Socket programming exercise

IDATA2304 Computer communication and network programming, Fall 2025

This document describes socket-programming exercises for the course. The exercises will contain the development of a single solution (a single application). However, the development will contain three parts. At the end of each part, you will have "a bit better product".

Part 1: Creating a Simple Smart TV Solution (Week 37-38)

In this part the point is to "get started" and create a solution according to the specification. You are allowed to be as sloppy as you want with the code structure. But you will need to fix it later 😊.

The task: create a Smart-TV solution according to the following specifications:

- There is a Smart TV, with the following functionality:
  - Smart TV will be a TCP server.
  - Turned off by default. When turned off, the only command the TV accepts is "Turn ON".
  - Can be turned on.
  - Can be turned off.
  - Can report whether it is on or off (for example, a method isOn() returns a boolean)
  - When turned on, the following functions become available:
    - Get the number of available channels C. This means, that you are allowed to switch channels to 1, 2, …, C.
    - Get the currently active channel number. It will be in the range [1, C]. Channel 1 is turned on by default when the TV starts the first time. After turning it off it remembers the previous channel (unless the server is restarted (electricity cut off)).
    - Set the channel to desired number c. This will fail if c is out of the allowed range [1, C].
- There could be several TVs, then each of them would have a different IP address or TCP port number.
- There are Smart remote controls, which will be TCP clients:
  - With buttons:
    - Turn on/off.
    - Switch one channel up.
    - Switch one channel down.
  - With a display – show the current channel of the TV it is connected to.
- Each remote can connect to one TV at a time.
- You need to model this as following applications:
  - One application for the TV, where you specify the TCP port number as a command-line parameter. If no port is specified, use 1238 by default.
  - The other application is a smart remote:

- Initially you can create it as a static application with no user input – just run through some predefined steps.
- If you wish, you can add a command-line interface or graphical user interface (GUI).
- The remote application takes two command-line arguments: the hostname (IP address or domain name) and TCP port number for the Smart TV to connect to

Your task

You need to **implement these two applications**: the Smart TV as a TCP server, and a remote control as a TCP client:

- **Design** the application-layer **protocol**.
- Write necessary **code for the server-side**.
- Write necessary **code for the client side**.

Note: You can use any programming language.

Figure 1 - Don't base your understanding of the task on this image.

Part 2: Refactoring and Testing (Week 37-38)

In this part your task is to improve the structure of the source code according to good practices. As a result of that, the code will become easier to test. You can write unit tests.

The Task Nr 1 – Refactor the Code

- Think about the structure of the code. What can you improve?
- Sketch a class diagram (on a piece of paper or digitally if you wish). Use this as a thinking-board.
- You can choose different solutions, as long as the following is true:

    - Extract the Smart TV logic into a separate class, so that it does not depend on communication in any way. The logic must have methods for the necessary TV functionality: turnOn(), turnOff(), getNumberOfChannels(), getChannel(), setChannel(int channel).
    - The logic-class must not depend on any other classes.
    - It should be easy to change the messages in the protocol. For example, if you use “on” as a command for turning on the TV, it should be easy to change it to “turn-on” or something else.
    - It should be easy to extend the system with new commands.
    - It should be easy to replace TCP with UDP.

Have you done this already when creating part 1? Good for you!

The task Nr 2 – Write Unit Tests (Optional)

If you have structured your code properly, you should be able to test the different parts separately.

Write the following tests:

- Some logic-tests, for example (choose at least 3 of these):
    - TV is OFF by default.
    - Can’t get channel list when the TV is OFF.
    - Can turn on the TV.
    - When turned on, getting the channel list works.
    - Setting the channel to a valid channel works.
    - Setting the channel to a negative number fails.
    - Setting the channel to a too-high number fails.
- Some message serialization tests, for example (adapt these to your protocol):
    - “on” is interpreted as an “turn on” command.
    - “list” is interpreted as a “get channel list” command.
    - “Chuck is da best” is interpreted as an invalid command.
- Some command interpretation tests, for example:
    - Get-channel-list returns the expected value (number of channels)
    - When a turn-on-command is executed on a TV, the TV changes it’s state from off to on
    - When a turn-off-command is executed, the TV changes the state to “off”
    - When a set-channel command is executed, the channel on the TV is switched

Part 3: Adding Multi-Client Support (Week 44)

Use multi-threading (and other techniques, if necessary) to extend the functionality of your solution:

- Allow multiple simultaneous client-connections on the server. I.e., one TV supports multiple remotes connected at the same time. If one remote “keeps the line busy” for a long time, this must not affect other connection of other clients.
- Whenever one remote control switches a channel, other remotes should get a notification about it right away. Implement this in an asynchronous way, without the clients polling the server all the time.

DELIVERABLE

- Zip file of the exercise or link to the repository where the source code is stored. The preferred way – use a public repository on GitHub. If you have some other solution, you can use that. Please make sure it must be accessible to the teacher.

Deadline

Friday on Week 44, October 31.