UNIVERSITY OF TARTU, INSTITUTE OF COMPUTER SCIENCE

Task 1: UDP Communication

Hadachi&Lind

Must Read:
The task 1 should be done individually and you have ***one week*** to complete it (must be submitted before the next seminar session). Please, Submit your solution for task using course page*: https://courses.cs.ut.ee/2017/ds/fall/Main/Seminars.*
The solution should include your source code created and read me file explaning how to run it.

1 IMPROVING THE UDP APPLICATION

1.1 TASK (WARM UP)

In the current implementation of the Message Board protocol, we are stamping the messages with timestamps of arrival at the server-side. However, we do not take into account the time needed for the message to be transferred. Ideally, we would like to stamp the messages with the timestamp when user did hit enter button (and not the timestamp of arrival at the server).

Take the UDP application part1 as base and implement the following additional routines, refer to Figure 1.1:

- User wants to know the current time (client requests the server to get the current timestamp)
- User wants to know how much time is needed for one request to be delivered to the server

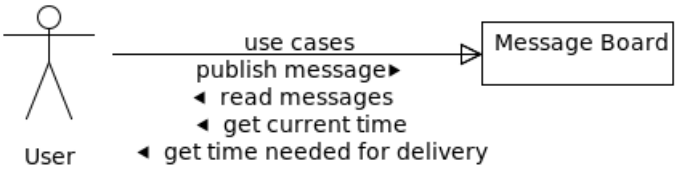


Figure 1.1: Use case diagram an improved MBoard

1.2 TASK (REAL-DEAL)

The current implementation of the sessions protocol allows only unidirectional sessions. This means, we can only issue session for transferring the data in one direction (unidirectional session). Moreover, the current session protocol is only allowing unidirectional sessions for uploading the data. For example, if we want to view all messages, we will need to start a session for downloading, and this is exactly what is missing in the current sessions protocol.

1.2.1 IMPROVING THE SESSIONS PROTOCOL

Based on the UDP application part2.

Modify the sessions protocol in such way that it would support downloading. The principal schema of the block-wise download in session could then be illustrated as shown in Figure 1.2.

As you can see, the getdata(...) routine issued by Client (c) is expecting, that there is a message prepared on the server-side (s) for Client (c). Therefore, an additional structure is required on the server-side; the OUTBOX for storing the messages intended to be delivered to clients (in case clients did not yet ask for the delivery). In fact, we already had one structure "messages" (M) for storing incoming messages that are delivered by the clients, we may now call it simply INBOX.

The whole sequence of actions of one server-side main loop cycle is as shown in Figure 1.5. The server first issues Sessions processing. Then, in Sessions module new packet UDP packet is received and processed, dependent on the header, the following actions may be taken by Sessions, assuming the UDP packet was received from client (c). This actions can as follows:

- Register new upload session for client (c)
- Put the received block into buffer of the active upload session for client (c)

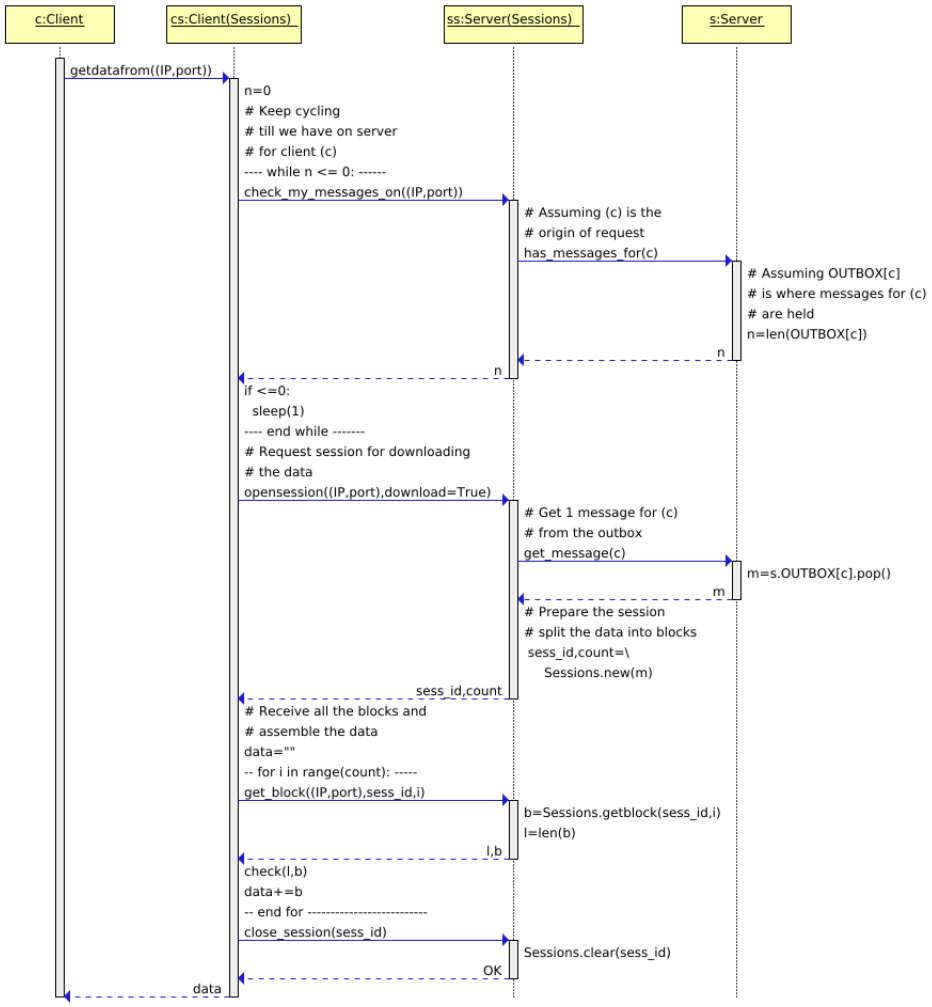


Figure 1.2: Sessions protocol, getdata() routine

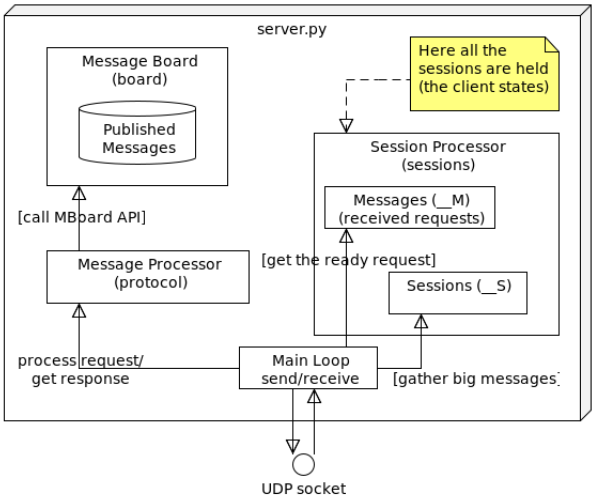
  - In case the last block received, assemble the whole message (M) and put into INBOX of the Message Processor module

- Terminate session for client (c)
- Register new download session for client (c)
  - Take one message for client (c) from the OUTBOX of the Message Processor module
- Prepare one block from the message to be delivered to client and sent it out

After processing Sessions, the server then processes the actual received requests (calling Message Processor module).

The MBoard client-side needs to be modified also. As we are now using Sessions for

both sending and receiving, the request/response interactions with the server have to be wrapped in senddata(...)/getdata(...) of the sessions protocol, refer to Figure 1.4 for more clarity.



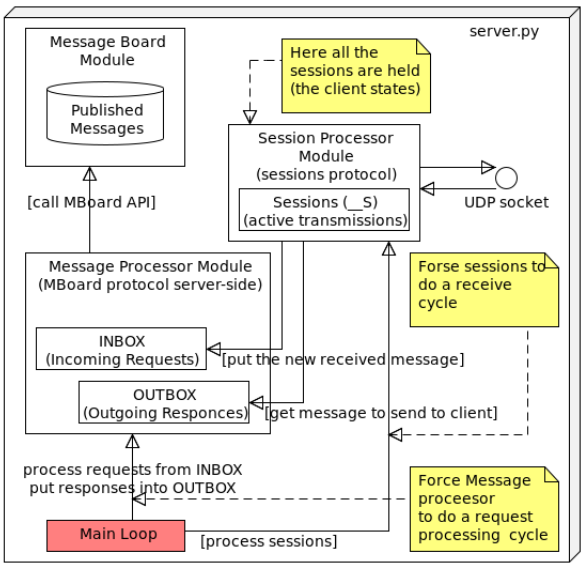


Figure 1.3: Server-side components interaction original (a) and improved (b)

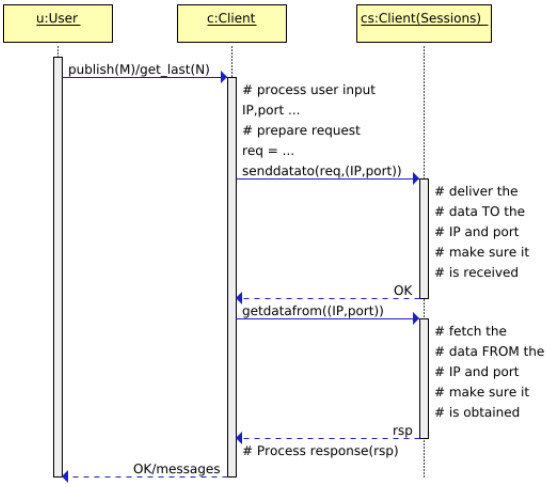


Figure 1.4: Client-side sequence request/response over sessions

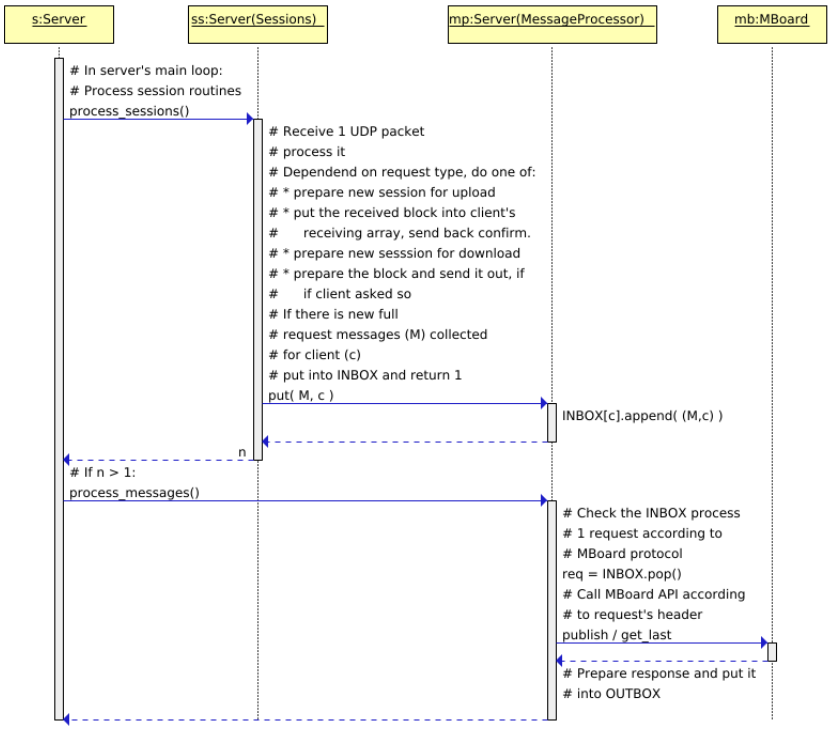


Figure 1.5: Server-side Session processing (UDP packets) and request/response messages using INBOX/OUTBOX