ST0255 TELEMÁTICA
PROYECTO N1
Programación en Red

Fecha de publicación: septiembre 9 de 2024
Última actualización: septiembre 9 de 2024.
Fecha de Entrega: octubre 9 de 2024.

1. Objetivo

El objetivo principal de este proyecto es que los estudiantes desarrollen un servidor DHCP funcional en C, que pueda manejar solicitudes de cualquier cliente que se conecte a la red. El cliente, que puede ser programado en cualquier lenguaje, debe ser capaz de solicitar una dirección IP al servidor y recibirla correctamente.

2. Introducción

El objetivo de este proyecto es desarrollar las competencias necesarias en el diseño y desarrollo de aplicaciones concurrentes en red. Para lograr esto, se empleará la API Sockets con el fin de escribir un protocolo de comunicaciones que permita a una aplicación cliente comunicarse con una aplicación servidor.

En este trabajo se requiere implementar un servidor DHCP así como el cliente. El servidor debe ser capaz de entregar parámetros IPv4 a una red local o remota, entre los parámetros que proporcionaría a los clientes se deben tener como mínimo Dirección IP, mascara de subred, default Gateway, DNS server y dominio. En la implementación se debe determinar el respectivo protocolo de capa de transporte.

3. Antecedentes

3.1. Sockets

Tradicionalmente, a los desarrolladores de aplicaciones, las APIs de desarrollo de los diferentes lenguajes de programación, les realizan abstracciones que les ocultan los detalles de implementación de muchos aspectos, y el proceso de transmisión de datos no es la excepción.

Un socket es una abstracción a través de la cual las aplicaciones pueden enviar y recibir datos. Al respecto, el socket se puede entender como el mecanismo que utilizan las aplicaciones para “conectarse” a la red, específicamente con la arquitectura de red o “stack” de protocolos y de esta forma comunicarse con otras aplicaciones que también se encuentran “conectadas” a la red. De esta forma, la aplicación que se ejecuta en una máquina escribe los datos que dese transmitir en el socket y estos datos los puede leer otra aplicación que se ejecuta en otra máquina.

3.2. Tipos de Sockets

Existen diferentes tipos de sockets. Para este proyecto vamos a utilizar la API Sockets Berkeley, específicamente los sockets tipo “Stream (SOCK_STREAM)” o “Datagram (SOCK_DGRAM)”. En el marco de este proyecto, acorde al protocolo que está implementando usted debe decidir qué tipo de socket va a emplear y justificar a la luz de los requerimientos de su aplicación.

3.3. DHCP

El protocolo DHCP se desarrollo a partir del protocolo BOOTP, permitiendo tener una evolución en la asignación dinámica y automática de parámetros IP a los clientes de una red. El DHCP le permite al administrador supervisar y distribuir de forma centralizada las direcciones IP necesarias y, automáticamente, asignar y enviar una nueva IP si fuera el caso en que el dispositivo es conectado en un lugar diferente de la red.

Las especificaciones de este servicio de aplicación se definen en las RFC 1531, 2131 y 1541. Adicionalmente, se desarrolló DHCPv6 para direccionamiento IPv6, sin embargo, esta versión no se trabaja en este proyecto. En la figura 1 se puede observar la arquitectura a implementar en la cual se observa la aplicación cliente, así como la aplicación servidor. La solución por implementar debe contar con los siguientes componentes:

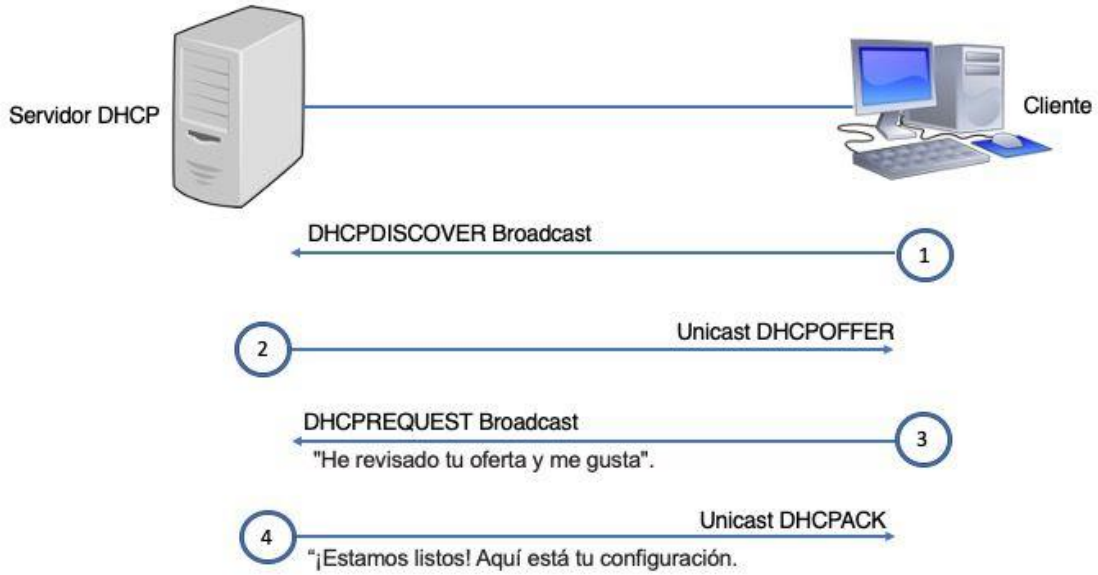

Figura 1. Arquitectura del servicio DHCP

4. Requerimientos.

Se requiere que usted implemente una versión del servicio de asignación de IP. De esta forma se solicita que implemente tanto la aplicación cliente como servidor. De esta forma el cliente y el servidor se comunicarán a través de los mensajes definidos para el protocolo DHCP.

4.1. Cliente DHCP

- El cliente puede ser programado en cualquier lenguaje (Python, Java, C, etc.).
- Debe enviar una solicitud de dirección IP (mensaje DHCPDISCOVER) al servidor.
- El cliente debe recibir una dirección IP y mostrar la información proporcionada por el servidor (dirección IP, máscara de red, puerta de enlace predeterminada, servidor DNS, etc.).
- Debe ser capaz de solicitar la renovación de la dirección IP cuando sea necesario.
- El cliente debe liberar la dirección IP al terminar la ejecución.

4.2. Servidor DHCP

- El servidor debe ser capaz de escuchar solicitudes DHCP de clientes en una red local o remota.
- Debe asignar direcciones IP disponibles de manera dinámica a los clientes que lo soliciten.
- El servidor debe gestionar un rango de direcciones IP (definido por el usuario) para asignar a los clientes.
- El servidor debe gestionar correctamente la concesión (*lease*) de las direcciones IP, incluyendo la renovación y liberación de estas.
- Debe ser capaz de manejar múltiples solicitudes simultáneamente.
- El servidor debe registrar todas las asignaciones de direcciones IP y el tiempo de arrendamiento.
- Debe soportar las principales fases y tipos de mensajes del proceso DHCP: **DISCOVER**, **OFFER**, **REQUEST**, **ACK** (opcionalmente, puede incluir **NAK**, **DECLINE**, y **RELEASE**).
- El servidor debe atender solicitudes de clientes que se encuentren en subredes diferentes a la del servidor (uso de un DHCP relay).
- Correcto manejo de errores y condiciones especiales (por ejemplo, cuando no hay más direcciones IP disponibles).

Finalmente, tenga en cuenta que las aplicaciones en red son concurrentes. En este sentido, el servidor DHCP que se está implementando no es la excepción. De esta forma, se requiere que su servidor se capaz de soportar de manera concurrente múltiples peticiones DHCP de manera simultánea. Para efectos de esta práctica, se permite el uso de hilos para soportar la concurrencia del servidor.

5. Requisitos técnicos y uso de la Aplicación.

A continuación, se detallan algunos aspectos que se deben considerar para la realización de su proyecto.

- Los estudiantes deberán reforzar el conocimiento de DHCP investigando el funcionamiento básico del protocolo, sus mensajes y la estructura de los paquetes que se intercambian entre cliente y servidor.

- La aplicación cliente puede ser escrita en el lenguaje de preferencia del grupo que soporte la API sockets tal como: Python 3.X, Java, C, C++, C#, Go, etc.
- La aplicación servidor **SOLO** debe ser implementada en lenguaje C.
    - **Nota**: No se recibe ningún proyecto, en su totalidad, con la aplicación servidor desarrollada en un lenguaje diferente a C.
- No se permite utilizar (tanto para cliente como para servidor) ninguna clase existente o personalizada de sockets. Se debe utilizar la API de Berkeley.
- La aplicación servidor se debe desplegar y ejecutar en un servidor en nube. Para esto utilice la cuenta de AWS Academy.
- Para documentar el diseño e implementación de su solución, por favor utilice herramientas como UML o cualquier otra con el fin de ilustrar las funcionalidades, así como el funcionamiento de este.

6. Aspectos Para Considerar para el Desarrollo

- **Equipo de trabajo:** El proyecto debe ser desarrollado en grupos de 3 personas como máximo. **NO** debe ser desarrollado de manera individual.
- **Cronograma**: Defina un plan de desarrollo, hitos, victorias tempranas, hasta la finalización del proyecto.
    - Tenga presente que tiene un mes calendario para desarrollar el proyecto. Se espera que el 60% del tiempo lo invierta en el análisis, diseño, implementación, pruebas y documentación para la aplicación servidor. El otro 40% restante, para la aplicación cliente.
- **Punto de Chequeo 1:**
    - **23 – 27 de septiembre** para esta fecha, debería tener la aplicación Servidor totalmente implementada.
- **Tiempo**: Empiece a trabajar apenas le publiquen su enunciado.
- **Consultas/dudas**: Si tiene alguna duda, por favor, acuda a su profesor y/o coordinador de la asignatura lo más pronto posible.
- **Cliente o Servidor:** Se sugiere que empiece por desarrollar la aplicación servidor y luego continue con la aplicación cliente

7. Entrega

- **Repositorio:** Trabaje con git. Cree un repositorio privado para su proyecto. Recuerde que usted no puede compartir el código de su trabajo con nadie.
- **Documentación:** La documentación se debe incluir en el repo en un archivo README.md. En este archivo se requiere que usted incluya los detalles de implementación donde como mínimo se esperan las siguientes secciones:
    - Introducción.
    - Desarrollo
    - Aspectos Logrados y No logrados.
    - Conclusiones
    - Referencias

- **Video**: Entregue y sustente al profesor mediante un video creado por usted, donde explique el proceso de diseño, desarrollo y ejecución (no más de 15 mins). Posteriormente se le citará a una sustentación presencial. Se debe ser claro en el video el funcionamiento de la solución, tanto para el cliente como para el servidor. Debe proveer instrucciones claras para compilar y ejecutar tanto el servidor como el cliente.
- **Fecha de entrega:** octubre 9 de 2024.
- **Mecanismo de entrega:** Por el buzón de Interactiva virtual se debe realizar la entrega. La entrega debe incluir un enlace a un repositorio en github. A partir de esta fecha y hora, no se puede realizar ningún commit al repositorio.

8. Evaluación

- Se realizará un proceso de sustentación para la verificación de la práctica acorde a lo entregado por el buzón y con lo explicado inicialmente en su video.
- Funcionamiento del servidor DHCP: capacidad para asignar IPs correctamente.
- Interacción del cliente con el servidor: correcto envío y recepción de mensajes DHCP.
- Documentación clara y detallada.

9. Referencias

- https://beej.us/guide/bgnet/
- https://beej.us/guide/bgc/
- https://www.geeksforgeeks.org/tcp-server-client-implementation-in-c/
- https://datatracker.ietf.org/doc/html/rfc2131
- https://datatracker.ietf.org/doc/rfc3456/