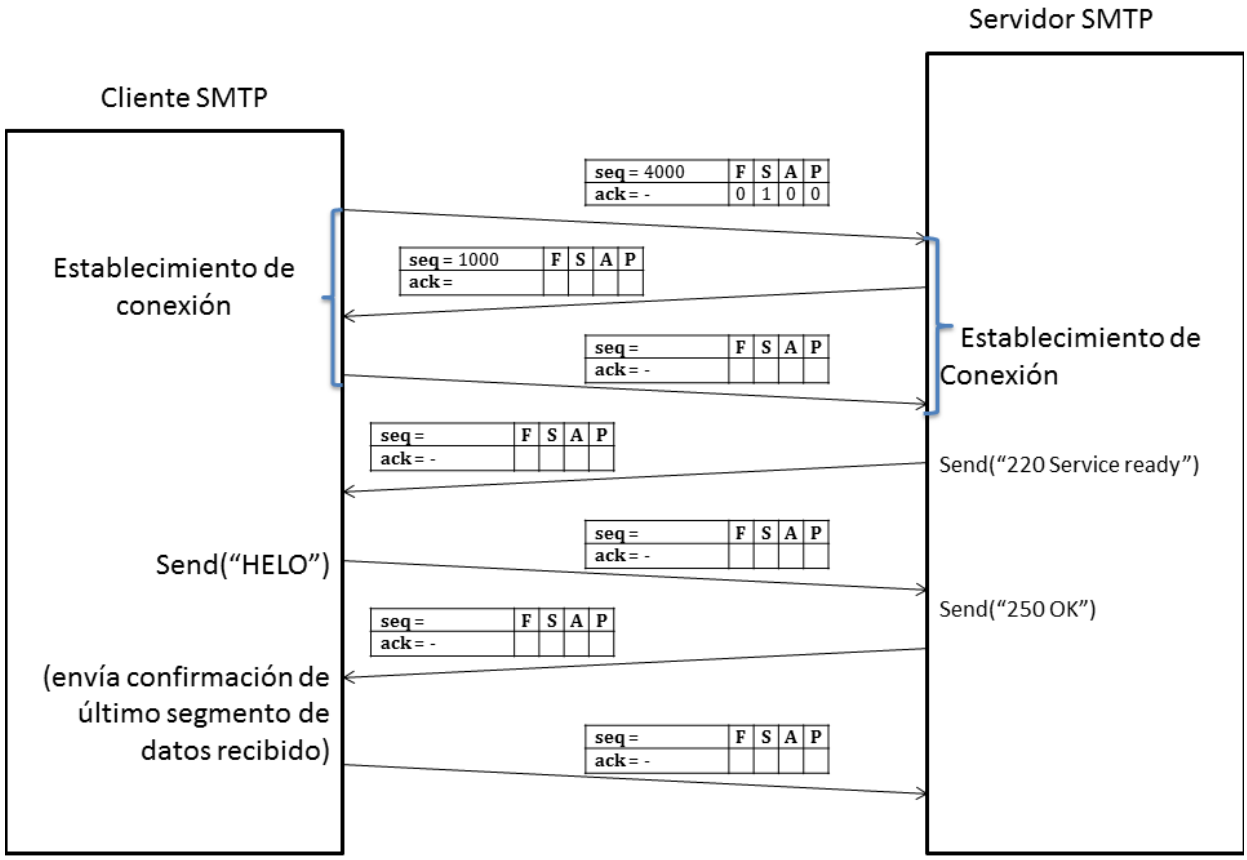
Redes y Sistemas Distribuidos

Relación de problemas del Tema 5

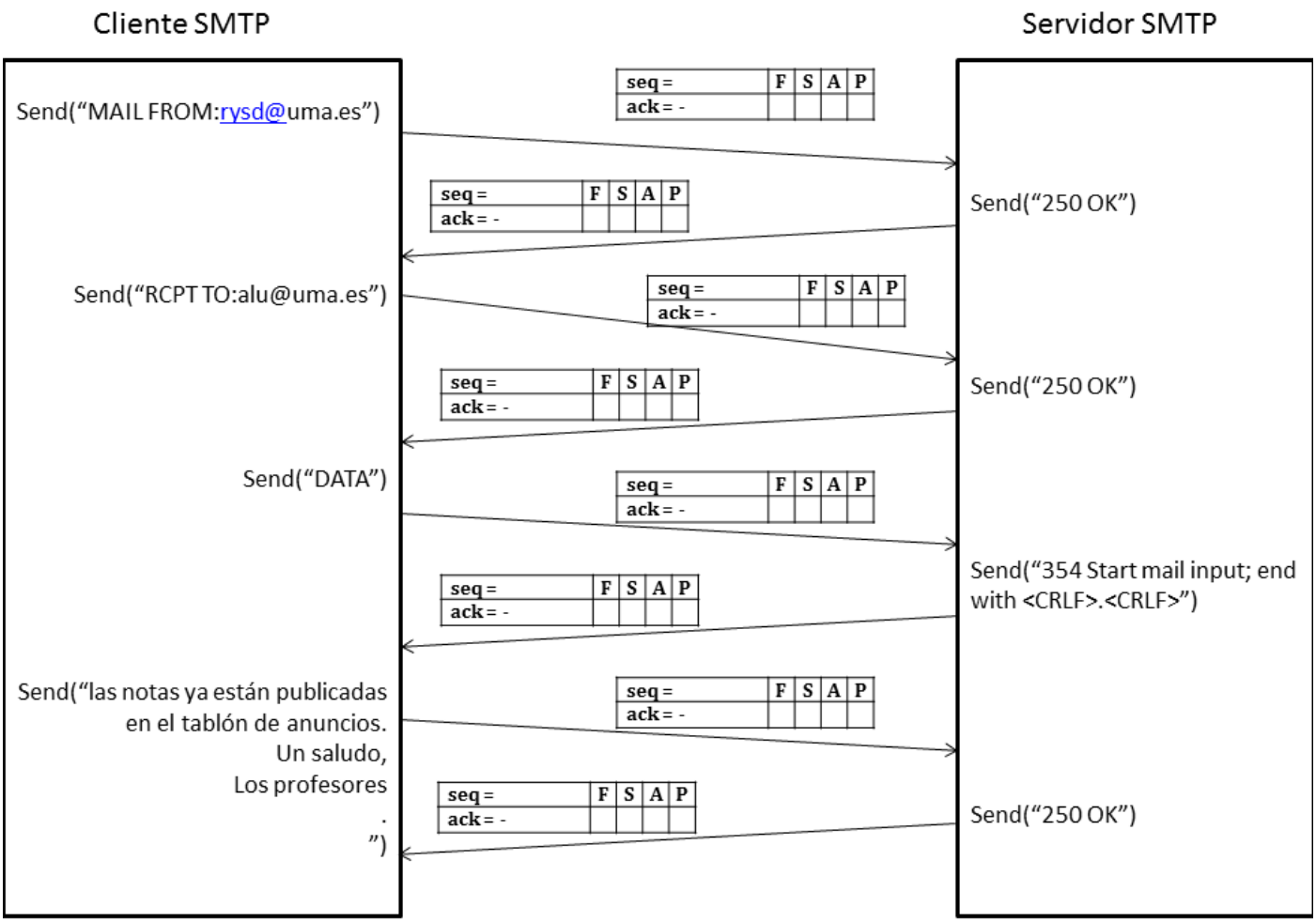
1. <u>Completa</u> el diagrama de secuencia que representa, a nivel de protocolo de transporte (TCP), la interacción entre un cliente y un servidor SMTP para el envío de un mensaje de correo electrónico. Cada línea orden y cada línea del mensaje de correo electrónico se envía en un segmento TCP.

a) Fase de Establecimiento:

1. El Emisor abre una conexión TCP con el Receptor.
2. El Receptor se identifica a sí mismo con "220 Service Ready" (17 bytes).
3. El Emisor se identifica a sí mismo con la orden HELO (4 bytes).
4. El Receptor acepta la identificación del Emisor con "250 OK" (6 bytes).

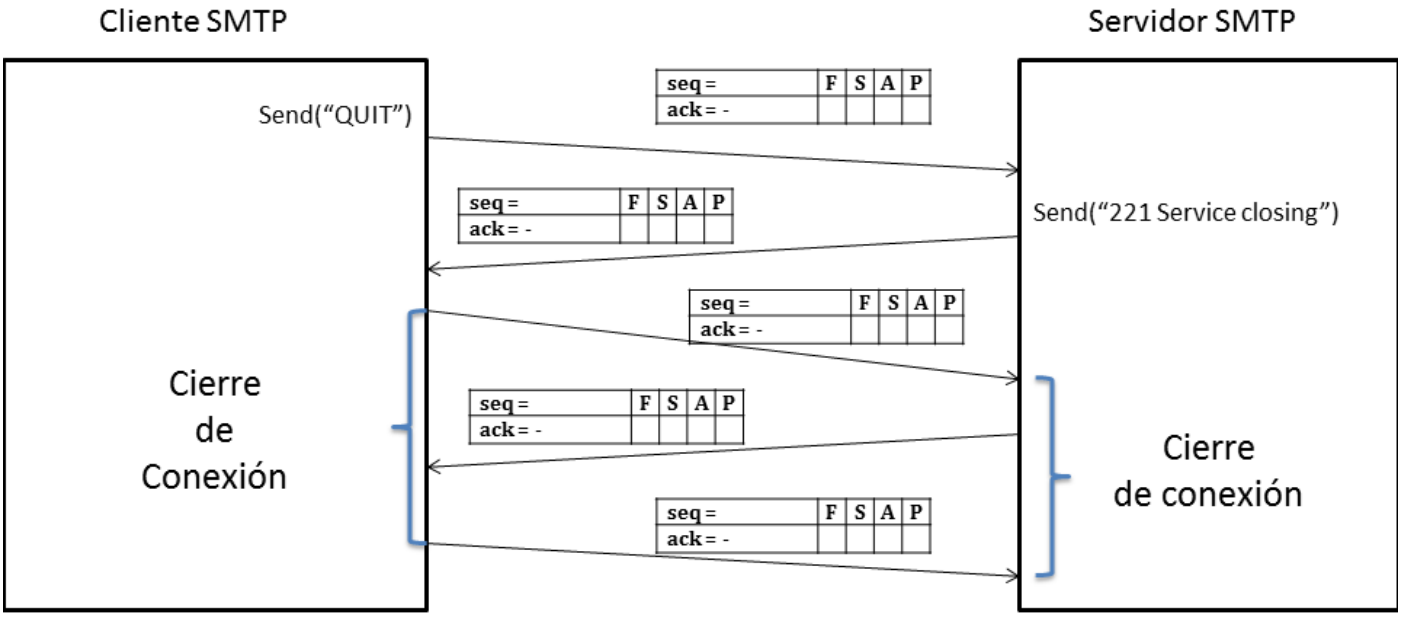


b) Fase de Transferencia de Correo: A continuación, se produce la siguiente transferencia del correo electrónico:



c) Fase de Cierre:

1. El Emisor envía QUIT, e inicia el cierre TCP
2. El Receptor responde "221 2.0.0 Bye", e inicia el cierre TCP



2. Un usuario envía un correo usando un programa similar al implementado en clase de la siguiente forma:

```
alumno@pc512~$ ./myEmailCLient
 From: rysd-profesor@rysd.es
 To: rysd-alumno@rysd.es
 Mensaje:
 Para hacer bien este examen hay que saber del protocolo SMTP.
 ¡Correo enviado con éxito!
alumno@pc512:~$
```

Se pide:

a) Indicar qué mensajes SMTP envía el cliente, en qué orden y con qué contenido:

C0:_____
C1:_____
C2:_____
C3:_____
C4:_____
C5:_____
C6:_____
C7:_____
C8:_____
C9:_____

b) Realizar un diagrama de secuencia TCP correspondiente al envío correcto del correo anterior. En cada mensaje debe indicar el número de secuencia, de confirmación, los flags activos, el tamaño completo del segmento y los datos que lleva. Tenga en cuenta lo siguiente:

- El tamaño de ventana de recepción anunciado en **todos** los segmentos es 50 B
- **Todos** los mensajes del servidor son "200 Vale"
- Los números de secuencia iniciales son 299 para el cliente y 399 para el servidor
- Recordad que el **retorno de carro ocupa dos bytes y los espacios un byte** (incluidlo donde proceda)

c) Anote el diagrama de secuencias con las primitivas sockets de Java que generan los mensajes

Para facilitar el cálculo de los números de secuencia en la siguiente tabla puede encontrar el número de bytes que ocupan los textos enviados y los comandos típicos de SMTP:

Texto	Tamaño	Texto	Tamaño
rysd-profesor@rysd.es	21 B	rysd-alumno@rysd.es	19 B
200 Vale	8 B	Para hacer bien este examen hay que saber...	61 B
HELO	4 B	MAIL FROM:	10 B
RCPT TO:	8 B	DATA	4 B
RSET	4 B	QUIT	4 B
VRFY	4 B	NOOP	4 B

3. Un usuario anónimo (IP: 192.168.177.49) quiere conectarse al servidor de FTP público de la UMA (IP: 150.214.40.67). La comunicación que se establece es la siguiente:

El servidor empieza a escuchar en el puerto estándar de FTP.
El cliente inicia la conexión al servidor.

Mensaje	ID	Origen	Tamaño
`200 FTP Server Ready`	F1	Servidor	22 bytes
`USER anonymous`	C1	Cliente	16 bytes
`331 Anonymous login ok, send your email address`	F2	Servidor	49 bytes
`PASS a@a.es`	C2	Cliente	13 bytes
`230-`	F3	Servidor	6 bytes
`QUIT`	C3	Cliente	6 bytes
`221 Goodbye`	F4	Servidor	13 bytes

El servidor cierra la conexión con el cliente.
El cliente también cierra la conexión con el servidor.

En la tabla anterior, el texto en `courier` son los mensajes enviados, el resto es información adicional: identificador de mensaje (no forma parte del protocolo, solo se utiliza para identificar los mensajes en el ejercicio), origen del mensaje y el tamaño del mismo. Suponiendo la anterior conversación realice el diagrama de secuencia TCP. Suponga que el número de secuencia utilizado por el cliente es 1000 y el del servidor 5000. En cada envío debe aparecer: número de secuencia, número de confirmación (si el bit ACK está activo), flags activos (A para ACK, S para SYN y F para FIN), mensaje enviado (si se envían datos, utilice los IDs de los mensajes en vez de su contenido) y el tamaño del segmento TCP (suponemos que nunca hay opciones).

4. Un usuario se conecta a una máquina mediante FTP y ejecuta una serie de comandos. A continuación, mostramos los mensajes intercambiados entre el cliente y servidor de FTP por la conexión de control.

```
220 FTP Server ready.
USER anonymous
331 Anonymous login ok, send your complete email address as your password
PASS
230-Bienvenido al FTP anonimo de la Universidad de Malaga
230 Anonymous login ok, restrictions apply.
CWD pub
250 CWD command successful
PORT 192,168,1,146,8,2
200 PORT command successful
RETR JAVA.GIF
150 Opening BINARY mode data connection for JAVA.GIF (3412 bytes)
226 Transfer complete
QUIT
221 Goodbye.
```

a) ¿Qué modo (activo o pasivo) se utiliza para el intercambio de datos? ¿Qué puertos utilizan la máquina del usuario y el servidor FTP para realizar dicho intercambio? En ambos casos justifique la respuesta.
b) Realice el diagrama de secuencia TCP relacionado con la conexión de datos de la sesión FTP. Suponga que cada segmento de datos recibido es confirmado inmediatamente por el receptor y que el MSS es 1400. También indique en el diagrama qué entidad TCP representa al servidor FTP y cuál al cliente FTP.

5. Un usuario en un navegador introduce la URL **`http://www.et.com/micasa.html`** y se abre una página cuyo contenido es poco más que una imagen de un teléfono. Examinando las trazas capturadas se observa que se establece una conexión HTTP persistente con el servidor **`www.et.com`** y hace dos peticiones HTTP de la siguiente forma:

Inicio de la conexión

P1: Petición de micasa.html (*71 bytes*)

```
GET /micasa.html HTTP/1.1
Host: www.et.com
Connection: keep-alive
```

R1: Respuesta correcta de micasa.html (*517 bytes*)

```
HTTP/1.1 200 OK
Content-Length: 428
Connection: Keep-Alive
Content-Type: text/html

<PÁGINA 428 Bytes>
```

P2: Petición de telefono.gif (*48 byte)*

```
GET /telefono.gif HTTP/1.1
Host: www.et.com
```

R2: Respuesta correcta de micasa.html (*2675 bytes*)

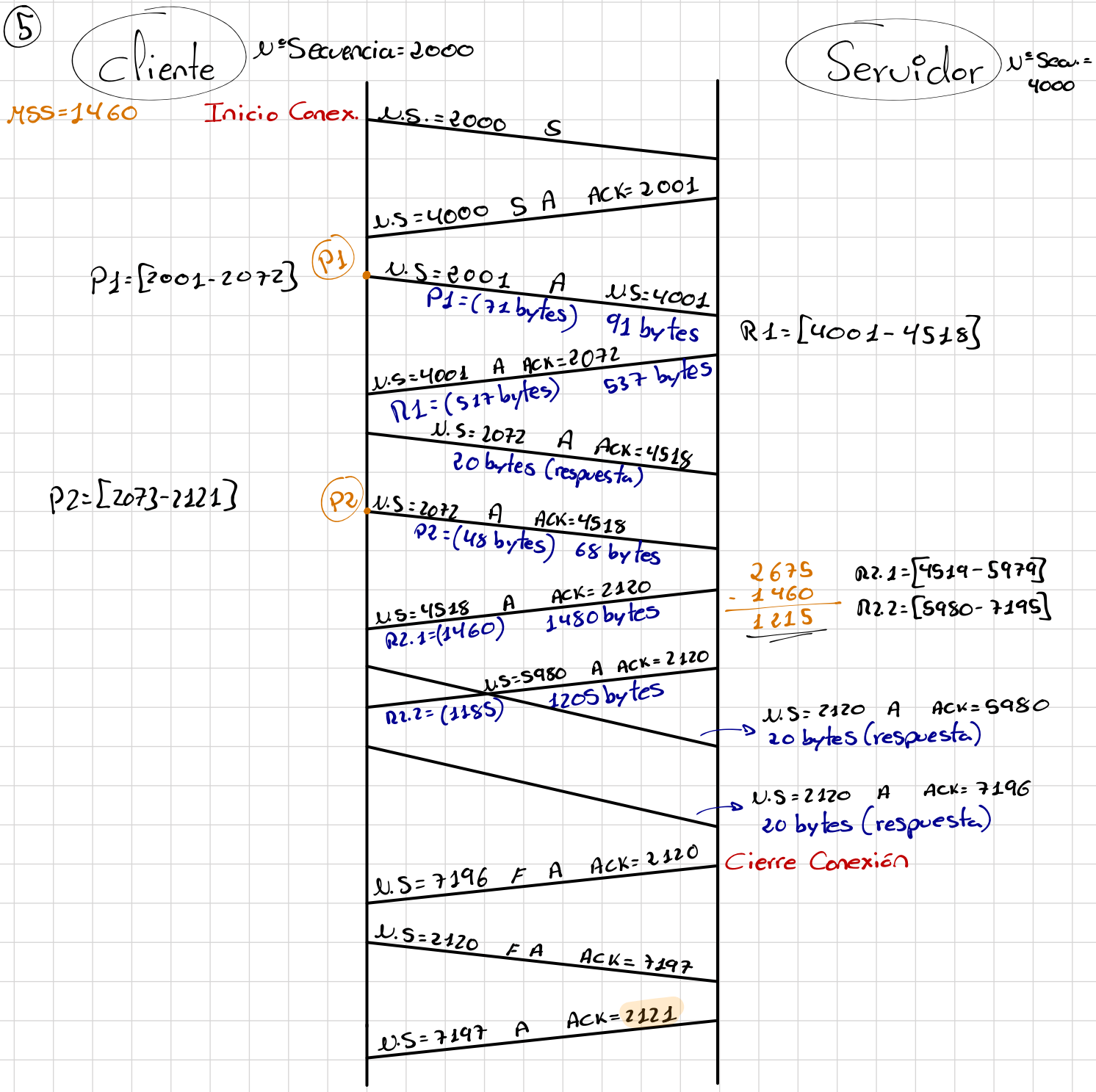
```
HTTP/1.1 200 OK
Content-Length: 2590
Connection: close
Content-Type: image/gif

<IMAGEN 2590 Bytes>
```

Cierre de la conexión (iniciado por el servidor)

Realice el diagrama de secuencia TCP relacionado con dicha conversación. Suponga que el **número de secuencia** utilizado por el **cliente** es **2000** y el del **servidor 4000**, que el **MSS** es **1460** y que **cada segmento** es **confirmado** (usando **piggybacking** cuando sea posible). En cada envío debe aparecer: **número de secuencia** (seq), **número de confirmación** (ack) (si el bit ACK está activo), *flags* activos (A para ACK, S para SYN y F para FIN), mensaje enviado (si se envían datos, se pueden utilizar los IDs -P1, R1, P2, R2- en vez de los datos completos) y el tamaño completo del segmento TCP.

5



Cliente: 2000 + 2 + 71 + 48 = 2121

Número Secuencia Inicial — P1 — P2 — Nº Secuencia Final

Servidor: 4000 + 2 + 517 + 2675 = 7194

6. Un alumno indica en su navegador su página favorita **www.rysd-mola.com/inicio.html** y el navegador genera una conexión persistente y con pipelining para solicitar esa página web tan interesante. El servidor responde de forma exitosa y ofrece la siguiente página:

```
<html>
 <head>
  <title>Redes y Sistemas Distribuidos</title>
 </head>
 <body>
  <h1>Oppppssss nos has pillado en mantenimiento .... vuelva en unos minutos</h1>
  <p><img src="http://www.rsyd-mola.com/hombre-trabajando.png"></p>
<p>
 <img src="http://www.rysd-mola.com/logo-rysd.png">
 <img src="http://www.uma.es/logo-uma.png">
</p>
 </body>
</html>
```

Como siguientes pasos el navegador para completar la visualización completa de la página, empieza a descargarse los recursos (en el orden que aparecen). Sabiendo la siguiente información:

Recurso	Tamaño		Mensaje de la respuesta
	Petición	Respuesta	
www.rysd-mola.com/inicio.html	75 B	215 B	Éxito
www.rsyd-mola.com/hombre-trabajando.png	85 B	2545 B	Éxito
www.rysd-mola.com/logo-rysd.png	80 B	70 B	No modificado
www.uma.es/logo-uma.png	70 B	65 B	No encontrado

Realice el **diagrama de secuencia TCP** completo de la comunicación del cliente con el servidor web **www.rysd-mola.com**, indicando, de forma esquemática, el contenido de cada segmento (número de secuencia y confirmación, flags y si transporta datos, qué datos transporta y de qué tamaño). El MSS es 1460 B, las ventanas de 8 KB y los **números de secuencia** iniciales son **999 (cliente)** y **199 (servidor)**. Se usa el control de congestión visto en clase y piggybacking para las confirmaciones (de hecho, antes de enviar la confirmación espera un poco para intentar confirmar varios segmentos con una única confirmación). Como se mencionó antes el servidor usa una conexión permanente y con pipelining. El cierre es iniciado por el servidor tras enviar el último recurso.

7. Un usuario en un navegador introduce la URL `http://www.got.com/sicansios.html` y se abre una página cuyo contenido es muy oscuro, prácticamente negro, pero consigue pulsar el enlace `http://www.got.com/repetirS8.html`. Examinando las trazas capturadas se observa el siguiente intercambio:

```
GET /sicansios.html HTTP/1.1
Host: www.got.com
Connection: keep-alive
```

Después de recibir respuesta, quiere mantener la conexión

Error por parte del cliente. No se cierra conexión

```
HTTP/1.1 304 Not Modified
Connection: close
```

```
GET /repetirS8.html HTTP/1.1
Host: www.got.com
Connection: keep-alive
```

```
HTTP/1.1 200 OK
Content-Length: 1510
Connection: close
Content-Type: text/html
```

<Fichero HTML de 1510 Bytes>

Complete la siguiente tabla:

Petición/respuesta	Tamaño cabecera HTTP	Tamaño Datos	Tamaño total mensaje HTTP
Petición 1 (P1)	75	0	75
Respuesta 1 (R1)	45	0	45
Petición 2 (P2)	75	0	75
Respuesta 2 (R2)	85	1510	1595

¿Cuántas conexiones TCP se utilizan para esta interacción HTTP?

Realice el diagrama de secuencia TCP relacionado con dicha conversación. En esta implementación del cliente y servidor para HTTP siempre usan como valores de secuencia iniciales los siguientes: el cliente 999 y el servidor 4999. Sabemos que el MSS es 1460 bytes, que cada segmento es confirmado (usando piggybacking cuando sea posible) y que los extremos siempre informan al otro extremo que su ventana de recepción es de 4 KB. En cada envío debe aparecer: número de secuencia (seq), número de confirmación (ack) (si el bit ACK está activo), flags activos (A para ACK, S para SYN y F para FIN), mensaje enviado (si se envían datos, se pueden utilizar los IDs -P1, R1, P2, R2- en vez de los datos completos) y el tamaño completo del segmento TCP.

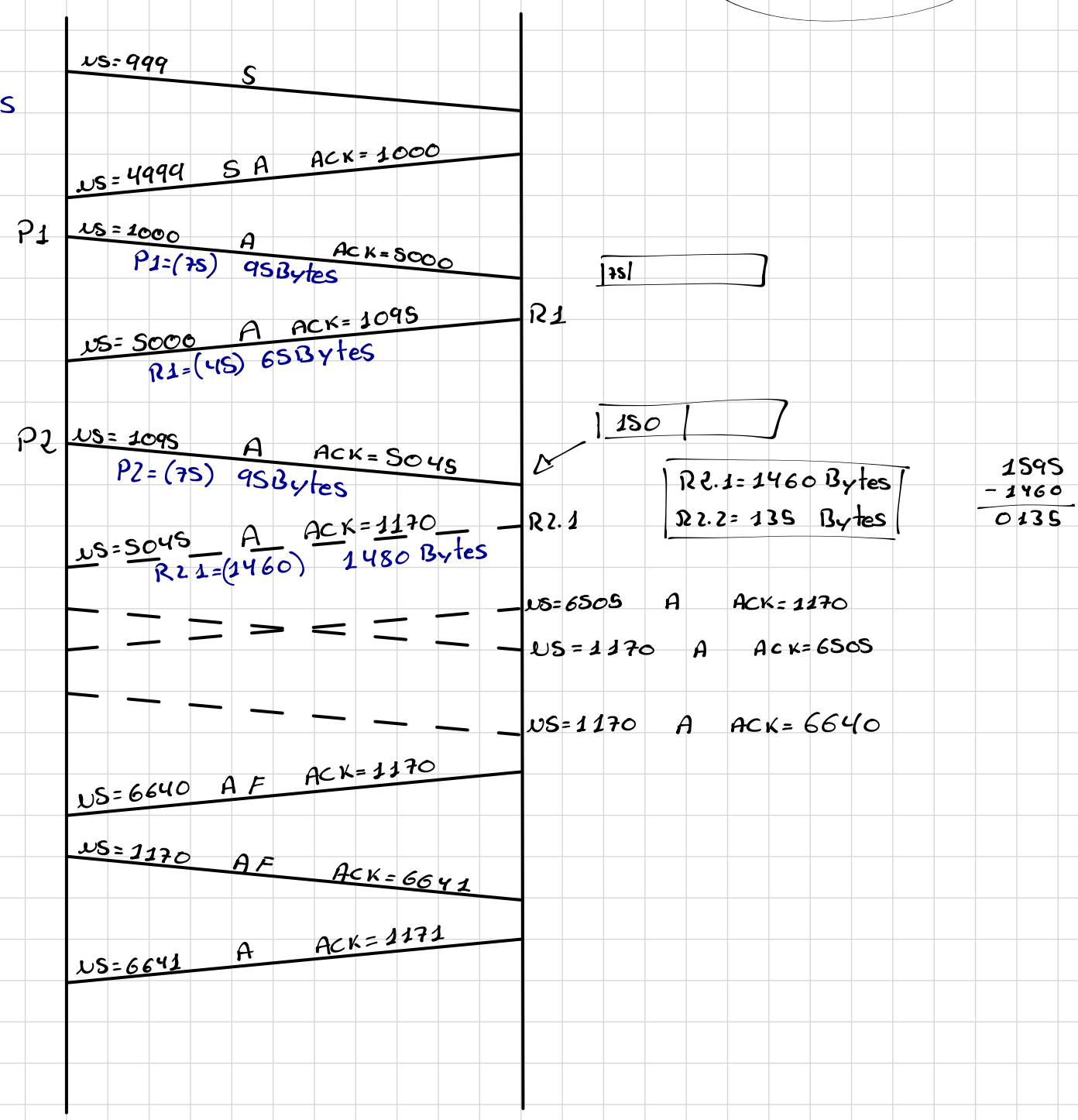
MSS: Tamaño Máx. Segmento
Piggyback: Incorporar confirma. en sentido contrario, siempre que se pueda.

Cliente = 999

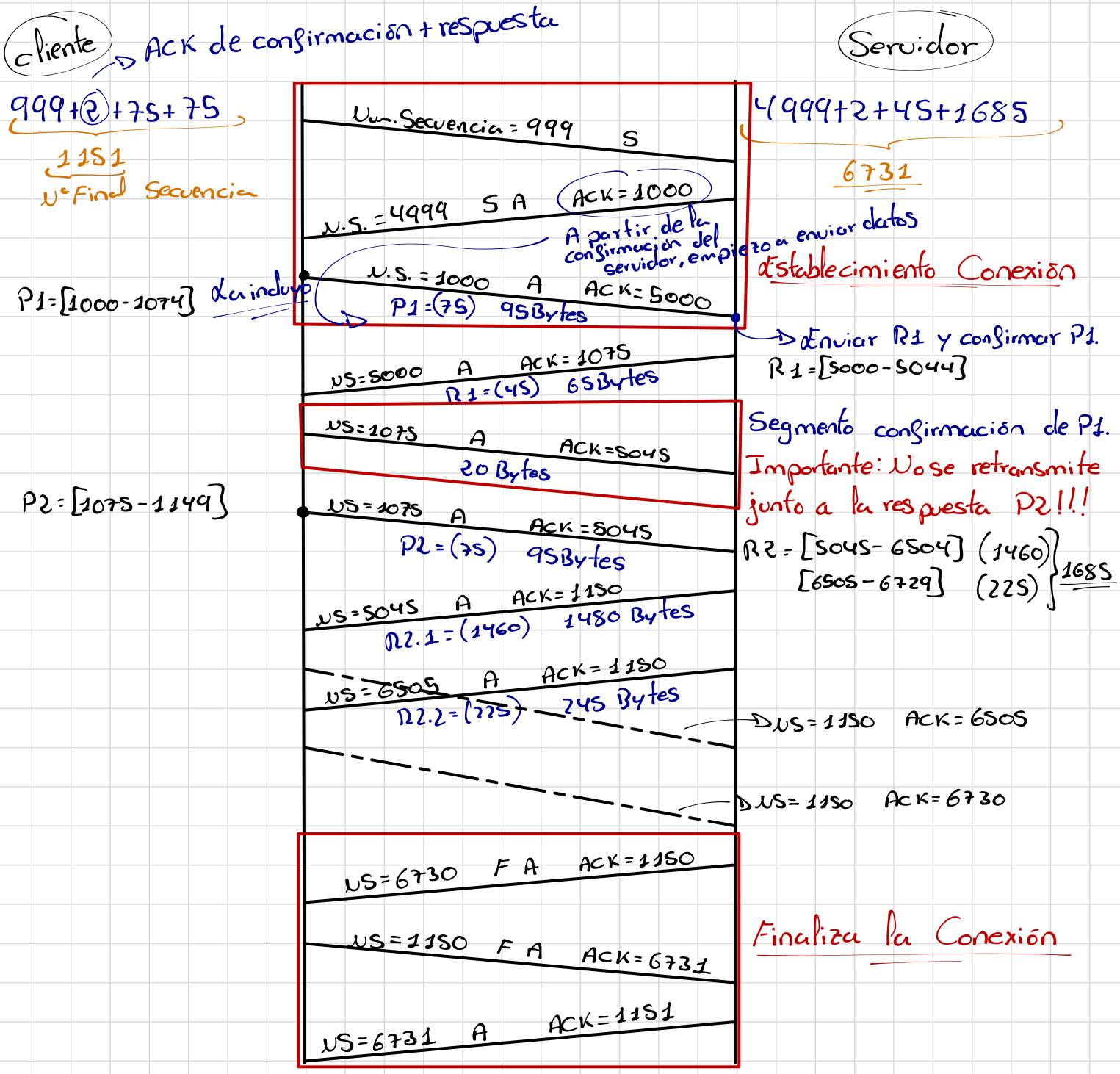
Servidor = 4999

MSS = 1460 Bytes

Ventana de Recepción = 4000 Bytes



- Ejercicio examen



Un usuario en un navegador introduce la URL `http://www.got.com/sicansios.html` y se abre una página cuyo contenido es muy oscuro, prácticamente negro, pero consigue pulsar el enlace `http://www.got.com/repetirS8.html`. Examinando las trazas capturadas se observa el siguiente intercambio:

```
GET /sicansios.html HTTP/1.1
Host: www.got.com
Connection: keep-alive
```

```
                                                    HTTP/1.1 304 Not Modified
                                                      Connection: keep-alive
```

```
GET /repetirS8.html HTTP/1.1
Host: www.got.com
Connection: keep-alive
```

```
                                                    HTTP/1.1 200 OK
                                                  Content-Length: 1510
                                                    Connection: close
                                                  Content-Type: text/html
                                                  <Fichero HTML de 1600 Bytes>
```

Complete la siguiente tabla:

Petición/respuesta	Tamaño cabecera HTTP	Tamaño Datos	Tamaño total mensaje HTTP (datos enviados por TCP)
Petición 1 (P1)	75		75
Respuesta 1 (R1)	45		45
Petición 2 (P2)	75		75
Respuesta 2 (R2)	85	1600	1685

Nº Secuencias:
- Cliente: 999
- Servidor: 4999

Nota: En TCP se intercambian mensajes HTTP (DNS se intercambian en UDP)

↳ No se retransmiten preguntas y respuestas DNS.

Cliente

1899+2+2+2190

4091

MSS = 1500 Bytes

Ventana = 2000 Bytes

Vent. Cong. = 1500 Bytes

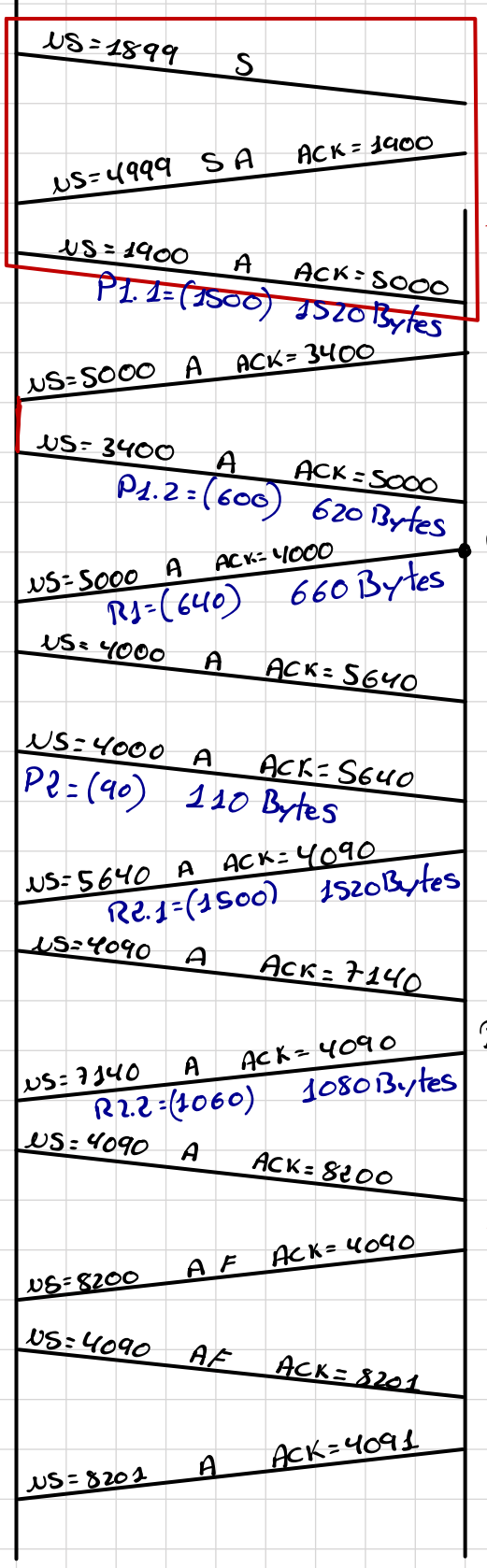
P1.1 = 1500
P1.2 = 600
} 2100

Igual al "MSS"

Servidor

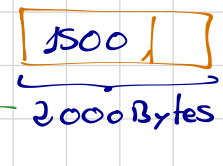
4999+2+640+2560

8201



Inicio Conexión

No envío el segundo fragmento de 600 bytes (aun habiendo piggybacking porque la suma de los fragmentos supera los 2000 de ventana receptora.



Por tanto, espero ACK de confirmación para liberar espacio en la ventana receptora.

R2.1 = 1500 Bytes

R2.2 = 1060 Bytes

Inicio Fin Conexión

En la siguiente tabla se muestran los tamaños de cada mensaje (a nivel de aplicación):

Petición DNS	120 B	Respuesta DNS	240 B
Petición HTTP1 (P1)	2100 B	Respuesta HTTP1 (R1)	640 B
Petición HTTP2 (P2)	90 B	Respuesta HTTP2 (R2)	2560 B

Realice el **diagrama de secuencia TCP** relacionado con dicha conversación. En esta implementación del cliente y servidor para HTTP siempre usan como valores de secuencia iniciales los siguientes: el cliente 1899 y el servidor 4999. Sabemos que el *MSS* es *1500* bytes, que cada segmento es confirmado (usando piggybacking cuando sea posible), que los extremos siempre informan al otro extremo que su ventana de recepción es de 2000 B, no usa el algoritmo de Nagle pero sí emplea el mecanismo de control de congestión visto en clase. En cada envío debe aparecer: número de secuencia (seq), número de confirmación (ack) (si el bit ACK está activo), flags activos (A para ACK, S para SYN y F para FIN), mensaje enviado (si se envían datos, se pueden utilizar los IDs -P1,R1,...- en vez de los datos completos) y el tamaño completo del segmento TCP.