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-B: Temperature Sensor

In this task, you will implement software for a wireless sensor node. The node is equipped with a temperature sensor and placed inside a temperature-controlled shipping container. For monitoring purposes, the sensor node occasionally transmits its data to a central server which checks the temperature data to ensure the cold chain is kept intact. The temperature (in °C) and a Unix timestamp (at ms resolution) are written to the `temps.data` file in an ASCII format by another application on the sensor node.

1 Requirements

Since the power budget of the WSN is limited, and it is attached via a low-capacity link, we want to reduce the amount of data transmitted over the network. For our application, it is enough to get a reading of the temperature at a **5-second granularity**.[1] To further reduce the amount of bytes that must be transmitted, the ASCII data must be converted to a binary format. To summarize, your application must do the following:

1. Open a TCP connection to the server.
2. Transmit the data contained in the **first line** of `temps.data`.
3. Read the next line of the file.
   - 3.1. *If* the read line refers to a temperature value that has been recorded **less than 5 seconds** after the last transmitted reading: Ignore it.
   - 3.2. *Else*: Transmit the timestamp and temperature value of the line to the server.
4. *If* at the end of the file: Close the connection to the server. *Else*: Continue at Step 3.

Example 1 illustrates this procedure for the first 10 lines in `temps.data`.

Example 1

Timestamp	*Temperature*	
1729087341292	2.2286…	**Transmitted**
1729087342764	3.2781…	**Ignored**
1729087344381	2.4790…	**Ignored**
1729087345806	3.3499…	**Ignored**
1729087347432	3.4089…	**Transmitted**
1729087349371	2.9184…	**Ignored**
1729087351315	2.7977…	**Ignored**
1729087352767	2.6319…	**Transmitted**
1729087353881	2.7027…	**Ignored**
1729087355256	2.9401…	**Ignored**
⋮		

[1] *Note:* The timestamps in `temps.data` are recorded at millisecond resolution.

After the client closes the connection, the server verifies whether the transmitted records match the expectation. The server prints the verification result to the terminal.

You have passed the task when the data can be verified in a reproducible manner.

2 Server Description

To help you develop your application, we provide a mock implementation of the server. The server is started by running

```
student@host:~/task-b$ ./temperature-sensor-server [port]
```

with an optional `port` parameter. The server starts and binds to `127.0.0.1:`*`port`*, i.e., to the specified port on the local host. When the `port` parameter is omitted, the server binds to port `4711` per default. Once a client connects, the server reads from the connection until it is closed by the client. The server then verifies whether the transmitted data matches the expected data (cf. Section 1). Thereafter, the server prints whether the check was successful (cf. Example 2).

Example 2

Server output when the check is successful:

```
student@host:~/task-b$ ./temperature-sensor-server
Server listening on 127.0.0.1:4711
Handling connection from 127.0.0.1:48370: [SUCCESS] Samples are equal!
```

Server output when the check is unsuccessful:

```
student@host:~/task-b$ ./temperature-sensor-server
Server listening on 127.0.0.1:4711
Handling connection from 127.0.0.1:48372: [FAILURE] Samples are not equal!
```

To assist you in developing your implementation, the server logs messages to the terminal whenever a notable event occurs.

3 Protocol Format

In order for the server to understand what you are transmitting, you must stick to the expected format. Assume n is the total number of lines to be transmitted. Let $timestamp_i$ be the timestamp, and $temp_i$ the temperature of record i. The protocol format is described in Figure 1. Therein, $timestamp_i$ is an 8-byte (signed) integer, and $temp_i$ is a 4-byte floating-point number. When all n data points have been transmitted, the client must close the connection.

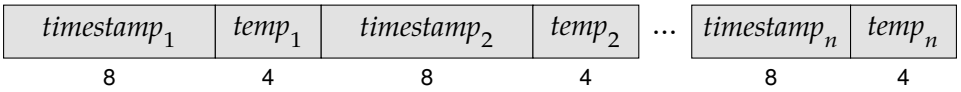

Figure 1: Format of data to be transmitted. The numbers below the fields indicate the field length in number of bytes.

4 Follow-Up

Prepare to answer the following questions:

1. Would you recommend the use of UDP over TCP for this application? Why or why not?
2. How could the amount of transmitted data be reduced further? Discuss any associated tradeoffs.

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!