Programming Assignments

COEN 233 Computer Networks - Fall Quarter 2017

General Guidelines

- Programming projects are individual assignments, each student should write his/her own code.
- Each project requires a demo, during which the student should explain how the code works.
- Demos are part of the grade. The student only will receive full credit if demo has correct results.
- In addition to the demo, each student should submit the source code, input/output files, and a README.txt file containing instructions on how to compile and run your source code.
- The program should be turned in on or before deadline; demo must be performed on or before the deadline but after the program files has been turned in.

Program assignment 1:

Client using customized protocol on top of UDP protocol for sending information to the server.

One client connects to one server.

Design a protocol with the following primitives:

Start of Packet identifier ….. 0XFFFF
End of Packet identifier ….. 0XFFFF
Client Id…….Maximum 0XFF (255 Decimal)
Length …….. Maximum 0XFF (255 Decimal)

Packet Types:

DATA…. …….…….0XFFF1
ACK (Acknowledge)…….0XFFF2
REJECT…….0XFFF3

Reject sub codes:

REJECT out of sequence……0XFFF4
REJECT length mismatch……...0XFFF5
REJECT End of packet missing……..0XFFF6
REJECT Duplicate packet……...0XFFF7

Data Packet Format:

Bytes:	2	1	2	1	1	255	2
	Start of Packet id	Client ID	DATA	Segment No	Length	Payload	End of Packet id

ACK (Acknowledge) Packet Format:

Bytes: 2	1	2	1	2
Start of Packet id	Client ID	ACK	Received Segment No	End of Packet id

REJECT Packet Format:

Bytes: 2	1	2	2	1	2
Start of Packet id	Client ID	REJECT	Reject sub code	Received Segment No	End of Packet id

Procedure:

Client sends five packets (Packet 1, 2, 3, 4, 5) to the server.
The server acknowledges with ACK for the correct packet from client by sending five ACKs.

Client then sends another five packets (Packet 1, 2, 3, 4, 5) to the server, emulating one correct packet and four packets with errors.
The server acknowledges with ACK for the correct packet from client, and with corresponding Reject sub codes for packets with errors.

The client will start an **ack_timer** at the time the packet is sent to server, if the ACK (Acknowledge) for each packet has not been received during **ack_timer** period by client before expiration of timer then client should retransmit the packet that was sent before.
The timer can be set at 3 seconds (recommended) and a retry counter should be used for resending the packet. If the ACK for the packet does not arrive before the timeout, the client will retransmit the packet and restart the **ack_timer**, and the **ack_timer** should be reset for a total of 3 times.
If no ACK was received from the server after resending the same packet 3 times, the client should generate the following message and display on screen:
"Server does not respond".

Error handling:

NOTE: All four error handling messages below should be simulated and displayed on the screen, the error response messages should be included in a (.pdf, .png, .jpg) file and turned in with your source code.

ACK error handling:
If the ACK timer expires and the ACK from server has not been received by client, an error message should be displayed on the screen by client prior to resending the packet.

Reject error handling with sub code:

Case-1: An error message should be displayed on the screen when the received packet at server is not in sequence with expected packet from client; an error message should be generated by server and sent to client.

For example, if server receives packet 0, 1 and then 3, packet 3 is out of sequence because the server is expecting 2 after receiving 1. The server will not increment the expected sequence number until packet 2 has been received.

Case-2: The server receives a packet which its length field does not match the length of data in the payload field, an error message should be generated by server and sent to the client.

For example, if the length field of a received packet indicates the data payload is 125 bytes, but the actual payload is only 12 bytes, this packet has a length mismatch error.

Case-3: The server receives a packet which does not have the End of Packet Identifier an error message should be generated by server and sent to the client

For example, if the last byte of the packet is xFFF0, this packet has a missing end of packet identifier error.

Case-4: The server receives a duplicated packet (sequence number), an error message should be generated by server and sent to the client.

For example, if server receives packet 0, 1 and then 1 again, the second packet 1 is a duplicate packet, the server will not increment the expected sequence number.

Program assignment 2:

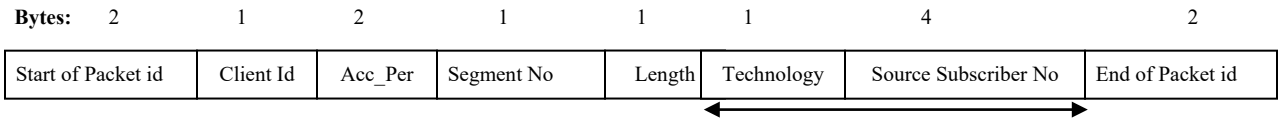
Client using customized protocol on top of UDP protocol for requesting identification from server for access permission to the network.

One client connects to one server.

The client requests access information from server; the server will verify the validity of the request and will respond accordingly.
The communication between client and server will use the **ack_timer,** which was described in the first program assignment.

For the program assignment 2 you can imagine client's software module is acting in behalf of a cell phone.
The client's software module sends the request for identification of their devices in a packet to server; the packet has the following information:

Access permission request packet format:



Start of Packet identifier ….. 0XFFFF
End of Packet identifier. ….. 0XFFFF
Client Id……...Maximum 0XFF (255 Decimal)
Acc_Per (Access Permission)…….0XFFF8
Length …….Maximum 0XFF (255 Decimal)
Source Subscriber No……..Maximum 0XFFFFFFFF (4294967295 Decimal)

Technology:
2 G……...02
3 G……...03
4 G……...04
5 G……...05

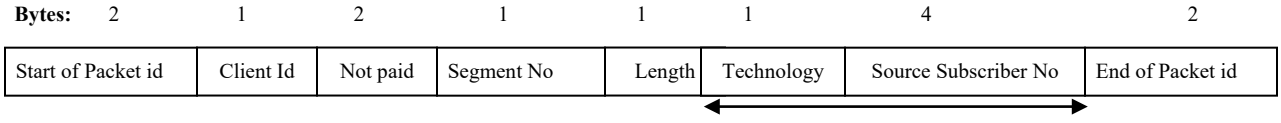
Server opens and reads a file named "**Verification_Database.txt**", the contents of the file will be saved on the server, which contains the Subscriber's Number, Technology, and payment status (paid = 1, not paid = 0).

Verification_Database Format:

Subscriber Number	Technology	Paid	
408-554-6805	04	1	(1 paid)
408-666-8821	03	0	(0 not paid)
408-680-8821	02	1	(1 paid)

After verifying the content of **Identification request packet** with the content of the **Verification_Database.txt** file, one of the following messages will be generated by the server:

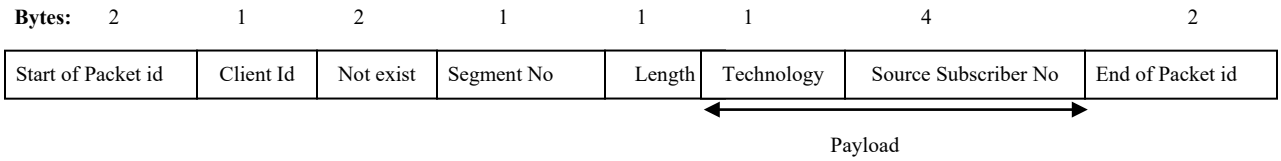
Subscriber has not paid message:



Start of Packet identifier ….. 0XFFFF
End of Packet identifier. ….. 0XFFFF
Client Id…….Maximum 0XFF (255 Decimal)
Not paid…. ……..….0XFFF9
Length …….. Maximum 0XFF (255 Decimal)
Source Subscriber No……..Maximum 0XFFFFFFFF (4294967295 Decimal)

Technologies:
2 G……...02
3 G……...03
4 G……...04
5 G……...05

Subscriber does not exist on database message:

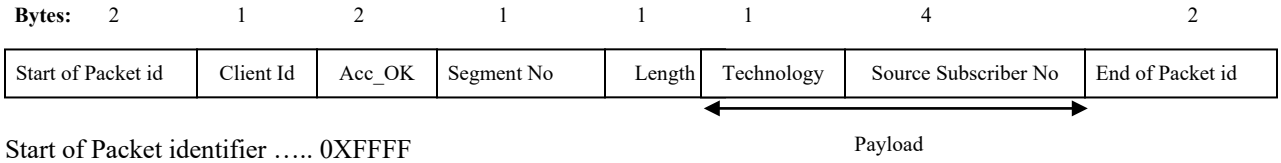


Consider two cases in which the subscriber does not exist:
1. Subscriber number is not found
2. Subscriber number is found, but the technology does not match

Start of Packet identifier ….. 0XFFFF
End of Packet identifier …... 0XFFFF
Client Id…….Maximum 0XFF (255 Decimal)
Not exist…. ……..…..0XFFFA
Length …….. Maximum 0XFF (255 Decimal)
Source Subscriber No……Maximum 0XFFFFFFFF (4294967295 Decimal)

Technologies:
2 G……...02
3 G……...03
4 G……...04
5 G……...05

Subscriber permitted to access the network message:



Start of Packet identifier ….. 0XFFFF
End of Packet identifier …... 0XFFFF
Client Id…….Maximum 0XFF (255 Decimal)
Access_OK …….…..….0XFFFB
Length …….. Maximum 0XFF (255 Decimal)
Source Subscriber No……Maximum 0XFFFFFFFF (4294967295 Decimal)

Technologies:
2 G……...02
3 G……...03
4 G……...04
5 G……...05