Computer Networks
Skill Check

Prof. Dr. rer. nat. Nils Aschenbruck

Leonhard Brüggemann, Bennet Janzen, Leon Richardt, Alexander Tessmer

General information:

- Read the **complete** task sheet **carefully** before you start working on it!
- By starting to work on the task, you declare that you are able to do so both physically and mentally.
- You must only use your assigned computer and work in the directory *Task* on the desktop. This is vital for submission (see below)!
- In case of an attempt to deceive, the assignment of all parties involved will be graded immediately as failed. There is no prior warning. Mobile phones that are switched on and other communication devices (such as smart watches) are also considered an attempt at deception! Changing the font size is also considered an attempt at deception.
- You are not allowed to eat in the digital exam labs, drinking is allowed.
- If you have to go to the bathroom, notify the supervisor. They will make sure that no two people go at the same time.
- Submission:
  - Make sure that your final version is saved in the directory *Task*.
  - Notify the supervisor to present your solution and **do not** turn off your computer.
  - The solution is automatically copied from your computer.
- System Information:
  - The tasks assume you are working on **Ubuntu 22.04**, using **Python 3.10**. We cannot guarantee support for other system environments.
  - Files that are part of this task are provided in the directory *Task* on the Desktop.
- Provided Documentation:
  - Linux man pages
  - Python documentation (might take a few seconds to open)
  - OpenSSL Cookbook
- Recommended IDEs: VS Code, vim, GNU nano, gedit

Please note: In order to pass, your solution must fulfill **all requirements** specified in the task. It is not sufficient to only complete a subset.

Task 1-D: `udp-uploader`

In this task, you will implement a client that reliably uploads a file to a provided UDP server. The server writes received data to disk in order and sends cumulative acknowledgments. It may randomly drop acknowledgments or send negative acknowledgments to simulate corruption. Your job is to implement a client in Python that copes with these impairments and completes the upload correctly.

1 Requirements

Implement the `udp-uploader` client in Python 3 in a file named `client.py`. Your implementation must:

1. Read the input file `data.data` from disk and split it into chunks of at most **1024 bytes** each. Do not change `data.data`.
2. Transmit each chunk as one D (Data) packet over UDP to the server. The packet format is specified in Section 3. The sequence number starts at **0** and increases by one per chunk.
3. For each outstanding packet, wait for a valid server response A (ACK) for up to **1 second**. Handle the following cases:
   a) You receive a packet A with sequence number *s* equal to the current packet's sequence number: consider the packet acknowledged and proceed to the next one.
   b) Timeout: retransmit the current packet.

   Use a reasonable **maximum retransmission limit** (e.g., 10) per packet to avoid infinite loops.
4. After all data chunks have been acknowledged, send a single-byte F (Finalize) packet and close the socket.
5. Print concise status information to the terminal, e.g., when sending, retransmitting, or receiving A.

You have passed the task when your client completes the upload and the server confirmes that the uploaded file matches the input file.

2 Server Description

A UDP server is provided to test your client. Start it with:

```
student@host:~/task-d$ ./server [port]
```

The server:

1. Listens on `127.0.0.1:`*`port`* (default `4711`).

2. Creates the output file `received.data`.
3. Starts a session upon receiving the first D from a client and will only accept packets from that same client for the session duration.
4. On receiving a D packet:
   - If the sequence number equals the expected one, writes the payload, advances the expected sequence number and acknowledges the packet.
   - Otherwise, does not write and acknowledges the last in-order sequence number.
5. On receiving a F packet, closes the file and resets the session state.

A single client session is supported at a time. The session starts with the first D received and ends on F.

The server may simulate packet loss on the data packets and the acknowledgements.

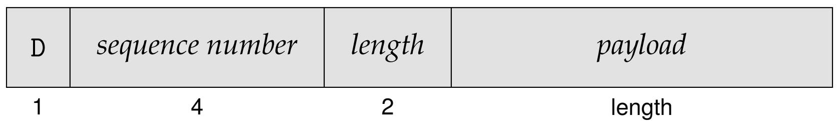
Example 1

In a correct implementation, this could be the output for the `udp-uploader`:

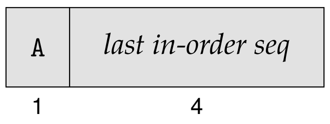
```
student@host:~/task-d$ python3 client.py data.data
[ACK] Seq=0 (attempt 1)
[ACK] Seq=1 (attempt 1)
[TIMEOUT] Seq=2, attempt 1/10
[TIMEOUT] Seq=2, attempt 2/10
[ACK] Seq=2 (attempt 3)
[ACK] Seq=3 (attempt 1)
...
[ACK] Seq=18 (attempt 1)
[ACK] Seq=19 (attempt 1)
Finalized: 20/20 packets sent in 4110.1 ms.
Sent FINALIZE, closing.
```

3 Protocol Format

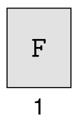
All packets are transported over UDP. Multi-byte integers are encoded in **big-endian**.



(a) D (Data) packet



(b) A (ACK) packet



(c) F (Finalize) packet

Figure 1: Message formats for the `udp-uploader` protocol.

4 Follow-Up

Try to answer the following questions:

1. What are the trade-offs between stop-and-wait and Go-Back-N/Selective Repeat in this setting?
2. How would you extend the protocol to support multiple simultaneous clients?

Note

For transparency: You will not have to answer these questions during the Skill Check. Nonetheless, we encourage you to think about how you would answer them. See them as an opportunity to improve your understanding of the relevant course material. Questions similar to these might be part of the final exam.

Good luck!