

# Práctica 4:

## Correo electrónico

---

### 1 Trabajo previo:

Para el correcto desarrollo de la práctica es conveniente acudir al laboratorio habiendo estudiado los protocolos SMTP y POP3. De esa forma podrás responder correctamente las cuestiones que se plantean y comprender mejor las capturas de tráfico realizadas. Por ello, previamente debes:

- Estudiar el capítulo 2.4 del libro de Kurose.

Si deseas información de consulta opcional, puedes consultar también los siguientes recursos:

- SMTP:  
<http://www.rfc-editor.org/info/rfc5321>
- POP3:  
<http://www.rfc-editor.org/info/rfc1939>
- Guía usuario Wireshark:  
<http://www.wireshark.org/docs/>
- Java API Specification:  
<http://docs.oracle.com/javase/8/docs/api/index.html>

### 2 Objetivos de la práctica

En esta práctica vamos a revisar los protocolos SMTP y POP3 capturando mediante Wireshark sesiones de los mismos y programando en Java unos sencillos clientes con sockets TCP que se conecten a los servidores SMTP y POP3 que hay en ejecución en el servidor `serveis-rdc.redes.upv.es`.

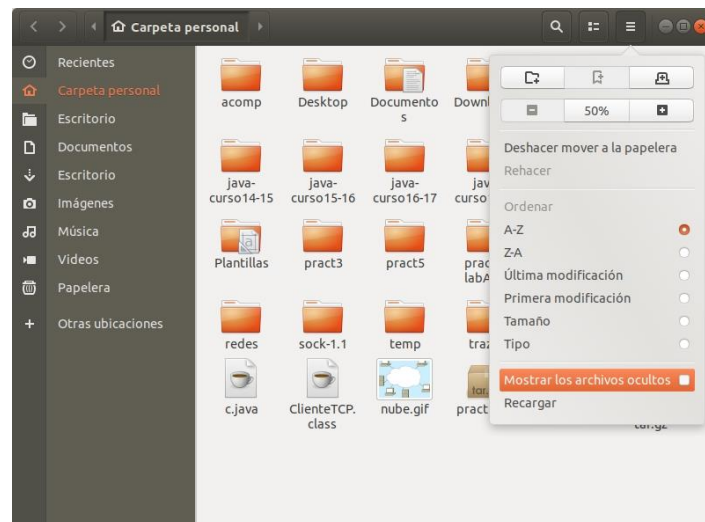
Al acabar la práctica deberías ser capaz de:

- Enumerar la secuencia de órdenes utilizadas por el cliente durante una sesión SMTP y una sesión POP3, para el envío (o recepción) de un mensaje de correo.
- Describir el significado de las órdenes de los protocolos SMTP y POP3.
- Explicar el formato de los mensajes de correo (incluida la separación entre cabeceras y cuerpo).
- Escribir un programa cliente básico en Java que sea capaz de realizar el diálogo con el servidor SMTP (o POP3) utilizando las órdenes necesarias (sin incluir el análisis de los posibles códigos de error del servidor).

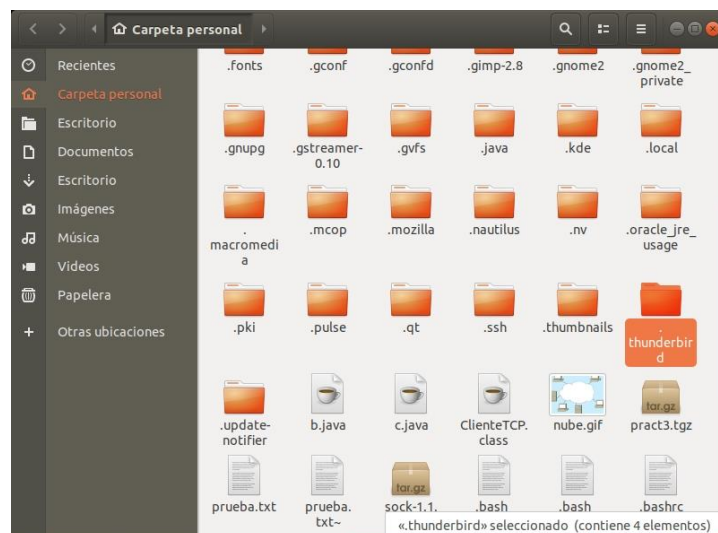
### 3 Primera parte. Captura de sesiones de los protocolos de correo.

La primera parte de la práctica consistirá en la captura y análisis de sesiones de los protocolos SMTP y POP3 mediante *Wireshark*. Para ello configuraremos el Agente de Usuario (UA) *Thunderbird* para que pueda acceder a las cuentas de correo electrónico definidas en el servidor *serveis-rdc.redes.upv.es*.

En primer lugar, antes de ejecutar *Thunderbird*, vamos a asegurarnos de que partimos de una estado limpio. Para ello borraremos los ficheros de configuración de *Thunderbird*, en caso de que los hubiera, abriendo el explorador de ficheros y seleccionando “Mostrar los archivos ocultos”, tal y como se muestra en la figura.



A continuación buscamos la carpeta “.thunderbird” y la movemos a la papelera. Es posible que dicha carpeta no exista si nunca hemos ejecutado *Thunderbird* antes.



## Correo Electrónico

Ahora ejecutaremos *Thunderbird*. Aparecerá la siguiente ventana, donde deberemos introducir los datos de la cuenta de correo.



The screenshot shows a window titled "Configurar una dirección de correo existente". It contains three input fields: "Su nombre:" with the value "Alumno de Redes", "Dirección de correo:" with the value "redes01@redes.upv.es", and "Contraseña:" with masked characters "\*\*\*\*\*". To the right of the password field is a checkbox labeled "Recordar contraseña" which is checked. Below the input fields are three buttons: "Obtener una nueva dirección de correo...", "Cancelar", and "Continuar".

En la nueva ventana introduciremos los datos del usuario con su dirección de correo, que será de la forma `redesXX@redes.upv.es` (sustituyendo XX por el número de tu puesto de trabajo), y la contraseña que te será suministrada por el profesor de prácticas.

Al darle a continuar aparece una ventana en la que deberemos configurar las opciones del protocolo de gestión del buzón de correo (entrante) y del protocolo de envío de correo (saliente), tal y como se muestra en la siguiente figura.



The screenshot shows a window titled "Configuración de cuenta de correo". It contains the same input fields as the previous window. Below these fields is a message: "Se ha encontrado la siguiente configuración sondeando el servidor suministrado". This is followed by a table of configuration options for incoming and outgoing mail. The table has five columns: "Entrante:", "Nombre del servidor", "Puerto", "SSL", and "Identificación". The "Entrante:" row shows "POP3", "serveis-rdc.redes.upv.es", "110", "Ninguno", and "Contraseña normal". The "Saliente:" row shows "SMTP", "serveis-rdc.redes.upv.es", "25", "Ninguno", and "Contraseña normal". Below the table are two input fields: "Nombre de usuario: Entrante:" with the value "redes01" and "Saliente:" with the value "redes01". At the bottom are four buttons: "Obtener una nueva cuenta", "Config. avanzada", "Cancelar", and "Volver a probar" (which is highlighted with a red box). A "Hecho" button is also present.

Mediante el botón *Volver a probar* podemos verificar que la configuración de los puertos es correcta.

Finalmente, pulsar el botón *Hecho*. Una ventana nos advertirá del problema de utilizar contraseñas no encriptadas en protocolos no seguros. Pinchamos en “Entiendo los riesgos” y después en el botón “Hecho”.



Para verificar que la configuración es la correcta vamos a capturar una sesión SMTP mediante *Wireshark*.

### Ejercicio 1: Captura SMTP

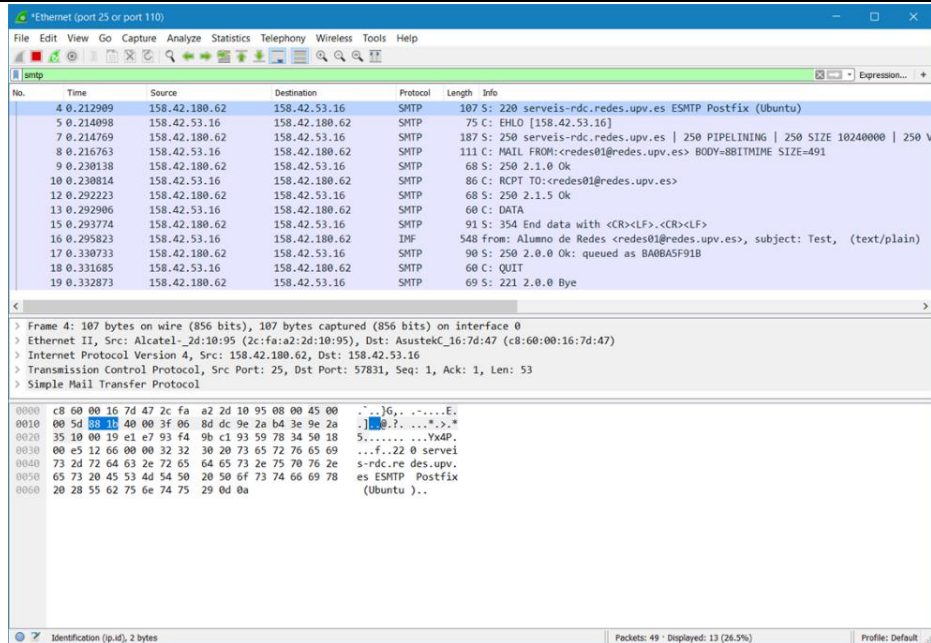
Captura una sesión SMTP. Deberás lanzar el programa *Wireshark* y ponerlo a capturar filtrando solamente el tráfico SMTP (puerto 25).

A continuación, en el *Thunderbird* compones un mensaje destinado a `redesXX@redes.upv.es` (sustituyendo XX por el número de tu puesto de trabajo) y lo envías.

También puedes filtrar la información relativa al protocolo SMTP mediante un filtro de visualización una vez realizada la captura.

Deberías obtener algo similar a lo siguiente:

## Correo Electrónico



Selecciona el primer mensaje SMTP y aplícale el análisis “*Follow TCP Stream*” y a continuación guarda el diálogo en un fichero de texto para su posterior análisis. La siguiente imagen muestra ese diálogo.



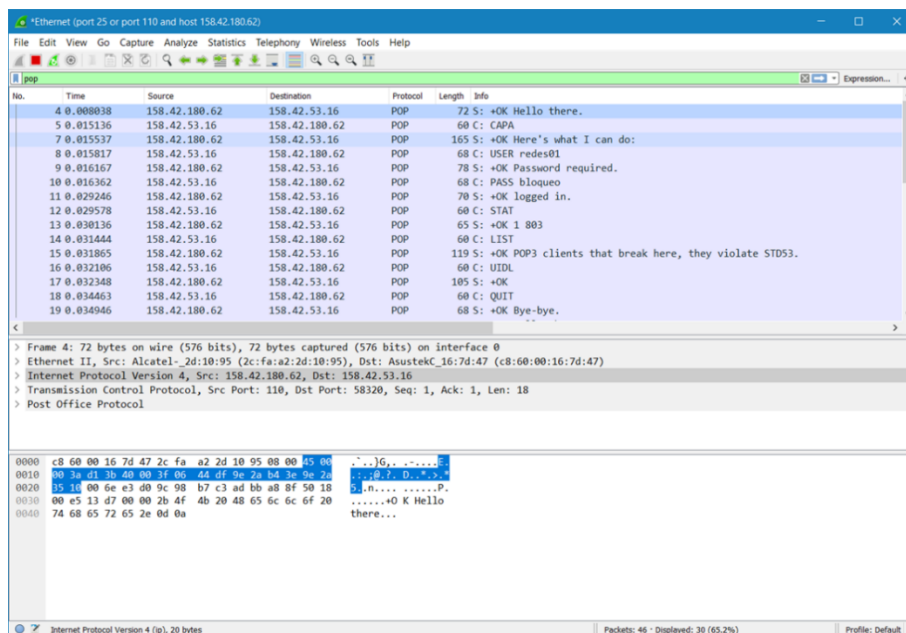
El diálogo que has guardado te ayudará a implementar un cliente SMTP más adelante en esta misma práctica.

## Ejercicio 2: Captura POP3

Captura una sesión POP3. Deberás lanzar el *Wireshark* y ponerlo a capturar filtrando el tráfico POP3 (puerto 110). Posteriormente, en el *Thunderbird*, compones un mensaje destinado a `redesXX@redes.upv.es` (sustituyendo XX por el número de tu puesto de trabajo) y lo envías (hasta aquí el tráfico generado es SMTP). A continuación, pulsas el botón “*Recibir mensajes*”.

Posteriormente filtras en *Wireshark* la información relativa al protocolo POP3 mediante el filtro de visualización “*pop*”.

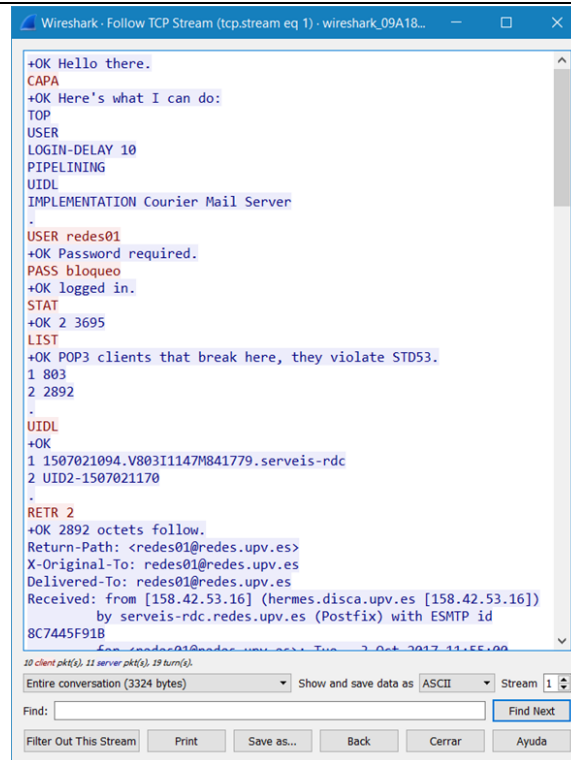
Deberías obtener algo similar a lo siguiente:



Al igual que en el ejercicio anterior, selecciona el primer mensaje POP3 y aplícale el análisis “*Follow TCP Stream*”.

A continuación, guarda el diálogo en un fichero de texto para su posterior análisis.

## Correo Electrónico



```
+OK Hello there.
CAPA
+OK Here's what I can do:
TOP
USER
LOGIN-DELAY 10
PIPELINING
UIDL
IMPLEMENTATION Courier Mail Server
.
USER redes01
+OK Password required.
PASS bloqueo
+OK logged in.
STAT
+OK 2 3695
LIST
+OK POP3 clients that break here, they violate STD53.
1 803
2 2892
.
UIDL
+OK
1 1507021094.V803I1147M841779.serveis-rcd
2 UID2-1507021170
.
RETR 2
+OK 2892 octets follow.
Return-Path: <redes01@redes.upv.es>
X-Original-To: redes01@redes.upv.es
Delivered-To: redes01@redes.upv.es
Received: from [158.42.53.16] (hermes.disca.upv.es [158.42.53.16])
by serveis-rcd.redes.upv.es (Postfix) with ESMTP id
8C7445F91B
from=redes01@redes.upv.es; Tue, 2 Oct 2017 11:55:00
-0400
```

El diálogo que has guardado te ayudará a implementar un cliente POP3 más adelante en esta misma práctica.

## 4 Segunda parte. Programación de un Cliente SMTP

Vamos a comenzar creando un sencillo programa en Java que se conecte al servidor SMTP que hay en ejecución en *serveis-rcd.redes.upv.es* de forma que envíe un correo electrónico a la cuenta *redesXX@redes.upv.es* (“XX” depende del puesto de trabajo del laboratorio donde realices la práctica). Con el fin de simplificar la programación del cliente realizado, las direcciones de correo electrónico del emisor y del receptor del mensaje serán fijas. Lo mismo ocurrirá con el mensaje enviado. En una versión más elaborada del programa podrían introducirse por teclado.

Para facilitar la labor, se ha preparado una primera versión incompleta del cliente SMTP, que se puede descargar del directorio de prácticas en *poliformaT*. El fichero a descargar se llama “ClienteSMTP\_incompleto.java”. **Ten en cuenta que el nombre del fichero no coincide con el nombre de la clase que define**, por lo que si utilizas el compilador **javac** deberás modificar uno de los dos nombres para que el compilador no de un error.



### Ejercicio 3: Cliente SMTP.

Completa el código del cliente SMTP proporcionado para que envíe un correo electrónico a la dirección `redesXX@redes.upv.es`. Como dirección del emisor puedes utilizar una de tus direcciones de correo.

El correo electrónico debe contener, al menos, las cabeceras “*From:*”, “*To:*” y “*Subject:*”. Además, en el cuerpo del mensaje, **debe aparecer alguna línea que empiece por el carácter de puntuación punto (.)** y que no sea la última del mensaje. Comprueba el correcto funcionamiento del programa ejecutándolo en tu puesto de trabajo. Prueba a modificar el programa y ejecutarlo varias veces con diferentes asuntos en la cabecera “*Subject:*”. También puedes enviarte un correo a tu dirección principal y comprobar que lo recibes. Comprueba la correcta recepción de los mensajes de correo con *Thunderbird*.

**NOTA:** Si utilizas como referencia la captura del ejercicio 1, ten en cuenta que el cliente *Thunderbird* utiliza la orden EHLO, que pertenece al protocolo ESMT (*Extended SMTP*). **En tu implementación utiliza la orden HELO.** De esta forma tu cliente utilizará el protocolo SMTP y el servidor te devolverá su respuesta en una única línea, en lugar de la respuesta de varias líneas obtenida en la captura.

## 5 Tercera parte. Programación de un Cliente POP3

A continuación, vamos a completar el código de un cliente POP3 para que se conecte al servidor de correo y pueda realizar diversas operaciones con el buzón.

Se ha preparado una primera versión incompleta del cliente POP3, que se puede descargar del directorio de prácticas en *poliformaT*. El fichero a descargar se llama “*ClientePOP3\_incompleto.java*”. **Ten en cuenta que el nombre del fichero no coincide con el nombre de la clase que define**, por lo que si utilizas el compilador **javac** deberás modificar uno de los dos nombres para que el compilador no de un error.

### Ejercicio 4: Cliente POP3

Completa el código del cliente POP3 proporcionado para que lea y borre los mensajes almacenados en el buzón de correo asociado a la dirección `redesXX@redes.upv.es` que se encuentra en `serveis-rdc.redes.upv.es`. Comprueba el correcto funcionamiento del programa ejecutándolo en tu puesto de trabajo. Prueba a leer y borrar los diversos mensajes que has enviado en el ejercicio anterior.



**Cuestiones:**

Cuestión 4.1. ¿Qué ha ocurrido con la línea del mensaje que comenzaba por un punto?

Cuestión 4.2. ¿Qué crees que hubiese ocurrido si hubieses enviado entre tus datos una línea que contuviese un único punto?

Como habrás observado en la línea que comenzaba con un punto, el punto inicial ha desaparecido. Esto se debe a que para permitir que el usuario pueda enviar cualquier tipo de texto, incluido una línea que solo contenga un punto, los clientes SMTP cuando encuentran una línea que comienza por un punto, lo duplican, es decir añaden un punto extra (pero nuestro cliente no ha tenido en cuenta esta condición del estándar). Por su parte, cuando el servidor encuentra una línea que empieza por un punto lo elimina. Si la línea incluye más caracteres además de los de fin de línea, la trata como una línea normal. Por el contrario, si la línea sólo contenía el punto y los caracteres <CRLF>, detecta el fin de mensaje.