

La práctica se enfoca en la implementación y análisis de algoritmos de enrutamiento en una red simulada utilizando el protocolo XMPP. En la primera parte del laboratorio, se desarrollaron y probaron los algoritmos de enrutamiento de forma estática, proporcionando manualmente la información necesaria para crear las tablas de enrutamiento y enviar paquetes de un nodo origen a un nodo destino.

En la segunda parte de la práctica, se simula una infraestructura de red utilizando el servicio de chat alumchat.xyz, donde cada nodo corresponde a un cliente con una dirección @alumchat.xyz. Los estudiantes utilizan sus identificadores de usuario "oficiales" de alumchat como JID y credenciales. Se establecen mapas de conexiones entre los nodos, y cada nodo debe ser capaz de enviar y recibir mensajes. Los algoritmos de enrutamiento a probar incluyen Flooding, Distance Vector y Link State Routing (con Link State Routing y Flooding como componentes).

Los nodos ejecutan procesos simultáneos de reenvío de mensajes y actualización de tablas de enrutamiento, utilizando mensajes específicos como paquetes ECHO para medir los retrasos entre nodos, paquetes DATA para transportar mensajes de usuario y paquetes TABLE/INFO para compartir información sobre tablas, rutas y vecinos. El objetivo principal es lograr la estabilidad de los algoritmos de enrutamiento, permitiendo que los mensajes fluyan adecuadamente a través de los nodos y se adapten a cambios en la red, como la incorporación de nuevos nodos o nodos caídos.

## **Explicación de los algoritmos utilizados**

- **Link State Routing**

Es un algoritmo que se utiliza para encontrar la ruta más corta entre dos nodos en un grafo ponderado. En el contexto de enrutamiento, se aplica para calcular las mejores rutas desde un nodo origen a todos los demás nodos en la red. Este algoritmo utiliza información sobre la distancia acumulada desde el nodo origen a cada nodo y selecciona la ruta con la menor distancia.

- **Flooding**

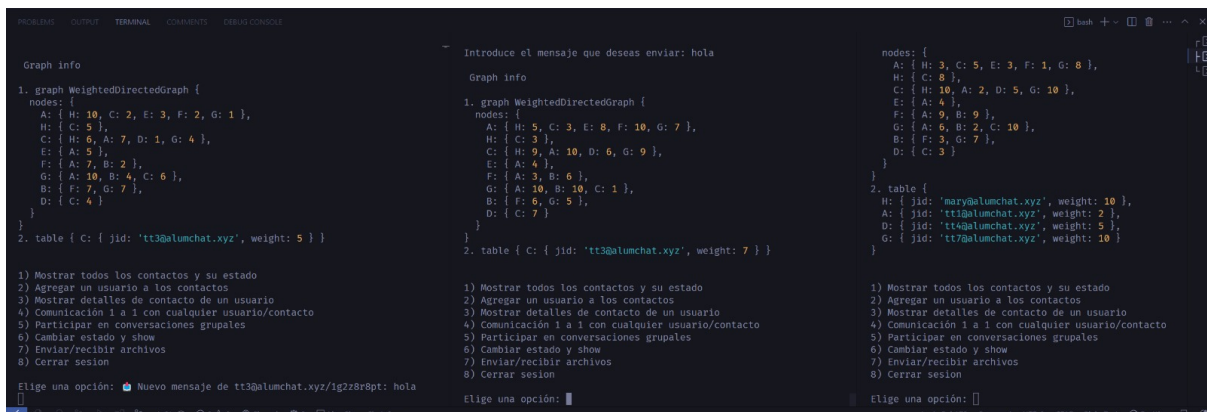
Flooding es un algoritmo simple de enrutamiento que se basa en el reenvío indiscriminado de paquetes a todos los nodos vecinos en la red. Cuando un nodo recibe un paquete, lo reenvía a todos sus vecinos, lo que resulta en una propagación

amplia de paquetes por toda la red. Si bien es un método ineficiente y puede generar tráfico innecesario, puede ser útil en situaciones donde no se tiene información sobre la topología de la red.

- Vector routing

Es un algoritmo de enrutamiento que se basa en mantener tablas de enrutamiento locales en cada nodo. Estas tablas contienen información sobre los nodos vecinos y las distancias a los destinos. Los nodos comparten sus tablas de enrutamiento con sus vecinos y actualizan sus rutas en función de la información recibida. Este enfoque permite a los nodos adaptarse a cambios en la red y calcular rutas óptimas.

## Resultados



```
Graph info
1. graph WeightedDirectedGraph {
  nodes: {
    A: { H: 10, C: 2, E: 3, F: 2, G: 1 },
    H: { C: 5 },
    C: { H: 6, A: 7, D: 1, G: 4 },
    E: { A: 5 },
    F: { A: 7, B: 2 },
    G: { A: 10, B: 4, C: 6 },
    B: { F: 7, G: 7 },
    D: { C: 4 }
  }
}
2. table { C: { jid: 'tt3@alumchat.xyz', weight: 5 } }

1) Mostrar todos los contactos y su estado
2) Agregar un usuario a los contactos
3) Mostrar detalles de contacto de un usuario
4) Comunicación 1 a 1 con cualquier usuario/contacto
5) Participar en conversaciones grupales
6) Cambiar estado y show
7) Enviar/recibir archivos
8) Cerrar sesion

Elige una opción: 4 Nuevo mensaje de tt3@alumchat.xyz/1g2r8r8pt: hola

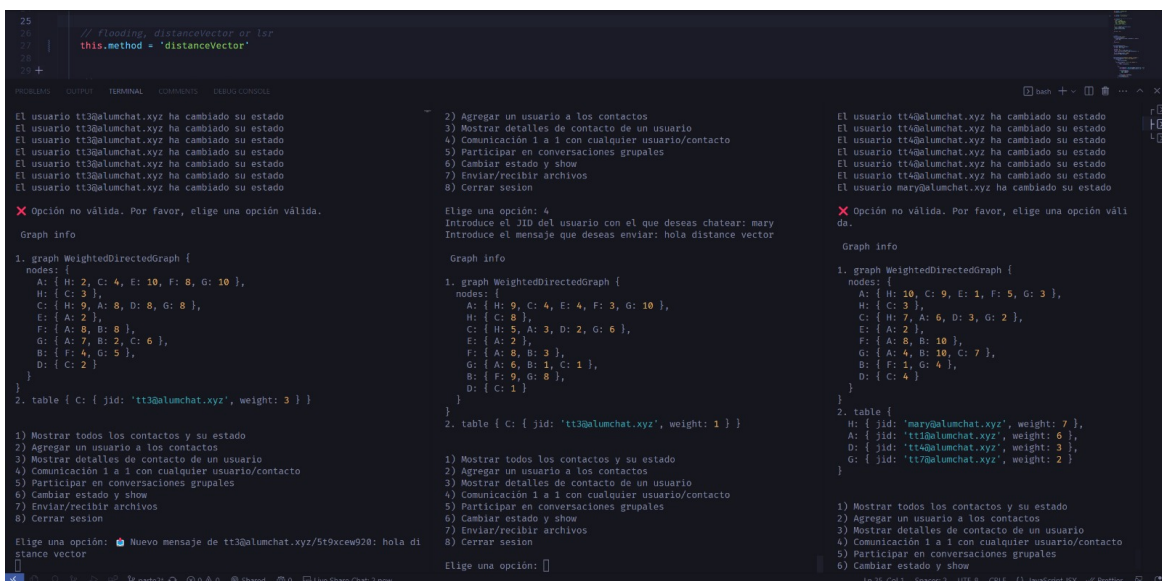
Introduce el mensaje que deseas enviar: hola

Graph info
1. graph WeightedDirectedGraph {
  nodes: {
    A: { H: 5, C: 3, E: 8, F: 10, G: 7 },
    H: { C: 3 },
    C: { H: 9, A: 10, D: 6, G: 9 },
    E: { A: 4 },
    F: { A: 3, B: 6 },
    G: { A: 10, B: 10, C: 1 },
    B: { F: 6, G: 5 },
    D: { C: 7 }
  }
}
2. table { C: { jid: 'tt3@alumchat.xyz', weight: 7 } }

1) Mostrar todos los contactos y su estado
2) Agregar un usuario a los contactos
3) Mostrar detalles de contacto de un usuario
4) Comunicación 1 a 1 con cualquier usuario/contacto
5) Participar en conversaciones grupales
6) Cambiar estado y show
7) Enviar/recibir archivos
8) Cerrar sesion

Elige una opción: 4
```

Figura 1. Prueba de funcionamiento utilizando flooding



```
25 // flooding: distancevector on lar
26 this.method = 'distanceVector'
27
28
29 +

El usuario tt3@alumchat.xyz ha cambiado su estado
El usuario tt3@alumchat.xyz ha cambiado su estado
El usuario tt3@alumchat.xyz ha cambiado su estado
El usuario tt3@alumchat.xyz ha cambiado su estado
El usuario tt3@alumchat.xyz ha cambiado su estado
El usuario tt3@alumchat.xyz ha cambiado su estado
El usuario tt3@alumchat.xyz ha cambiado su estado
El usuario tt3@alumchat.xyz ha cambiado su estado

X Opción no válida. Por favor, elige una opción válida.

Graph info
1. graph WeightedDirectedGraph {
  nodes: {
    A: { H: 2, C: 4, E: 10, F: 8, G: 10 },
    H: { C: 3 },
    C: { H: 9, A: 8, D: 8, G: 8 },
    E: { A: 2 },
    F: { A: 8, B: 8 },
    G: { A: 7, B: 2, C: 6 },
    B: { F: 4, G: 5 },
    D: { C: 2 }
  }
}
2. table { C: { jid: 'tt3@alumchat.xyz', weight: 3 } }

1) Mostrar todos los contactos y su estado
2) Agregar un usuario a los contactos
3) Mostrar detalles de contacto de un usuario
4) Comunicación 1 a 1 con cualquier usuario/contacto
5) Participar en conversaciones grupales
6) Cambiar estado y show
7) Enviar/recibir archivos
8) Cerrar sesion

Elige una opción: 4 Nuevo mensaje de tt3@alumchat.xyz/5t9xcedw20: hola distance vector

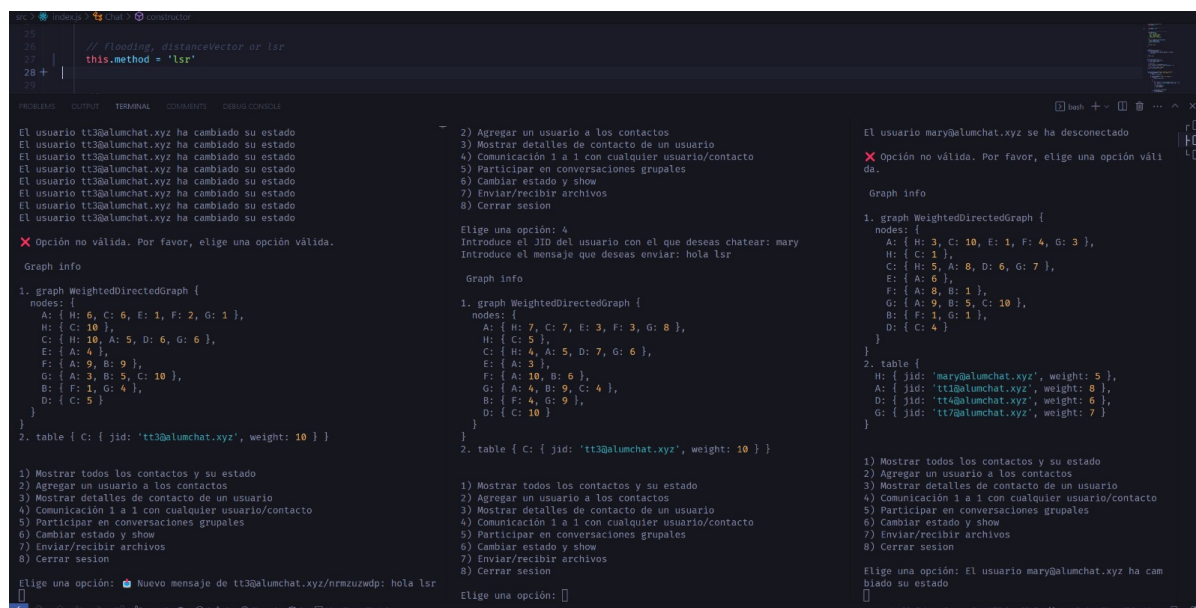
Introduce el jid del usuario con el que deseas chatear: mary
Introduce el mensaje que deseas enviar: hola distance vector

Graph info
1. graph WeightedDirectedGraph {
  nodes: {
    A: { H: 9, C: 4, E: 4, F: 3, G: 10 },
    H: { C: 8 },
    C: { H: 5, A: 3, D: 2, G: 6 },
    E: { A: 2 },
    F: { A: 8, B: 3 },
    G: { A: 6, B: 1, C: 1 },
    B: { F: 9, G: 8 },
    D: { C: 1 }
  }
}
2. table { C: { jid: 'tt3@alumchat.xyz', weight: 1 } }

1) Mostrar todos los contactos y su estado
2) Agregar un usuario a los contactos
3) Mostrar detalles de contacto de un usuario
4) Comunicación 1 a 1 con cualquier usuario/contacto
5) Participar en conversaciones grupales
6) Cambiar estado y show
7) Enviar/recibir archivos
8) Cerrar sesion

Elige una opción: 4
```

Figura 2. Prueba de funcionamiento utilizando distance vector



**Figura 3. Prueba de funcionamiento utilizando Link State Routing**

## Discusión

En el transcurso de este laboratorio, nos pudimos adentrar en el mundo de los algoritmos de enrutamiento, explorando su aplicabilidad en el contexto de una red simulada basada en el protocolo XMPP. Durante esta experiencia, se implementaron tres algoritmos clave: Link State Routing, Flooding y Vector Routing, con el objetivo de establecer rutas eficientes para la comunicación en un chat en línea. A medida que avanzamos en nuestra discusión, explicaremos los resultados obtenidos, analizando cuestiones cruciales como la velocidad, la eficiencia y la adaptabilidad de estos algoritmos en el entorno dinámico de un chat.

Como primera instancia, en cuanto a la velocidad y eficiencia, pudimos notar algunas diferencias entre los algoritmos implementados. El algoritmo de Link State Routing demostró ser el más rápido en calcular rutas óptimas en comparación con Flooding y Vector Routing. Esto se debe a su capacidad para determinar la ruta más corta utilizando una estrategia de búsqueda eficiente. Sin embargo, esta velocidad viene acompañada de un mayor consumo de recursos computacionales, lo que podría ser una preocupación en escenarios de red con un alto tráfico. Por otro lado, Flooding, resultó ser el más lento en términos de velocidad, ya que involucra un reenvío indiscriminado de paquetes a todos los nodos vecinos. Sin embargo, en un entorno de chat donde la topología de la red puede cambiar constantemente y no se dispone de información detallada, Flooding demostró ser una opción viable y eficaz, ya que garantiza que los mensajes alcancen su destino incluso en condiciones inciertas.

Por otra parte, en cuanto a la eficacia en un chat, Vector Routing se destacó por su capacidad para adaptarse a cambios en la red de manera eficiente. Esto resulta crucial en un chat en línea, donde los nodos pueden unirse o abandonar la red en cualquier momento. Vector Routing permitió a los nodos locales ajustar sus tablas de enrutamiento en función de la información actualizada de los nodos vecinos, lo cual nos logró garantizar que los mensajes continuaran fluyendo bastante bien, incluso en situaciones dinámicas. Aunque Link State Routing es rápido en calcular rutas óptimas, su eficacia puede verse comprometida cuando se producen cambios constantes en la topología de la red, ya que requiere actualizaciones frecuentes de las tablas de enrutamiento.

Por último, consideramos que la elección del algoritmo depende de las necesidades específicas de la red de chat. Si la prioridad es la velocidad y se cuenta con información fiable sobre la topología de la red, Link State Routing puede ser la mejor opción. Sin embargo, si la red es dinámica y se necesita adaptabilidad, Vector Routing y, en ciertas circunstancias, Flooding pueden ser opciones viables. Cada algoritmo tiene sus ventajas y desventajas, y su elección debe basarse en una cuidadosa consideración de los requisitos del sistema y las condiciones de la red en tiempo real.

### **Comentario grupal**

Nos sentimos bastante satisfechos con los resultados obtenidos en el laboratorio de implementación de algoritmos de enrutamiento en la red simulada con el protocolo XMPP. Esto, se debe porque durante esta experiencia, nos enfrentamos a diversos desafíos técnicos interesantes y logramos avances significativos en la comprensión y aplicación de estos algoritmos.

En lo que respecta a la implementación de los algoritmos, cada uno de los miembros del grupo pudo contribuir de manera significativa. El algoritmo de Link State Routing, a pesar de su complejidad, resultó ser una pieza clave para calcular rutas óptimas en tiempo real. Flooding, aunque menos eficiente, demostró su utilidad en situaciones donde la información de la topología de la red era limitada. Por su parte, el Vector Routing permitió adaptarse eficazmente a los cambios en la red y calcular rutas locales de manera efectiva.

La simulación de la infraestructura utilizando el chat alumchat.xyz fue sin duda uno de los momentos más emocionantes del laboratorio. Como grupo pudimos observar cómo los algoritmos previamente realizados, interactuaban con la red, permitiendo el envío de mensajes y el reajuste de rutas en función de las condiciones cambiantes. En general, esta experiencia ha sido valiosa para comprender mejor el funcionamiento de los algoritmos de enrutamiento y su aplicabilidad en escenarios prácticos.

### **Conclusiones**

En conclusión, el laboratorio de implementación de algoritmos de enrutamiento en el contexto de un chat basado en el protocolo XMPP nos proporcionó, como grupo, una visión profunda de la complejidad y versatilidad de estos algoritmos en un entorno práctico. Durante este proceso, se exploraron tres algoritmos fundamentales: Link State Routing, Flooding y Vector Routing, cada uno con sus propias ventajas y desafíos.

Observamos que Link State Routing se destacó por su velocidad en el cálculo de rutas óptimas, pero requirió una gestión meticulosa de las actualizaciones de la tabla de enrutamiento en un entorno dinámico. Flooding demostró ser una opción sólida en situaciones donde la topología de la red era incierta, garantizando la entrega de mensajes incluso en condiciones cambiantes, aunque a expensas de una mayor latencia. Por otro lado, Vector Routing se destacó por su adaptabilidad a los cambios en la red, lo que lo convierte en una elección valiosa para entornos en línea donde la flexibilidad es esencial.

En última instancia, la elección del algoritmo de enrutamiento depende de las necesidades específicas de la red y las condiciones en tiempo real. La velocidad, la eficiencia y la capacidad de adaptación son factores críticos que deben sopesarse cuidadosamente. Este laboratorio no solo amplió nuestro conocimiento sobre los algoritmos de enrutamiento, sino que también nos enseñó la importancia de la selección estratégica de estos algoritmos para garantizar un rendimiento óptimo en una red dinámica como un chat en línea.

### **Anexo:**

<https://github.com/angelhigueros/redes-lab3-1/tree/parte2>