INSTITUTO TECNOLÓGICO DE BUENOS AIRES

72.07 - Protocolos de Comunicación

Trabajo Práctico Especial

Estudiantes

61105 - Causse, Juan Ignacio
61590 - De Caro, Guido
62774 - Mindlin, Felipe
62351 - Sendot, Francisco
61374 - Garcia Lauberer, Federico Inti

Docentes

GARBEROGLIO, Marcelo (Teórica)
CODAGNONE, Juan F. (Laboratorio)
KULESZ, Sebastián (Laboratorio)

Junio 2024

Índice

1. Introducción

El trabajo práctico especial de la materia Protocolos de Comunicación consistió en la construcción de un servidor SMTP respetando los RFC 5321 [1] y RFC 4954 [2], y el desarrollo y la puesta en funcionamiento de un cliente que utilice este protocolo. En el siguiente informe se detalla el proceso de creación del servidor y el protocolo.

En la sección 2 se detalla la descripción del protocolo.

En la sección 3 se explicarán los problemas surgidos durante la construcción del servidor e implementación del protocolo, y en los casos que aplique, la solución encontrada para ellos.

En la sección 4 se verán algunas limitaciones del proyecto.

En la sección 5 se verán posibles extensiones del servidor que van más allá del alcance de este proyecto.

En la sección 6 se dará la conclusión del informe.

En la sección 7 se mostrarán algunos casos de prueba del servidor, basadas en las clases prácticas.

En la sección 8 se dará una guía de instalación detallada, seguido de instrucciones para su configuración en la sección 9 y ejemplos de configuración y monitoreo en la sección 10.

Finalmente, en la sección 11 se mostrará un documento de diseño del proyecto, mostrando su arquitectura general.

2. Descripción detallada de los protocolos y aplicaciones desarrolladas:

2.1. SMTP

SMTP es un protocolo de transporte confiable y orientado a texto y a conexión, esencial en la infraestructura de correo electrónico. En el proyecto, se siguieron rigurosamente las especificaciones de los RFC 5321 [1] y RFC 4954 [2] que definen cómo deben operar los servidores SMTP. El RFC 4954 [2] detalla mecanismos de autenticación para validar clientes antes de aceptar correos. El RFC 5321 [1] detalla los estados de la sesión SMTP, y los comandos necesarios para iniciar, mantener y finalizar una conexión SMTP. A lo largo de la sesión se transiciona entre estos estados.

Al iniciar la sesión, se entra en el estado WELCOME. En este estado se dispone de los comandos *HELO*, *EHLO*, *QUIT* y *NOOP*.
A través del comando *HELO* se transiciona al estado *GREETING*. En este estado se debe recibir MAIL_FROM para transicionar al estado *MAIL_FROM_INPUT*, en el cual se escribe el correo del remitente. Una vez validado correctamente el correo, se transiciona al estado *MAIL_FROM_OK*.
Aquí se debe transiciona al estado *RCPT_TO_INPUT* y *RCPT_TO_OK* de manera análoga a los estados de *MAIL_FROM_INPUT* y MAIL_FROM.
Una vez en RCPT_TO_OK, se utiliza el comando DATA para entrar en el estado *DATA_INPUT*, el cual recibe los datos del correo hasta encontrar un '.<*CRLF*>', a través del cual se transiciona de vuelta al estado *GREETING*.
Y por último, a través del comando QUIT se accede al estado QUIT_ST y se finaliza la conexión.

El servidor SMTP en este proyecto está basado en una arquitectura no bloqueante implementada a través de la *API selector* y la función *select*.

2.2. MMP

Además de la implementación del SMTP, se implementó un servicio de monitoreo y configuración, que permite averiguar estadísticas y cambiar la configuración del servidor SMTP. Este servicio utiliza el protocolo MMP. Este protocolo está basado sobre UDP – no confiable y no orientado a conexión.

En esta versión, el servidor responde a las siguiente peticiones:

1. Recuento de conexiones totales.
2. Recuento de conexiones concurrentes.
3. Ver bytes transferidos.
4. Ver el estado de las transformaciones.
5. Habilitar transformaciones.
6. Deshabilitar transformaciones.

El protocolo se estructura en dos etapas principales:

1. **Solicitud Cliente→ Servidor:**
   El cliente inicia la comunicación enviando un paquete de solicitud estructurado.
   El paquete incluye la firma del protocolo, la versión, un identificador, datos de autenticación y un comando.
2. **Respuesta Servidor →Cliente:**
   Al recibir una solicitud válida, el servidor responde con un paquete de respuesta estructurado.
   La respuesta incluye la firma del protocolo, la versión, un identificador, el estado de la respuesta, la cantidad de datos (si corresponde) y una bandera booleana que indica un estado adicional.

3. Problemas encontrados durante el diseño y la implementación.

Para implementar el selector, se tuvo que recurrir a múltiples estructuras de datos que necesitamos implementar desde cero.

Una dificultad recurrente durante el desarrollo del proyecto fue la complejidad del mismo y la necesidad de hacer trabajar en conjunto todas las diferentes partes.
A la vez, esto hacía más difícil el encontrar errores.

La comunicación entre sockets también demostró una complejidad que requirió de trabajo minucioso y peer programming para solucionar los problemas que se presentaron.

Y por último el manejo de strings y caracteres a bajo nivel de C también causó problemas debido al nivel de detallismo requerido.

4. Limitaciones de la aplicación.

Para que el selector funcione correctamente, la data guardada debe ser siempre del mismo tipo. Si no es así, puede fallar al liberarlo.

5. Posibles extensiones de la aplicación.

En futuras versiones, se podría implementar un cierre de sesión automático por inactividad o timeout, de acorde a las especificaciones impuestas por *RFC 5321* [1] .

6. Conclusiones.

En este trabajo práctico se utilizaron los conocimientos adquiridos a lo largo de la cursada, permitiendo ponerlos en práctica en un proyecto de mayor escala, siguiendo los estándares establecidos para asegurar retrocompatibilidad y escalabilidad.

Se aprendió a manejar la comunicación entre clientes y servidores, se profundizó sobre nuestro entendimiento del protocolo SMTP durante su implementación, sobre sockets pasivos y activos, procesos bloqueantes, no bloqueantes y el manejo eficiente de lectura y escritura a través de selectores

7. Ejemplos de prueba.

8. Guía de instalación detallada y precisa e instrucciones de configuración.

Para compilar el servidor se debe ejecutar el script **build_server.sh**.
En 'main.c', cambiar la dirección default del smtpd log file definida en la constante CONFIG LOG_FILE.
Luego, se debe ejecutar el archivo generado **smtpd.bin**.

Los flags de **smtpd** son:

-d <domain name>: Nombre del dominio para el servidor
-s <SMTP port>: Puerto para el servidor SMTP
-p <management port>: Puerto para el servidor manager.
‘l <log file path>: Dirección deseada para el output de logging del servidor (*smtpd.lo*g)
Opcionales:

-L <log level>: Nivel de log mínimo.
-t <command path>: El comando de transformación a utilizar
-f <vrfy dir>: El directorio donde las direcciones de correo ya verificadas son guardadas y donde se guardarán las nuevas.
-v: Imprime la información de versión y corta la ejecución.
-h: Imprime los flags disponibles con su información pertinente.

Para compilar el manager se debe ejecutar el script **build_manager.sh**.
Luego, se debe ejecutar el archivo generado **manager.bin.**

Los flags de **manager** son:

-d <domain name>: Dominio del servidor SMTP
-p <Manager port in SMTP server>: Puerto del manager en el servidor SMTP.

El archivo **build.sh** compila ambos archivos en simultáneo, por lo que se generarán tanto **smtpd.bin** como **manager.bin**.

10. Ejemplos de configuración y monitoreo.

La configuración del lado de flags ya fue explicada en la anterior sección. En la sección actual veremos instrucciones para la configuración en tiempo de ejecución desde el **manager.**

```
federico@federico-HP-Laptop-15-bs0xx:~/Desktop/Protos/protos/src/manager
$ ./manager.bin -i 127.0.0.1 -p 6060

Command Menu:
0. Number of historical connections
1. Number of concurrent connections
2. Number of bytes transferred
3. Check transformation status
4. Transformations ON
5. Transformations OFF
Select a command (0-5):
```

Luego, en este menú elegimos la opción de configuración o de estadísticas que queremos obtener. Por ejemplo utilicemos **4** ***y*** **5.**

```
Command Menu:
0. Number of historical connections
1. Number of concurrent connections
2. Number of bytes transferred
3. Check transformation status
4. Transformations ON
5. Transformations OFF
Select a command (0-5): 4
 * Received 15 bytes

Received response:
Status: 0
Amount: 0
Boolean: 1

Transformation status = ON
```

```
Command Menu:
0. Number of historical connections
1. Number of concurrent connections
2. Number of bytes transferred
3. Check transformation status
4. Transformations ON
5. Transformations OFF
Select a command (0-5): 5
 * Received 15 bytes

Received response:
Status: 0
Amount: 0
Boolean: 0

Transformation status = OFF
```

Como se ve, se activó y desactivó la transformación en tiempo real.
Con la opción **3** se puede ver el estado actual del booleano de la transformación.

```
Command Menu:
0. Number of historical connections
1. Number of concurrent connections
2. Number of bytes transferred
3. Check transformation status
4. Transformations ON
5. Transformations OFF
Select a command (0-5): 3
 * Received 15 bytes

Received response:
Status: 0
Amount: 0
Boolean: 0
```

Y por último, los comandos **0** a **2** muestran las estadísticas del servidor.

11. Documento de diseño del proyecto.

La gestión de argumentos se realiza mediante el archivo *'args.c'*. Este se encarga de analizar y validar los argumentos recibidos por línea de comando en el programa principal. Recibe la dirección IP del servidor y el número de puerto y valida su sintaxis antes de pasar esos valores a '*main.c'*.

La gestión, lectura y escritura de sockets se realiza mediante el archivo '*sockets.c'*, el cual es responsable de la creación y configuración de los sockets utilizados en la aplicación.

Para una gestión no bloqueante y eficiente de los sockets se utiliza la API selector en el archivo *'selector.c'*, la cual despierta los file descriptors asociados a medida que estén listos para realizar sus operaciones de lectura / escritura.

Trabajando con el selector para realizar las acciones correspondientes a las operaciones de red, el archivo '*sock_types_handler.c'* define controladores para los distintos tipos de sockets, IPv4, IPv6, del lado servidor SMTP y del manager.

El archivo '*parser.c'* es el encargado del flujo principal del protocolo. Analiza los mensajes recibidos y actualiza la máquina de estados en base a estos, asegurándose que la comunicación siga el formato especificado – RFC 5321[1], 4954 [2].
Como fue mencionado en la descripción del protocolo, los estados son *WELCOME,VRFY_PARAM, HELO_DOMAIN, EHLO _DOMAIN, GREETING, MAIL_FROM_INPUT, MAIL_FROM_OK, RCPT_TO_INPUT, RCPT_TO_OK, DATA_INPUT, QUIT_ST.*
En base al estado actual del parser, la función *parseCmd()* es la encargada de decidir cómo procesar el texto de entrada y actualizar el estado del parser, ejecutando las funciones de transición pertinentes. Luego, retorna *ERR* si hubo un error procesando un comando, *SUCCESS* si se procesó con éxito, y *TERMINAL* si se debe finalizar la conexión.

En el archivo '*main.c*' se encuentra el punto de entrada principal de la aplicación. Inicializa los componentes principales de la misma – *args.c, sockets.c selector.c, central.c.*
Es el encargado de coordinar la interacción entre estos, gestionando la ejecución de la aplicación y la interacción con el usuario a través del cliente interactivo.

Para interactuar con el sistema, se utiliza el cliente interactivo ubicado en '*manager.c'*. Utiliza funciones de sockets y del parser para enviar solicitudes al servidor y procesar las respuestas recibidas, desplegando un menú para que el usuario decida qué comando correr. Es capaz de ver métricas y cambiar la configuración del servidor SMTP en tiempo de ejecución.

El archivo *'stats.c'* proporciona la funcionalidad de poder obtener y medir estadísticas del protocolo.

Bibliografía.

[1] RFC 5321: Simple Mail Transfer Protocol. The Internet Engineering Task Force. October 2008.

[2] RFC 4954: SMTP Service Extension for Authentication. The Internet Engineering Task Force. July 2007.

[3] RFC 3778: Taft, E., Pravetz, J., Zilles, S., and L. Masinter, "Theapplication/pdf Media Type", RFC 3778, DOI 10.17487/RFC3778, May 2004