Imagen que contiene Icono

El contenido generado por IA puede ser incorrecto.Dibujo en blanco y negro

El contenido generado por IA puede ser incorrecto.

Instituto Politécnico Nacional

Escuela Superior de Cómputo

Ingeniería en Sistemas Computacionales

Aplicaciones para comunicaciones en red

Academia “Sistemas Distribuidos”

Plan 2020

Práctica 21:

“**Protocolo SMTP**”

2015090269

González González Armando Omar

Profesor: Ojeda Santillan Rodrigo

Contenido

[Objetivo 3](#_Toc201738905)

[Introducción 3](#_Toc201738906)

[Desarrollo 4](#_Toc201738907)

[Conclusión 8](#_Toc201738908)

[Pregunta 8](#_Toc201738909)

[¿Cómo podrían generar un negocio a través de lo visto en la práctica? 8](#_Toc201738910)

[Bibliografía 8](#_Toc201738911)

## Objetivo

Desarrollar un cliente en lenguaje C capaz de comunicarse con un servidor SMTP para enviar un correo electrónico. El objetivo es comprender el funcionamiento y la secuencia de comandos del Protocolo de Transferencia de Correo Simple (SMTP), estableciendo una conexión de red, intercambiando los comandos necesarios y transmitiendo el contenido de un mensaje de prueba.

## Introducción

El Protocolo de Transferencia de Correo Simple, o SMTP (Simple Mail Transfer Protocol), es el protocolo de comunicación estándar utilizado para enviar correos electrónicos a través de una red. Su función principal es gestionar la transferencia de mensajes desde un cliente de correo hacia un servidor, y entre diferentes servidores de correo, hasta alcanzar el destino final. SMTP es un protocolo basado en texto que opera en la capa de aplicación del modelo OSI, utilizando típicamente el puerto 25 para conexiones estándar o el puerto 587 para conexiones seguras.

El proceso de envío de un correo mediante SMTP sigue un flujo de comandos bien definido:

* **Conexión:** El cliente establece una conexión TCP con el servidor SMTP.
* **Saludo (HELO/EHLO):** El cliente se presenta al servidor enviando un comando de saludo. El servidor responde con un código de estado "250 OK" si todo es correcto.
* **MAIL FROM:** El cliente indica la dirección del remitente del correo.
* **RCPT TO:** El cliente especifica la dirección del destinatario.
* **DATA:** El cliente envía este comando para señalar que a continuación transmitirá el contenido del mensaje. El servidor responde indicando que está listo para recibirlo.
* **Envío del Mensaje:** Se envían las cabeceras (Asunto, De, Para) y el cuerpo del mensaje, finalizando la transmisión con una línea que contiene únicamente un punto (.).
* **QUIT:** El cliente envía este comando para terminar la sesión y cerrar la conexión.

Es importante destacar que SMTP se especializa únicamente en el envío de correos; para la recepción y lectura de correos por parte del usuario final, se utilizan otros protocolos como POP3 o IMAP.

## Desarrollo

Para la realización de esta práctica, se utilizó un servidor SMTP local para pruebas llamado MailHog y se desarrolló un cliente en lenguaje C para interactuar con él.

### **3.1 Configuración del Servidor de Pruebas (MailHog)**

Para simular un entorno real sin necesidad de configurar un servidor de correo completo, se utilizó MailHog. MailHog es una herramienta que actúa como un servidor SMTP falso, capturando todos los correos enviados a través de él y mostrándolos en una interfaz web, lo cual es ideal para desarrollo y pruebas. Por defecto, MailHog escucha las conexiones SMTP en el puerto 1025.

### **3.2 Implementación del Cliente SMTP (**client.c**)**

Se desarrolló un programa en C que implementa la lógica de un cliente SMTP. El programa establece una conexión con el servidor MailHog y sigue la secuencia de comandos del protocolo para enviar un correo electrónico de prueba.

El código se estructura de la siguiente manera:

1. **Definiciones y Cabeceras:** Se incluyen las bibliotecas necesarias para la programación de sockets (arpa/inet.h, unistd.h) y para la manipulación de cadenas (string.h, stdio.h). Se definen constantes para la dirección IP del servidor, el puerto y el tamaño del buffer.
2. **Función send\_smtp\_command:** Esta función auxiliar se encarga de enviar un comando SMTP al servidor, recibir la respuesta, imprimir ambas en la consola para depuración y verificar que la respuesta del servidor contenga el código de estado esperado (ej. "250").
3. **Función main:** Es el punto de entrada del programa.
   * Crea un socket de tipo AF\_INET (IPv4) y SOCK\_STREAM (TCP).
   * Configura la dirección del servidor y el puerto para la conexión.
   * Establece la conexión con el servidor usando connect().
   * Una vez conectado, sigue el flujo de SMTP enviando secuencialmente los comandos: HELO, MAIL FROM, RCPT TO y DATA.
   * Después del comando DATA, envía las cabeceras del correo (Asunto, De, Para) y el cuerpo del mensaje, terminando con la secuencia \r\n.\r\n.

A continuación se muestra el código completo del cliente:

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <unistd.h>

#include <arpa/inet.h>

#define SERVER "127.0.0.1" // Dirección IP del servidor SMTP (MailHog)

#define PORT 1025 // Puerto de MailHog (1025 por defecto)

#define BUFSIZE 1024

void send\_smtp\_command(int sockfd, const char \*cmd, const char \*expected\_response)

{

char buffer[BUFSIZE] = {0};

// Enviar el comando SMTP

send(sockfd, cmd, strlen(cmd), 0);

printf("Enviado: %s", cmd);

// Leer la respuesta del servidor

recv(sockfd, buffer, BUFSIZE, 0);

printf("Respuesta: %s", buffer);

// Verificar si la respuesta es la esperada

if (strstr(buffer, expected\_response) == NULL)

{

fprintf(stderr, "Error: no se recibió la respuesta esperada.\n");

exit(EXIT\_FAILURE);

}

}

int main() {

int sockfd;

struct sockaddr\_in server\_addr;

// Create socket

if ((sockfd = socket(AF\_INET, SOCK\_STREAM, 0)) < 0) {

perror("Socket creation failed");

exit(EXIT\_FAILURE);

}

// Configure server address

memset(&server\_addr, 0, sizeof(server\_addr));

server\_addr.sin\_family = AF\_INET;

server\_addr.sin\_port = htons(PORT);

// Convert IP address from text to binary form

if (inet\_pton(AF\_INET, SERVER, &server\_addr.sin\_addr) <= 0) {

perror("Invalid address/Address not supported");

exit(EXIT\_FAILURE);

}

// Connect to server

if (connect(sockfd, (struct sockaddr \*)&server\_addr, sizeof(server\_addr)) < 0) {

perror("Connection failed");

exit(EXIT\_FAILURE);

}

// printf("Connected to SMTP server at %s:%d\n", SERVER, PORT);

// Read initial server greeting

char buffer[BUFSIZE];

ssize\_t bytes\_received = recv(sockfd, buffer, BUFSIZE - 1, 0);

if (bytes\_received < 0) {

perror("Error reading initial greeting");

exit(EXIT\_FAILURE);

}

buffer[bytes\_received] = '\0';

printf("Server greeting: %s\n", buffer);

// Enviar los comandos SMTP

send\_smtp\_command(sockfd, "HELO localhost\r\n", "250");

send\_smtp\_command(sockfd, "MAIL FROM:<test@example.com>\r\n", "250");

send\_smtp\_command(sockfd, "RCPT TO:<recipient@example.com>\r\n", "250");

send\_smtp\_command(sockfd, "DATA\r\n", "354");

// Enviar el contenido del correo

send\_smtp\_command(sockfd, "Subject: Test Mail\r\n", "");

send\_smtp\_command(sockfd, "From: test@example.com\r\n", "");

send\_smtp\_command(sockfd, "To: recipient@example.com\r\n\r\n", "");

send\_smtp\_command(sockfd, "Este es un mensaje de prueba.\r\n.\r\n", "250");

return 0;

}

A screenshot of a computer

AI-generated content may be incorrect.  
Figura 1. Ejecución de MailHog.

A screenshot of a computer

AI-generated content may be incorrect.  
Figura 2. Ejecución del cliente y envío y confirmación de comandos del protocolo.

## Conclusión

La práctica se completó exitosamente, logrando enviar un correo electrónico desde un cliente desarrollado en C a un servidor SMTP de prueba. El desarrollo permitió comprender de manera práctica la naturaleza de SMTP como un protocolo de texto basado en una secuencia estricta de comandos y respuestas. La función send\_smtp\_command fue clave para modularizar la comunicación y verificar cada paso del proceso, asegurando que la sesión progresara correctamente.

Se concluye que SMTP, a pesar de su simplicidad, es un protocolo robusto y fundamental para la infraestructura de internet. La realización de esta práctica ha sido fundamental para visualizar y entender a bajo nivel cómo se lleva a cabo una de las tareas más comunes en la red: el envío de un correo electrónico.

## Pregunta

### ¿Cómo podrían generar un negocio a través de lo visto en la práctica?

Se podría crear un servicio de "SMTP Relay" o API de correo transaccional. Muchas empresas necesitan enviar correos automatizados (confirmaciones de compra, notificaciones, restablecimiento de contraseñas) de forma masiva y confiable. Un negocio podría ofrecer una plataforma SMTP robusta y de alta disponibilidad, garantizando que los correos de sus clientes no caigan en spam y se entreguen rápidamente, cobrando una tarifa por volumen de envío.

## Bibliografía

[1] J. Klensin, "RFC 5321: Simple Mail Transfer Protocol," Internet Engineering Task Force (IETF), Oct. 2008. [En línea]. Disponible: <https://www.rfc-editor.org/rfc/rfc5321>.

[2] J. F. Kurose and K. W. Ross, Computer Networking: A Top-Down Approach, 8th ed. Boston: Pearson, 2021.

[3] W. R. Stevens, B. Fenner, and A. M. Rudoff, UNIX Network Programming, Volume 1: The Sockets Networking API, 3rd ed. Boston: Addison-Wesley Professional, 2003.