PROGRAMACIÓN EN RED (SOCKETS)

(CC) Moreno, A. M. & Bravo, S. (Redes I, 2022)

Contenido

- Presentación
  - Objetivos y entorno de desarrollo
  - Servicio SMTP simplificado
    - Especificaciones del protocolo
      - Mensajes
      - Ejemplos de dialogo
    - Requisitos

Objetivos y entorno de desarrollo

- El objetivo de esta práctica es implementar un aplicación en red como
  - Usuario del nivel de transporte y
  - Según el modelo cliente-servidor
- Entorno de desarrollo
  - Estación de trabajo con S.O. Debian GNU/Linux 9 (*stretch*) (nogal.usal.es)
  - Sockets de Berkeley
  - Lenguaje de programación C

Especificaciones del protocolo

- El servicio que vamos a implementar se corresponde con el protocolo para transferencia simple de correo (SMTP *Simple Mail Transfer Protocol*) definido inicialmente en agosto de 1982 por el RFC 821 (para la transferencia) y el RFC 822 (para el mensaje)
- No obstante, nuestro servidor no implementará todo el protocolo, sino únicamente un subconjunto muy reducido de este
- Además, SMTP se proporciona sobre TCP, pero nosotros realizaremos también una versión para UDP

Mensajes SMTP (I)

- SMTP emplea dos tipos de mensajes:
  - Peticiones de los clientes a los servidores
  - Respuestas de los servidores a los clientes
  - Las líneas de los mensajes SMTP
    - Son siempre líneas de caracteres terminadas con los caracteres CR-LF (retorno de carro "\r" (ASCII 13 (0x0D)), - avance de línea "\n" (ASCII 10 (0x0A)))
    - Para facilitar la implementación en UDP supondremos que la longitud máxima de las líneas no deben exceder 516 bytes, contando todos los caracteres (incluido el CR-LF final)
- **Peticiones del cliente al servidor**
  - El dialogo con el servidor comienza con la orden HELO
  - A continuación se envía un correo con tres pasos:
    - Con la orden MAIL se proporciona la información del originador
    - Los receptores (uno o más) se especifican con la orden RCPT
    - El contenido del mensaje con la orden MAIL
    - Si no se desea enviar más correos se finaliza con la orden QUIT

Mensajes SMTP (II)

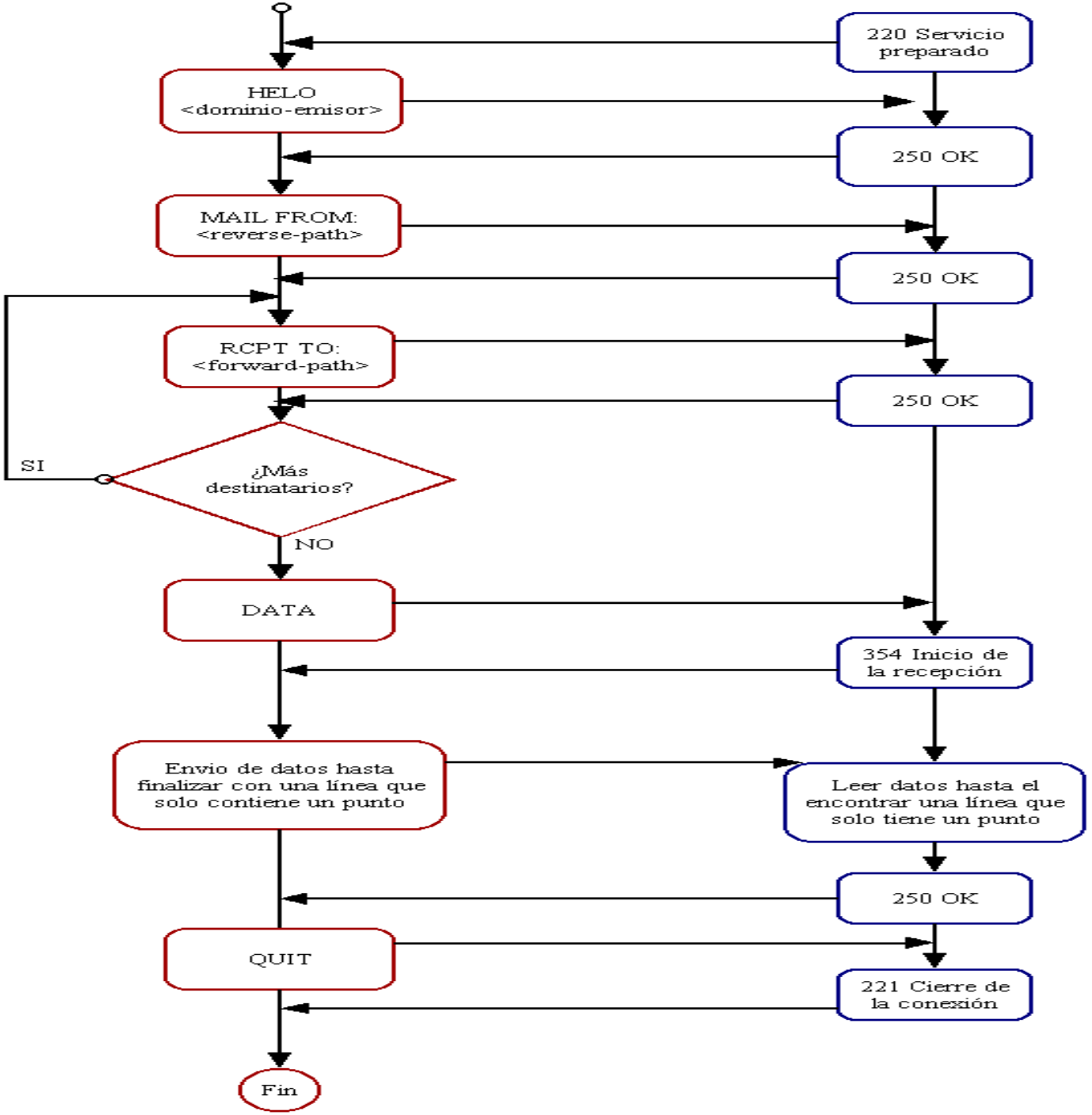
- **Peticiones del cliente al servidor**
  - La orden HELO se forma: HELO sender-domain[CR-LF]
    - Donde sender-domain es el dominio del originador y [CR-LF] los caracteres de retorno de carro y salto de línea. Por ejemplo: usal.es
  - La orden MAIL se forma: MAIL FROM: <reverse-path>[CR-LF]
    - Donde *reverse-path* es el correo electrónico del originador
  - La orden RCPT: RCPT TO: <forward-path>[CR-LF]
    - Donde *forward-path* es el correo electrónico del receptor
  - La orden DATA:
    - DATA[CR-LF]
    - ...
    - .[CR-LF]

    - Los puntos suspensivos serán el texto del mensaje que finaliza cuando se mande una línea que sólo contenga un punto
  - Se pueden enviar los correos que se desee hasta que se finaliza con la orden QUIT

Mensajes SMTP (III)

- **Respuestas del servidor**

Núm.	Cadena	Descripción
220	Servicio de transferencia simple de correo preparado	Respuesta cuando el cliente realiza la conexión
250	OK	Respuesta correcta a las ordenes MAIl, RCPT, DATA
354	Comenzando con el texto del correo, finalice con .	Respuesta al envío de la orden DATA
221	Cerrando el servicio	Respuesta a la orden QUIT
500	Error de sintaxis	Respuesta a errores de sintaxis en cualquier orden



NOTA: En la versión para UDP el cliente enviará un primer mensaje con una línea en blanco que sólo contenga los caracteres CR-LF puesto que en UDP no hay petición de conexión

Mensajes SMTP (IV)

Ejemplo de dialogo (S: Servidor, C: Cliente)

- S:220 Servicio de transferencia simple de correo preparado
- C: HOLA usal.es
- S: 500 Error de sintaxis
- C: HELO usal.es
- S: 250 OK
- C: MAIL FROM pepe
- S: 500 sintaxis
- C: MAIL FROM: <Smith@Alpha.ARPA>
- S: 250 OK
- C: RCPT TO: <Jones@Beta.ARPA>
- S: 250 OK
- C: RCPT TO: <Brown@Beta.ARPA>
- S: 250 OK
- C: DATOS
- S: 500 Error de sintaxis
- C: DATA
- S: 354 Comenzando con el texto del correo, finalice con .
- C: Bla bla bla...
- C: ...etc. etc. etc.
- C:.
- S: 250 OK
- C: QUIT
- S: 221 Cerrando el servicio

NOTA: Las líneas de los mensajes SMTP son siempre líneas de caracteres terminadas con los caracteres CR-LF (retorno de carro "\r" (ASCII 13 (0x0D)), - avance de línea "\n" (ASCII 10 (0x0A)))

Requisitos (I)

- **Programa Servidor**
  - Aceptará peticiones de sus clientes tanto en TCP como en UDP
  - Registrará todas las peticiones en un fichero de "log" llamado peticiones.log en el que anotará:
    - Fecha y hora del evento
    - Descripción del evento:
      - Comunicación realizada: nombre del host, dirección IP, protocolo de transporte, nº de puerto efímero del cliente
      - Una línea por cada mensaje recibido y enviado indicando nombre del host, dirección IP, protocolo de transporte, nº de puerto efímero del cliente y todos los datos del mensaje
      - Comunicación finalizada: nombre del host, dirección IP, protocolo de transporte, nº de puerto efímero del cliente
  - Se ejecutará como un “daemon”
- **Programa Cliente**
  - Se comunicará con el servidor bien con TCP o con UDP
  - Leerá por parámetros el nombre o IP del servidor y el protocolo de transporte TCP o UDP de la siguiente forma:
    - cliente nombre_o_IP_del_servidor TCP
  - Realizará peticiones al servidor como se ha indicado anteriormente
  - Realizará las acciones oportunas para su correcta finalización

Requisitos (II): pruebas

- Durante la fase de pruebas el cliente podrá ejecutarse como se muestra en el ejemplo de diálogo anterior, pero en la versión para entregar el cliente
  - Leerá de un fichero las órdenes que ha de ejecutar. El nombre del fichero lo recibirá como parámetro
  - Escribirá las respuestas obtenidas del servidor y los mensajes de error y/o depuración en un fichero con nombre el número de puerto efímero del cliente y extensión .txt

Requisitos (III): versión entregable

- Para verificar que esta práctica funciona correctamente y permite operar con varios clientes, se utilizará el *script* lanzaServidor.sh que ha de adjuntarse obligatoriamente en el fichero de entrega de esta práctica
- El contenido de lanzaServidor.sh es el siguiente:

```
# lanzaServidor.sh
# Lanza el servidor que es un daemon y varios clientes
# las ordenes están en un fichero que se pasa como tercer
  parámetro
./servidor
./cliente nogal TCP ordenes.txt &
./cliente nogal TCP ordenes1.txt &
./cliente nogal TCP ordenes2.txt &
./cliente nogal UDP ordenes.txt &
./cliente nogal UDP ordenes1.txt &
./cliente nogal UDP ordenes2.txt &
```

Requisitos (IV): documentación

- Entregar un informe en formato PDF que contenga:
  - Detalles relevantes del desarrollo de la práctica
  - Documentación de las pruebas de funcionamiento realizadas

PROGRAMACIÓN EN RED (SOCKETS)

(CC) Moreno, A. M. & Bravo, S. (Redes I, 2022)