Proyecto 1 - TCP/UDP

Email SMTP/IMAP (TCP) y Comunicaciones Distribuidas con UDP

Ciencias de la Computación VIII

Resumen

Diseñar e implementar un sistema de correo electrónico que permita el envío y consulta de correos utilizando implementaciones propias de los protocolos de capa de aplicación SMTP (RFC 5321) e IMAP (RFC 3501) que utilizan en la capa de transporte TCP desarrolladas en Java, interactuando con el cliente de Mozilla Thunderbird, y permitiendo la comunicación entre servidores de dominios diferentes a través de UDP.

Componentes del sistema:

- Cliente/Servidor SMTP (Java)
- Servidor IMAP (Java)
- Base de datos o estructura de buzones locales
- Servidor de dominio remoto UDP (para forwarding de correos a otros proyectos)
- Cliente Thunderbird (como front-end gráfico)
- Logs o monitor de tráfico para análisis didáctico

Introducción

En este proyecto se busca abrir la mente para ver cómo funciona la teoría del protocolo poniéndolo en práctica frente a un cliente comercial funcional, esto ayudará a visualizar e implementar de cómo los protocolos de internet se utilizan en aplicaciones reales. Además, fomenta la lógica distribuida, el pensamiento sistémico y el análisis de protocolos mediante la observación directa del tráfico entre componentes.

Para esta implementación vamos a utilizar el programa **Thunderbird** que existe aplicación para Mac, Windows y Linux, el cual lo utilizaremos como cliente SMTP e IMAP.
Descargar: https://www.thunderbird.net/

Implementación

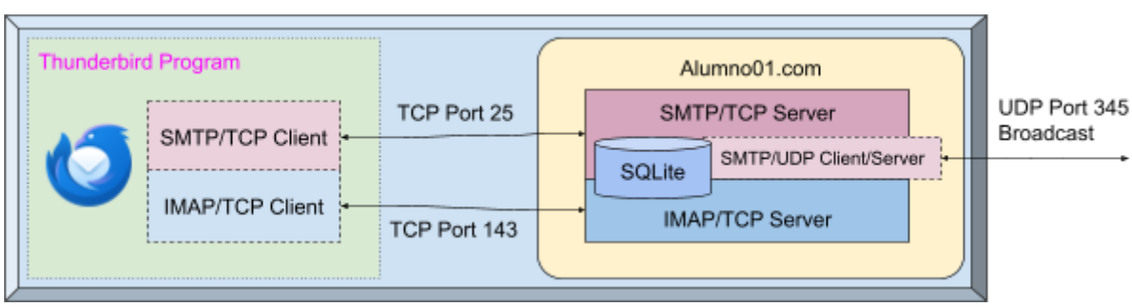
Para apoyar el desarrollo técnico de este proyecto, pueden reutilizar y adaptar conceptos y componentes previamente trabajados en el Laboratorio 1 y 2. Particularmente, pueden tomar como referencia:

- La estructura y uso de un Makefile para compilar y ejecutar los distintos módulos del servidor (SMTP, IMAP, UDP).
- La implementación de un ThreadPool para manejar múltiples conexiones de clientes de forma concurrente, especialmente en los servidores SMTP e IMAP.
- La experiencia con sockets TCP, fundamental para la implementación de los protocolos de aplicación.
- El uso del Logger personalizado, útil para registrar y depurar la comunicación entre cliente y servidor, así como la interacción entre dominios mediante UDP.

No obstante, el uso de elementos del laboratorio 1 y 2 no es obligatorio, tienen la libertad de implementar el sistema completamente desde cero, siempre y cuando se respeten los protocolos y funcionalidades definidas en el proyecto.

El sistema completo será capaz de:

- Recibir y Enviar emails utilizando el SMTP por TCP y UDP
- Consultar correos a través del protocolo IMAP por TCP
- Definir múltiples dominios de correo por ejemplo, universidad.com, Alumno01.com y que intercambian mensajes entre otros mediante UDP utilizando cliente/servidor SMTP y almacenarlos en SQLite.
- Interactuar con cliente real de Thunderbird, permitiendo probar y visualizar en tiempo real el envío y lectura de correos desde una interfaz gráfica.
- Entregar una estructura de LOGS utilizando Logger



Guía de Implementación:

- ➔ Como recomendación separé la lógica de cada protocolo en diferentes clases/archivos y así separar los protocolos de cada capa y reutilizar su función.

- ➔ Se recomienda iniciar por el SMTP/TCP Server junto a la estructura de Base de Datos de SQLite, la cual se conectará a Cliente de Thunderbird configurando el servidor SMTP al que se conectará el programa.

- ➔ Recuerde que la implementación de SQLite debe ser la misma clase para todos para evitar problemas como Race Condition, File Locking o Deadlock, al momento de leer o escribir a la base de datos.

- ➔ Al tener el SMTP/TCP con el SQLite almacenando los correos, puede iniciar en paralelo la implementación del SMTP/UDP Cliente/Servidor y el IMAP Server.

- ➔ Si no cree posible el anterior punto para abordar el problema, se recomienda seguir con el IMAP Server para mostrar los correos en el Buzón de Thunderbird y luego realizar el SMTP/UDP Cliente/Servidor para la Comunicaciones Distribuidas con UDP.

- ➔ Para la Distribución por UDP se realizará por medio de Broadcast, por lo que no se distribuirá comando a comando se debe transmitir toda la secuencia del SMTP Client en un solo mensaje sin las respuestas del SMTP Server y el Servidor UDP debe de discriminar los mensajes que no sean de su dominio o no exista el destinatario en sus buzones.

  NOTA: El tamaño máximo teórico del Payload de UDP son 65,507 bytes, el tamaño máximo sin fragmentar es aproximadamente de 1,472 bytes por lo que el tamaño recomendado es entre 512 – 1,200 bytes, esto para asegurar que solo está enviando un paquete y no tenga que lidiar con fragmentación.

  Recuerde que su mejor aliado es el Logger para poder descubrir qué es lo que está pidiendo Thunderbird, leer los RFCs.

SMTP (RFC 5321) (Simple Mail Transfer Protocol)

Es un protocolo de red utilizado para el envío y retransmisión de correos electrónicos a través de redes IP. SMTP especifica cómo los servidores de correo se comunican entre sí para transferir mensajes desde el cliente emisor hasta el servidor receptor. Su función principal es enviar correos salientes, no recibirlos, y normalmente opera sobre el puerto 25 o 587 para envío autenticado.

En su versión más simplificada que abordaremos en el proyecto utilizaremos al menos los siguientes comandos del Cliente al Servidor.

Comando	**Descripción**
HELO domain.com	Inicia la comunicación con el servidor
MAIL FROM:<user@domain.com>	Define el remitente del mensaje.
RCPT TO:<user@otherdomain.com>	Define el destinatario
DATA	Comienza el cuerpo del mensaje
. (punto)	Finaliza el cuerpo del mensaje
QUIT	Termina la Sesión

Ejemplo Teórico, en negrita los mensajes del servidor:

S: 220 mail.servidor.com SMTP ready
C: HELO universidad.com
S: 250 Hello universidad.com
C: MAIL FROM:<profesor@universidad.com>
S: 250 OK
C: RCPT TO:<alumno@maestros.com>
S: 250 OK
C: DATA
S: 354 End data with <CR><LF>.<CR><LF>
C: Subject: Tarea
C: Hola, esta es tu tarea.
C: .
S: 250 Message accepted
C: QUIT

IMAP (RFC 3501) (Internet Message Access Protocol)

Es un protocolo que permite acceder y gestionar correos electrónicos directamente desde el servidor, sin descargarlos. Así puedes ver tus correos desde varios dispositivos y siempre estarán sincronizados, normalmente opera sobre el puerto 143 o 993 para envío autenticado.

En su versión más simplificada que abordaremos en el proyecto utilizaremos al menos los siguientes comandos, al igual que SMTP estos comandos son los de Cliente al Servidor.

Comando	**Descripción**
LOGIN <usuario> <clave>	Autentica al usuario con su nombre y contraseña.
LIST "" "*"	Lista todos los buzones disponibles (en este caso, solo responde con "INBOX").
SELECT INBOX	Selecciona el buzón INBOX para visualizar y manipular mensajes.
FETCH <n> BODY[]	Recupera el contenido completo del mensaje número n.
FETCH <n> BODY[HEADER]	Recupera solo los encabezados del mensaje número n.
FETCH <n> BODY[TEXT]	Recupera solo el cuerpo del mensaje número n, sin encabezados.
LOGOUT	Finaliza la sesión IMAP del cliente.

Recuerde que Thunderbird tiene la implementación del IMAP Client completo por lo que el programa le va a estar solicitando más comandos de los que están listados posiblemente, la tarea en esta parte es poder responder correctamente a Thunderbird sin que este cause algún error o quede satisfecho con su respuesta.

Ejemplo Teórico, en negrita los mensajes del servidor:

S: * OK Servidor IMAP listo
C: A001 LOGIN usuario@ejemplo.com clave123
S: A001 OK LOGIN exitoso

C: A002 LIST "" "*"
S: * LIST (\HasNoChildren) "/" "INBOX"
S: A002 OK LIST completado

C: A003 SELECT INBOX
S: * FLAGS (\Seen)
S: * 3 EXISTS
S: * 0 RECENT
S: A003 OK INBOX seleccionado

C: A004 FETCH 1 BODY[HEADER]
S: * 1 FETCH (BODY[HEADER] {29}
Subject: Hola

)
S: A004 OK FETCH completado

C: A005 FETCH 1 BODY[TEXT]
S: * 1 FETCH (BODY[TEXT] {35}
Este es el cuerpo del mensaje 1.
)
S: A005 OK FETCH completado

C: A006 FETCH 2 BODY[HEADER]
S: * 2 FETCH (BODY[HEADER] {30}
Subject: Aviso

)
S: A006 OK FETCH completado

C: A007 FETCH 2 BODY[TEXT]
S: * 2 FETCH (BODY[TEXT] {35}
Este es el cuerpo del mensaje 2.
)
S: A007 OK FETCH completado

C: A008 FETCH 3 BODY[HEADER]
S: * 3 FETCH (BODY[HEADER] {32}
Subject: Reunión

)
S: A008 OK FETCH completado

C: A009 FETCH 3 BODY[TEXT]
S: * 3 FETCH (BODY[TEXT] {36}
Este es el cuerpo del mensaje 3.
)
S: A009 OK FETCH completado

C: A010 LOGOUT
S: * BYE Sesión cerrada
S: A010 OK LOGOUT completado

Notas finales:
El Cliente/Servidor SMTP en UDP que van a realizar no es estándar pero el protocolo UDP en modo Broadcast nos va a ayudar a transmitir el mensaje a todos los proyectos de la clase, este Servidor debe de poder soportar concurrencia, aunque este concepto se vió en el laboratorio 2 con el Threadpool con TCP ya que este está orientado a conexión, pero en UDP por el procesamiento del Paquete recibido también es necesario para procesar el mensaje y determinar si descartar el mensaje o guardarlo en el buzón de correo que se recibió.

Vuelvo a mencionar que el SQLite lo va a utilizar 4 procesos que son el SMTP/TCP Server, IMAP/TCP Server, SMTP/UDP Client y SMTP/UDP Server todos van a tener acceso a esa base de datos por lo que sincronizar correctamente es fundamental para evitar problemas.

Entrega

- Suba un archivo ZIP al GES con todos los archivos fuente.
- **La calificación se realizará de manera presencial**, en dado caso que no pueda presentarse enviar un video Explicando su Código, mostrando la compilación y ejecución, hacer varios ejemplos de su Funcionalidad leyendo los LOGs. Todo el grupo debe de explicar.
- Use Java 21 o 22 para su implementación.
- El Proyecto puede tener una calificación de cero si:
    - Si no compila con el Makefile (en java 21 o 22)
        - Servicio SMTP Relay (Server/Client).
        - Servicio IMAP Server.
    - Si no siguió las instrucciones del funcionamiento.
    - Si no cumple con los comandos IMAP/SMTP para la conexión con Thunderbird.