Telematic Applications

Mandatory assignment - Chat server

This document describes the protocol to be implemented during the term. This document is intentionally written in English.

Objectives: Design and development of a SERVER that conforms to the protocol described in this document. The server can be written in any language.

Server behavior

- The server should accept multiple connections and serve them concurrently
- Should use the TCP protocol stack and listen in the port 8888
- The server **should never close a connection in the absence of errors**.
- TCP Connections can only be closed by clients.
- The server should flag the client as logged out if the client closes the connection before sending a LogOut message

Message structure

Field	Protocol Tag	Length	cmd	msgId	destination_id	origin_id	Payload
Size	4 bytes	4 bytes	2 bytes	4 bytes	4 bytes	4 bytes	0-256 bytes
Default	0x11223344						

Protocol tag: the protocol tag should be present in every message with the value in hexadecimal 0x11223344

Length: this field indicates the size of the payload only. Since the payload size may range from 0 to 256 its value should be contained in [0,256]

cmd: the identifier of the command that is delivered. Should be one of the following:

cmd_LogIn 0;
cmd_LogInResponse 1;
cmd_TextSend 2;
cmd_TextSendResponse 3;
cmd_LogOut 4;
cmd_LogOutResponse 5;

msgId: is an identifier for a transaction that is generated by the client and should be respected by the server (the server does not have to generate it).

destination_id and origin_id: is an identifier for the endpoint that receives (destination) or sends (origin) the message. The message Id for the server is always 0 and clients can use values from 1 to 1000.

Payload: this field contains the information for the command in use and its size and format depends on the command. A detailed view of the payloads is given in the following lines.

Login transaction

After opening a connection with the server a client should log in order to use the chat. The login process is started by the client by sending the LogIn message to the server. And the server should answer with a LogInResponse message.

The Login message uses the following payload:

Field	id	password
Size	4 bytes	8 bytes
Range	[1-1000]	Ascii text containing "pass" and the number in text format adding '0' to the beginning until it is 4 characters long. e.g. "pass0001" for id=1, "pass0234" for id=234

The login response uses the following payload:

Field	status
Size	7 bytes
Range	Asci text containing the word "success" (for a successful log in) or "failure" otherwise

Example of successful log in:

Login message from the client to the server:

```
MSG (length  34) 1 -> 0 | payLength 12 | cmd 0(cmd_LogIn) | msgid 1723197940 | Payload
         @id 1
         @password pass0001
|   Tag  | length | c  |  msgId |  dest  |  orig  | payload ->
|11223344|0000000c|0000|66b5e9f4|00000000|00000001|00 00 00 01 70 61 73 73 30 30 30 31
                                                    p  a  s  s  0  0  0  1
```

Login response sent from the server to the client:

```
MSG (length  29) 0 -> 1 | payLength 7 | cmd 1(cmd_LogInResponse) | msgid 1723197940 | Payload
         @status success
|   Tag  | length | c  |  msgId |  dest  |  orig  | payload ->
|11223344|00000007|0001|66b5e9f4|00000001|00000000|73 75 63 63 65 73 73
                                                    s  u  c  c  e  s  s
```

Please note the following regarding LogIn message:

- The id the origin field should match the id in the payload
- The id in the destination should be 0 since the destination is the server
- The msgId (66b5e9f4) generated by the client in the Login message should be used by the server in the Login Response message

Logout transaction

The logout message can be sent by the client after a successful login, otherwise the server should answer with error. It is started by the client by sending the LogOut message to the server. And the server should answer with a LogOutResponse message.

The LogOut message has an empty payload:

The logout response uses the following payload:

Field	status
Size	7 bytes
Range	Asci text containing the word "success" (for a successful log in) or "failure" otherwise

Example of successful log out:

Logout message from the client to the server:

```
MSG (length  22) 1 -> 0 | payLength 0 | cmd 4(cmd_LogOut) | msgid -287320318 | Payload
        @nopayload
|   Tag  | length | c  |  msgId |  dest  |  orig  | payload ->
|11223344|00000000|0004|eedfd702|00000000|00000001|
```

Logout response sent from the server to the client:

```
MSG (length  29) 0 -> 1 | payLength 7 | cmd 5(cmd_LogOutResponse) | msgid -287320318 | Payload
        @status success
|   Tag  | length | c  |  msgId |  dest  |  orig  | payload ->
|11223344|00000007|0005|eedfd702|00000001|00000000|73 75 63 63 65 73 73
                                                    s  u  c  c  e  s  s
```

Please note the following regarding LogOut message:

- The id the origin field should match the id of the client
- The id in the destination should be 0 since the destination is the server
- The msgId (eedfd702) generated by the client in the Logout message should be used by the server in the Logout Response message

Text send transaction

A text send transaction involves two clients, the one sending and the one receiving, and the server that mediates.

The sender (client) sends a TextSend message to the server and uses as origin its origin id and as destination the id of the client that should receive the message (note in this case the destination is not the server). Once the server receives the TextSend message, it first answers the sender with a TextSend response message. After that, it forwards the TextSend message received from the sender to the appropriate destination (client that receives).

The TextSend message uses the following payload:

Field	text
Size	variable
Range	Ascii text containing the message to be delivered

The TextSendResponse uses the following payload:

Field	status
Size	7 bytes
Range	Asci text containing the word "success" (for a successful log in) or "failure" otherwise

Consider the following Text Send example:

```
MSG (length  53) 2 -> 3 sock id(2)| payLength 31 | cmd 2(cmd_TextSend) | msgid 1437806877 | Payload
         @text Hi there. Sending to 3 from 2!!
|   Tag  | length | c  |  msgId |  dest  |  orig  | payload ->
|11223344|0000001f|0002|55b3311d|00000003|00000002|
 48 69 20 74 68 65 72 65 2e 20 53 65 6e 64 69 6e
  H  i     t  h  e  r  e  .     S  e  n  d  i  n
 67 20 74 6f 20 33 20 66 72 6f 6d 20 32 21 21
  g     t  o     3     f  r  o  m     2  !  !
```

Please note the following regarding TextSend message:

- The destination id should be the id of the client that should receive the message (despite it goes through the server it should not be 0)
- The payload is encoded in ASCII

Please note the following regarding TextSendResponse message:

- The origin id should be the 0 in this case (indicating it comes from the server)

Other text send examples:

```
Send Text send from id 14 to id 12
MSG (length  55) 0000000e -> 0000000c sock id(14)| payLength 33 | cmd 2(cmd_TextSend) | msgid f5c9cd
13 | Payload
```

```
        @text Hi there. Sending to 12 from 14!!
|   Tag  | length | c  |  msgId |  dest  |  orig  | payload ->
|11223344|00000021|0002|f5c9cd13|0000000c|0000000e|
 48 69 20 74 68 65 72 65 2e 20 53 65 6e 64 69 6e
  H  i     t  h  e  r  e  .     S  e  n  d  i  n
 67 20 74 6f 20 31 32 20 66 72 6f 6d 20 31 34 21
  g     t  o     1  2     f  r  o  m     1  4  !
 21
  !
14:46:07 INFO CHAT TextSend Received STATUS_OK
MSG (length  55) 0000000e -> 0000000c sock id(12)| payLength 33 | cmd 2(cmd_TextSend) | msgid f5c9cd
13 | Payload
        @text Hi there. Sending to 12 from 14!!
|   Tag  | length | c  |  msgId |  dest  |  orig  | payload ->
|11223344|00000021|0002|f5c9cd13|0000000c|0000000e|
 48 69 20 74 68 65 72 65 2e 20 53 65 6e 64 69 6e
  H  i     t  h  e  r  e  .     S  e  n  d  i  n
 67 20 74 6f 20 31 32 20 66 72 6f 6d 20 31 34 21
  g     t  o     1  2     f  r  o  m     1  4  !
 21
  !
14:46:07 INFO CHAT TextSendResponse Received with status  STATUS_OK
MSG (length  29) 00000000 -> 0000000e sock id(14)| payLength 7 | cmd 3(cmd_TextSendResponse) | msgid
 f5c9cd13 | Payload
        @nopayload
|   Tag  | length | c  |  msgId |  dest  |  orig  | payload ->
|11223344|00000007|0003|f5c9cd13|0000000e|00000000|
 73 75 63 63 65 73 73
  s  u  c  c  e  s  s
```