Universidad del Valle de Guatemala

Redes

Sección 20

Laboratorio #3

Diego Alejandro Morales Escobar - 21146

Javier Alejandro Azurdia Arrecis - 21242

Angel Sebastían Castellanos Pineda - 21700

Descripción de los Algoritmos Utilizados

Flooding

El algoritmo de flooding es una técnica de enrutamiento simple en la que cada paquete entrante se envía por todas las interfaces de salida, excepto por la que se recibió. Esto garantiza que el paquete llegue a su destino si es posible, pero puede generar tráfico redundante y consumir ancho de banda innecesariamente.

Características principales:

1. Simplicidad: Es fácil de implementar ya que no requiere información de estado del enlace ni cálculos complejos de ruta.
2. Redundancia: Debido a que los paquetes se envían por todas las rutas posibles, se asegura la entrega, pero también puede causar duplicación de paquetes y congestión en la red.
3. Uso en redes específicas: Se utiliza en redes ad-hoc, protocolos de enrutamiento como OSPF y DVMRP, y en situaciones donde la dirección de destino es desconocida.

(Saxena, 2023).

Se realizó una implementación del algoritmo de flooding para la propagación de mensajes en un servidor XMPP utilizando JavaScript. Utiliza la biblioteca Xmpp.js para establecer una conexión con un servidor XMPP y gestionar el envío y recepción de mensajes.

El servidor recibe un mensaje de un usuario específico y verifica si el mensaje está destinado al usuario conectado. Si el mensaje no está dirigido al usuario, incrementa un contador de saltos (hops) y lo reenvía a todos los contactos del nodo, excepto al remitente original, replicando el proceso de "flooding". De esta manera, el mensaje se propaga por toda la red hasta llegar al destinatario final.

El código también maneja la conexión inicial al servidor, la autenticación del usuario, la recepción de mensajes y el cierre de la sesión del cliente XMPP de manera asíncrona. Además, proporciona un punto de entrada (main) que inicializa la conexión, envía un mensaje y activa el proceso de flooding.

LSR

El algoritmo LSR es una técnica utilizada en redes de computadoras para determinar la mejor ruta entre dos puntos. A diferencia del enrutamiento por vector de distancia, este algoritmo permite que cada router tenga una visión completa de la topología de la red.

Características Principales:

1. Conocimiento del Vecindario: Cada router recopila información sobre sus enlaces directos y la comparte con todos los demás routers en la red.

2. Inundación de Información: La información sobre el estado de los enlaces se envía a todos los routers mediante un proceso llamado inundación. Esto asegura que todos los routers tengan la misma información sobre la red.
3. Base de Datos de Estado de Enlace: Cada router mantiene una “base de datos” que contiene el estado de todos los enlaces en la red.
4. Cálculo de rutas: Utilizando el algoritmo de Dijkstra, cada router calcula la ruta más corta a cada destino en la red.

(Kumar, 2023)

No se pudo completar la implementación del algoritmo de Link State Routing (LSR) debido a problemas de sincronización. Estos problemas causaron que algunos contactos enviaran respuestas antes de que el cliente tuviera la oportunidad de recibir y procesar los contactos de los vecinos. Esta desincronización generó inconsistencias en la información recopilada sobre los nodos de la red, impidiendo la correcta ejecución del algoritmo.

El algoritmo de Link State Routing (LSR) implementado en JavaScript busca determinar el estado de los enlaces en una red utilizando el protocolo XMPP. La idea es que cada nodo (cliente XMPP) recopile información sobre sus vecinos y comparta esa información con otros nodos, permitiendo así que cada nodo construya un mapa completo de la red.

El diseño planteado del algoritmo LSR es el siguiente:

1. A través de una estrategia muy parecida a la de flooding puro (*al menos en teoría porque en la práctica implica muchos más retos, sobre todo de sincronización*), se inunda la red con peticiones de información sobre los vecinos de cada uno de los nodos.
2. Al tener todas las respuestas, se construye una tabla con la siguiente estructura:

   {

   nodo1: [vecino1, costo], [vecino2, costo], [vecino3, costo]...,

   nodo2: [vecino1, costo], [vecino2, costo], [vecino3, costo]...,

   …

   }

   Es importante notar que vecino1 y nodo2 pueden ser el mismo router. Por medio de esta tabla, podemos construir un grafo ponderado que se utilizará a continuación.
3. Con el grafo ponderado y utilizando el algoritmo de Dikjstra, se elige el camino con menor costo desde un nodo hacia otro (no vecino).
4. Se guarda en el mensaje la estructura del camino que debe seguir el mensaje y se envía desde el router inicial.
5. Al enviar este mensaje desde el router de inicio, pasa por todos los routers intermedios hasta llegar al destinatario, usando la ruta menos costosa.

Resultados

Flooding

En esta implementación, hay dos clientes involucrados: cas@alumchat.lol *(Cliente 1)* y mor21146@alumchat.lol *(Cliente 2)*. Ambos clientes tienen al otro como contacto, y se probó el algoritmo de flooding para la propagación de mensajes entre ellos. El objetivo del algoritmo de flooding es propagar mensajes a través de la red, asegurando que todos los nodos reciban el mensaje hasta que llegue al destinatario final.

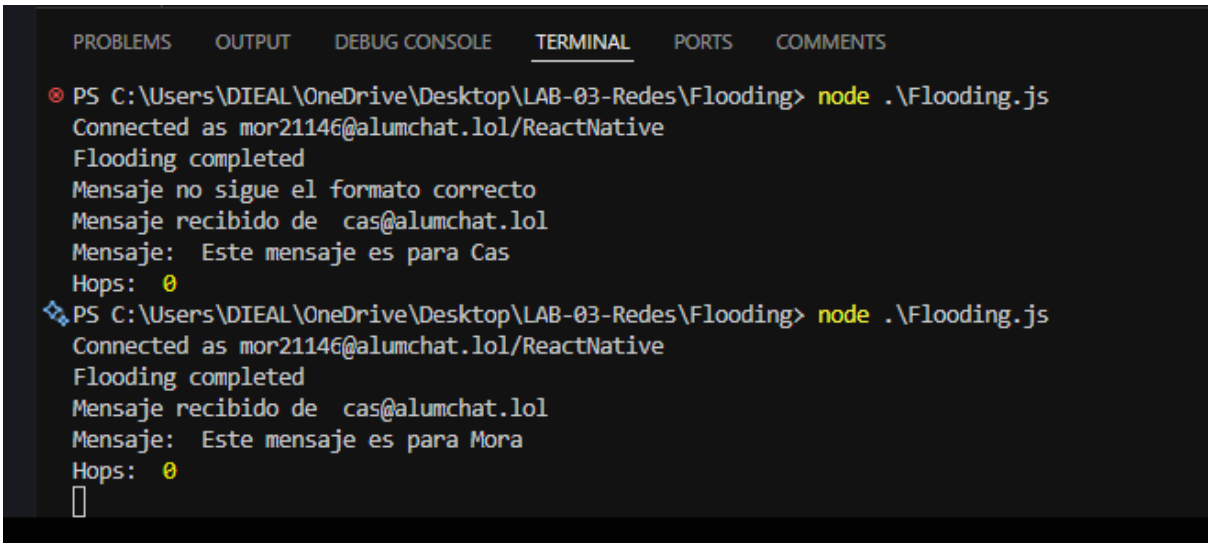
Resultados del Cliente 1 (cas@alumchat.lol):

1. Primera Ejecución:
    a. El cliente cas@alumchat.lol se conecta exitosamente al servidor XMPP.
    b. Se envía un mensaje a mor21146@alumchat.lol con el contenido: "Este mensaje es para Mora".
    c. El flooding se completa exitosamente, indicando que el mensaje se ha propagado a todos los contactos de cas@alumchat.lol.
2. Segunda Ejecución:
    a. El cliente se vuelve a conectar al servidor XMPP.
    b. El proceso de flooding se completa, pero no se recibe un nuevo mensaje en esta instancia.

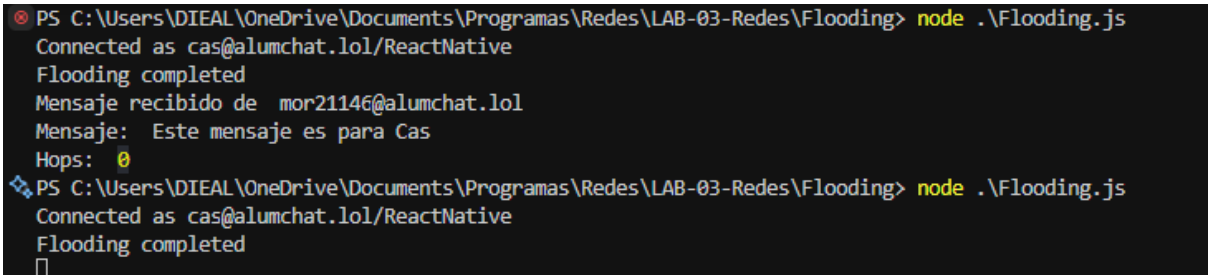
Resultados del Cliente 2 (mor21146@alumchat.lol):

1. Primera Ejecución:
    a. El cliente mor21146@alumchat.lol se conecta al servidor XMPP.
    b. El cliente recibe un mensaje de cas@alumchat.lol. Sin embargo, indica que el formato del mensaje recibido es incorrecto, lo que sugiere un posible problema en el análisis del mensaje.
    c. Posteriormente, se recibe un mensaje con el formato correcto de cas@alumchat.lol con el contenido: "Este mensaje es para Cas".
    d. El número de saltos (hops) es 0, lo que indica que el mensaje fue recibido directamente sin intermediarios.
2. Segunda Ejecución:
    a. El cliente mor21146@alumchat.lol se vuelve a conectar al servidor XMPP.
    b. Se recibe otro mensaje de cas@alumchat.lol con el contenido: "Este mensaje es para Mora".
    c. El número de saltos (hops) sigue siendo 0, mostrando que el mensaje se entregó directamente nuevamente.

Conexión con mor21146@alumchat.lol



Conexión con cas@alumchat.lol



LSR

No fue posible obtener resultados concluyentes del algoritmo de Link State Routing (LSR) debido a los problemas de sincronización identificados durante la ejecución. La desincronización ocasionó que algunos nodos enviaran respuestas antes de que el cliente pudiera recibir y procesar adecuadamente la información de los contactos de otros vecinos, lo que resultó en datos incompletos e inconsistentes sobre la topología de la red. Estos inconvenientes impidieron la construcción precisa del mapa de la red y, por lo tanto, el análisis y evaluación correctos del algoritmo.

Discusión

En los experimentos realizados con los algoritmos de Flooding y Link State Routing (LSR) utilizando el protocolo XMPP, se observaron diversos resultados que subrayan las fortalezas y las limitaciones de cada técnica. El algoritmo de Flooding mostró ser efectivo en la propagación de mensajes entre nodos dentro de una red pequeña, cumpliendo con su objetivo de distribuir mensajes hasta el destinatario final sin bucles infinitos. Sin embargo, se identificaron limitaciones significativas en cuanto a su escalabilidad, ya que, en redes de mayor tamaño, el volumen de mensajes enviados se incrementa exponencialmente, lo que podría generar una sobrecarga de tráfico en la red y consumo excesivo de recursos. Este resultado sugiere que, si bien el Flooding es útil para situaciones en las que se necesita una rápida difusión de mensajes en redes pequeñas, su uso en redes extensas requeriría mejoras adicionales, como la implementación de mecanismos de control para limitar la redundancia de mensajes.

En contraste, el algoritmo de LSR se encontró con problemas técnicos que impidieron obtener resultados concluyentes. La principal dificultad fue la desincronización en el intercambio de información de topología entre nodos. Durante la ejecución, se observó que algunos nodos enviaban respuestas antes de recibir completamente la información de los vecinos, lo que resultó en datos incompletos e inconsistentes sobre la estructura de la red. Esta situación dificultó la construcción precisa del mapa de la red, una etapa fundamental para el correcto funcionamiento del algoritmo de LSR. En teoría, LSR puede superar las limitaciones en cuanto a escalabilidad de flooding. Si bien, para redes pequeñas el consumo de recursos siempre será mayor, se evidencia su efectividad en redes más grandes al eliminar las redundancias.

Ambos algoritmos mostraron tener potencial para la propagación de mensajes y la construcción de topologías de red, pero también dejaron claro que su implementación requiere ajustes específicos para ser efectiva en entornos más complejos.

Conclusiones

- El algoritmo de Flooding se implementó con éxito, pero su eficiencia es limitada en redes grandes.
- La implementación de LSR no logró resultados concluyentes debido a problemas de sincronización en el intercambio de datos.
- Es necesario mejorar el control del flujo de mensajes en LSR para garantizar la precisión de la topología de red.

Comentario

La actividad, aunque nos pareció interesante, se asignó en un momento del semestre en el que estamos muy cargados de trabajos y proyectos importantes y pesados para el semestre, lo que dificultó dedicarle el tiempo necesario a la investigación e implementación de cada algoritmo de forma correcta. Con menos carga, sentimos que hubiéramos podido realizar este laboratorio sin problemas y no hubiéramos tenido que “sacrificar” el tiempo para este laboratorio en realizar otras actividades de la universidad. Se le dedicó todo el tiempo posible con el que contamos, pero lamentablemente no se pudo obtener el resultado deseado.

Hubiese sido de mucha ayuda haber observado durante la clase la implementación a nivel de pseudocódigo de LSR y de flooding, ya que conceptualizar algo tan abstracto representó una dificultad que no se pudo superar con el tiempo disponible.

Referencias

Kumar, S. (2023, January 31). *Link State Routing Algorithm*. PrepBytes. Retrieved August 29, 2024, from https://www.prepbytes.com/blog/miscellaneous/link-state-routing-algorithm/

Saxena, A. (2023, May 3). *Flooding in Computer Networks*. Scaler. Retrieved August 29, 2024, from https://www.scaler.com/topics/flooding-in-computer-networks/