1 Applications (6 points)

The following presents the messages exchanged between a user and a server:

```
Trying 10.1.1.13...
Connected to mail.etu1.plateforme.lan.
Escape character is '^]'.
+OK Hello there.
USER etudiant
+OK Password required.
PASS mmqos
+OK logged in.
LIST
+OK POP3 clients that break here, they violate STD53.
 1 690
 2 752
 3 722
 4 714
RETR 4
+OK 714 octets follow.
Return-Path: <etudiant@mail.etu1.plateforme.lan>
X-Original-To: etudiant@mail.etu1.plateforme.lan
Delivered-To: etudiant@mail.etu1.plateforme.lan
Received: from [10.1.1.11] (unknown [10.1.1.11])
  by mail.etu1.plateforme.lan (Postfix) with ESMTP id DB8EB1C467
  for <etudiant@mail.etu1.plateforme.lan>; Thu, 16 Oct 2014 12:04:20 +0000 (UTC)
From: etudiant sur la VM3 <etudiant@mail.etu1.plateforme.lan>
To: etudiant@mail.etu1.plateforme.lan
Content-Type: text/plain
Date: Thu, 16 Oct 2014 12:24:01 +0000
Message-ID: <1413462241.14505.2.camel@1vm1.etu1.plateforme.lan>
Mime-Version: 1.0
X-Mailer: Evolution 2.30.3
Content-Transfer-Encoding: 7bit

Bravo !!!
```

1. What is the user doing? What are the application layer protocols employed by the user?

2. Give the command line employed by the user to connect to the server.

3. What is purpose of the LIST command? What does "3 722" mean in the response?

4. What does the user want to do when typing "RETR 4"?

5. In the message retrieved by the user, indicate four header lines that have been added by the mail server.

6. The subject of the message entered by the user has been re-encoded by the email application. What is the original character-encoding scheme of the subject? What is the encoding scheme used for re-encoding? What is the format of the re-encoded text?

7. What is the content of the message?

8. If the user wants to delete this message, write down the necessary command.
2 Transport layer (7 points)

To illustrate a part of an exchange of TCP segment between host A and B, we use the following formalism:

- A sloping arrow from A’s timeline to B’s timeline (or going the other direction) represents a segment (a TCP packet), with the number of bytes of application-layer data being shown in the pentagon attached to the arrow.

- Each segment is annotated by a label attached by a thin line, containing the following information:
  - the sequence number of the segment (Seq.)
  - the acknowledgment number of the segment (Ack.)
  - the value of the advertised Receive Window (RWIN).

Thus, in the example below, we have:

- A segment of 1200 bytes of data going from A to B. Its sequence number is 9000, its acknowledgment number is 8300 and its advertised Receive Window is 3000.
- A second segment, zero size, is emitted from B to A before receiving the previous one. Its sequence number is 8300, its acknowledgment number is 7000 and the advertised Receive Window is 4000.

```
   A                   B
  9000  8300  1000    Seq. Ack. RWin.
   1200 B
   8300  7000  4000
```

Explain the probable reason for the temporary cessation of transmission of segments from A to B around the middle of this exchange (justify your answer).

Fill in the blanks for sequence numbers, acknowledgement numbers, and Receive Window sizes in the following portion of a TCP exchange that has already begun.

```
   A                   B
       Seq. Ack. RWin.
   1000  7000  2000
   1000  7000  2000
   1000  7000  2000
   1000  7000  2000
   1000  7000  2000
```

---

The exam consists of 3 sheets recto/verso, including the questions and spaces in which to write your answers. You must hand back only those sheets at the end of the exam. To ensure anonymity, you must not write your name on the exam. Be absolutely sure that the anonymous ID number assigned to you by the exam supervisors appears on each such sheet.

You must write your answers in the frames provided for that purpose.
3 Multi-protocol analysis (7 points)

1. Decode the following frame. Draw and write directly on it. Carefully delimit each protocol field and provide an accurate interpretation of its value. You may refer to the Appendix (page 7) to help in the analysis.

```
0000 7c d1 c3 dc ed 00 00 5e 00 01 4e 00 00 45 00 |...T...^..N..E.
0010 01 3c d8 13 00 00 3f 11 00 68 84 e3 4a 22 84 e3 .<....?..h..J''..
0020 4e 4d 00 35 de 79 01 28 1e d5 35 89 81 80 00 01 NM.5.y.(..5....
0030 00 04 00 02 00 04 03 77 77 77 07 6c 65 6d 6f 6e ........www.lemon
0040 64 65 02 66 72 00 00 01 00 01 c0 0c 00 05 00 01 de.fr.......... 
0050 00 00 37 ca 00 2f 03 77 77 77 07 6c 65 6d 6f 6e ..7/..www.lemon
0060 64 65 02 66 72 0e 32 3d 30 3d 32 3d 37 31 64 2d de.fr.2-01-271d-
0070 30 30 30 64 03 63 64 78 07 63 65 64 65 65 78 79 73 000d.cdx.cedexis
0080 03 6e 65 74 00 c0 2c 00 05 00 01 00 00 00 0e 00 .net............
0090 17 03 77 61 63 64 03 33 36 30 34 0b 65 64 67 65 63 ..vac.3604.edgee
00a0 61 73 74 63 64 6e c0 56 c0 67 00 05 00 01 00 00 astcdn.V.g......
00b0 09 1a 00 0a 03 67 70 31 03 77 61 63 c0 70 c0 8a .....gpl.vac.p...
```

2. The frame you have just decoded was sent in response to another frame. Below, write out that frame in hexadecimal format, using two hexadecimal digits per box. (If you cannot deduce the value in a box, leave it blank.

```

<table>
<thead>
<tr>
<th>Byte num</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0010</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0020</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0030</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0040</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0050</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

3. Which action from the user may have initiated this message exchange (justify)?
Appendix

Ethernet Frame Layout
without preamble/CRC:

- 46 bits: 46 bytes
- 32 bits: 32 bytes
- 64 bits: 64 bytes
- 128 bits: 128 bytes

<table>
<thead>
<tr>
<th>Destination</th>
<th>Source</th>
<th>Type</th>
<th>Data</th>
<th>... Options</th>
<th>... Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Address</td>
<td>Address</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

IPv4 Packet Layout:

- 40 bits: 40 bytes
- 64 bits: 64 bytes
- 128 bits: 128 bytes

<table>
<thead>
<tr>
<th>Ver</th>
<th>IHL</th>
<th>TOS</th>
<th>Total Length</th>
<th>Ident</th>
<th>Flags</th>
<th>Offset</th>
<th>TTL</th>
<th>Protocol</th>
<th>Checksum</th>
<th>... Data</th>
</tr>
</thead>
</table>

Ver = IP Version
IHL = IP Header Length (32 bits words)
TOS = Type of service
Total Length = Total length of the IP packet (in bytes)
ID = Identification
Flags = Fragmentation flags
Offset = Fragment Offset
Checksum = Value to multiply by 8 bytes
TTL = Time To Live (hop count)

A few protocol fields:
1 = ICMP 33 = DCCP
2 = IPv 64 = IP Encapsulation
6 = TCP 89 = HTTP
8 = UDP 123 = SCTP
17 = UDP ...

ICMP Datagram Layout

- 40 bits: 40 bytes
- 80 bits: 80 bytes

<table>
<thead>
<tr>
<th>Type</th>
<th>Code</th>
<th>Checksum</th>
<th>... Variables</th>
<th>... Options</th>
<th>... Data</th>
</tr>
</thead>
</table>

A few ICMP types:
0 = Echo request
1 = Destination unreachable
3 = Redirection
8 = Echo response
11 = Time exceed

TCP Segment Layout

- 40 bits: 40 bytes
- 80 bits: 80 bytes

<table>
<thead>
<tr>
<th>THL</th>
<th>Flags</th>
<th>Acknowledgment Number</th>
<th>Source Port</th>
<th>Destination Port</th>
<th>... Data</th>
</tr>
</thead>
</table>

THL = TCP header length on 4 bits (32 bits words)
Flags = series of options encoded as:
0 byte 00 = end of options (if needed)
1 byte 01 = no operation
1 byte 10 = set timestamps
1 byte 11 = the value of the option on L-3 bytes

A few protocol fields:
1 = SDP 33 = DCCP
2 = TCP 69 = HTTP
8 = UDP 123 = SCTP
17 = UDP ...

UDP Datagram Layout

- 40 bits: 40 bytes
- 80 bits: 80 bytes

<table>
<thead>
<tr>
<th>Source Port</th>
<th>Destination Port</th>
<th>... Data</th>
</tr>
</thead>
</table>

ICMP Messages Layout

- 32 bits: 32 bytes
- 64 bits: 64 bytes
- 128 bits: 128 bytes

<table>
<thead>
<tr>
<th>Ident</th>
<th>Type</th>
<th>Code</th>
<th>Checksum</th>
<th>... Variables</th>
<th>... Options</th>
<th>... Data</th>
</tr>
</thead>
</table>

DNS Messages Layout

- 40 bits: 40 bytes
- 80 bits: 80 bytes

<table>
<thead>
<tr>
<th>Id</th>
<th>Flag</th>
<th>QueryName</th>
<th>QueryClass</th>
<th>QueryType</th>
<th>QueryClass</th>
<th>QueryLength</th>
<th>QueryData</th>
<th>... Options</th>
<th>... Data</th>
</tr>
</thead>
</table>

A few ports associated services:

- 38 bits: 38 bytes
- 76 bits: 76 bytes

<table>
<thead>
<tr>
<th>From Port</th>
<th>To Port</th>
<th>Service</th>
<th>Port Number</th>
</tr>
</thead>
</table>

Options:

- 1 byte = operation (noop, options, etc.)
- 0x00 = end of options
- 0x01 = no operation
- 0x02 = set timestamps
- 0x04 = the value of the option on L-3 bytes

Don't hand back this sheet!