Appendix A: Protocol specification basic server

The protocol of the black box level 1 server is described below. The protocol describes the following scenarios:

- Setting up a connection between client and server.
- Broadcasting a message to all connected clients.
- Periodically sending heartbeat to connected clients.
- Disconnection from the server.
- Handling invalid messages.

In the description below, *C* represents a message from the client and *S* a message from the server.

1. Establish connection

The client first sets up a socket connection to which the server responds with a welcome message. The client supplies a username on which the server responds with an OK if the username is accepted or a FAILxx in case of an error.

Note: A username may only consist of characters, numbers, and underscores ('_') and has a length between 3 to 14 characters. Implement the validation of this format in your server.

Happy flow:

Client sets up the connection with server.

S: INIT <welcome message>

C: IDENT <username>

S: OK IDENT <username>

To other clients (Only applicable when working on Level 2):

S: JOINED <username>

Error messages:

S: FAIL01 User already logged in

S: FAIL02 Username has an invalid format or length

S: FAIL04 User cannot login twice

2. Broadcast message

Sends a message from a client to all other clients. The sending client does not receive the message himself but gets a confirmation that the message has been sent.

Happy flow:

C: BCST <message>

S: OK BCST <message>

Other clients receive the message as follows:

S: BCST <username> <message>

Error messages:

S: FAIL03 Please log in first

3. Heartbeat message

Sends a ping message to the client to check whether the client is still active. The receiving client should respond with a pong message to confirm it is still active. If after 3 seconds no pong message has been received by the server, the connection to the client is close (before closing the client is notified with a DSCN message). The server periodically sends a ping message.

Happy flow:

S: PING

C: PONG

Error messages:

S: DSCN Pong timeout

Server disconnects the client.

S: FAIL05 Pong without ping

S: FAIL12 Pong was not received!

4. Terminate connection

When the connection is terminated, the client sends a quit message. This will be answered (with an OK message) after which the server will close the socket connection.

Happy flow:

C: QUIT

S: OK Goodbye

Error messages:

None

5. Invalid message

If the client sends a message other than the above, the server replies with an error "Unknown command". The client remains connected.

Example flow:

C: MSG This is an invalid message

S: FAIL00 Unkown command

6. Private message

One client sends a message to another client. The receiving client receives the message containing the name of the sender and the message itself.

Example flow:

C: PRIVATEMESS <username> <message>

S: OK PRIVATEMESS <username> <message>

Other clients receive the message as follows:

S: PRIVATEMESS <sender_username> <message>

Error Messages:

S: FAIL06 Unknown username

S: FAIL07 Cannot send empty message

7. See list

Example flow:

C: LIST

S: OK LIST username_1 username_2

8. Create Survey

C: SURVEYREQUEST

S: OK SURVEYREQUEST <username_2> <username_3>

C: SURVEY username1-username2-username3//Question1:Answer1-Answer2-Answer3//Question2:Answe4r-Answer5-Answer6

S: OK SURVEY username1-username2-username3//Question1:Answer1-Answer2-Answer3//Question2:Answe4r-Answer5-Answer6

Other clients receive the message as follows:

S: SURVEY username1-username2-username3//Question1:Answer1-Answer2-Answer3//Question2:Answer4-Answer5-Answer6

Error Messages:

S: FAIL08 Cannot start survey while there is an ongoing survey already!

S: FAIL09 Cannot start survey while there are less than 3 users online!

9. Fill in Survey

C: SUBMITSURVEY //question1:answer//question2:answer//

S: OK SUBMITSURVEY //question1:answer//question2:answer//

S: SURVEYSTATISTICS …

Error messages:

S: FAIL11 You cannot submit a survey twice!

10. File Request

C1: FILETRANSFERREQ <file_name> <file_size> <receiver> <uuid>

S: OK FILETRANSFERREQ <file_name> <file_size> <receiver> <uuid>

The other client gets the message as follows:

S: FILETRANSFERREQ <file_name> <file_size> <sender> <uuid>

Error message:

S: FAIL06 unknown username

11. File Transfer

1st route:

C2: FILETRANSFERACCEPT <file_name> <sender> <uuid>

The other client sees the message as:

S: FILETRANSFERACCEPT <file_name> <receiver> <uuid>

1.1 route -> After file successfully downloaded by receiver:

C: FILESUCCESS <uuid> <file_name>

The other client sees the message as follows:

S: FILESUCCESS <file_name> <sender>

1.2 route -> After file checksum mismatch:

C: FILEFAIL <uuid> <file_name>

The other client sees the message as follows:
S: FAIL10 File rejected because of checksum mismatch. File name: <file_name>

2nd route:

C2: FILETRANSFERREJECT <file_name> <sender> <uuid>

The other client sees the message as follows:

S: FILETRANSFERREJECT <file_name> <receiver> <uuid>

Error message:

S: FAIL06 unknown username

S: FAIL10 File rejected because of checksum mismatch. File name: <file_name>

12. Send public key and create session key:

C1: PUBKEY <receiver> <public_key_string>

The other client sees the message as follows:

S: PUBKEY <sender> <public_key_string>

C2: SESSIONKEY <sender> <session_key_string>

The other client sees the message as follows:

S: SESSIONKEY <receiver> <session_key_string>

Error message:

S: FAIL06 unknown username

13. Send encrypted message to desired user:

C1: ENCRYPTED <receiver> <message>

S: OK ENCRYPTED <receiver> <message>

The other client sees the message as follows:

S: ENCRYPTED <sender> <message>

Error message:

S: FAIL06 unknown username