



# Homework 1, Sockets

## Network Programming, ID1212

### 1 Goal

- You can develop a distributed application, i.e. define tasks, assign tasks to processes and develop an application-specific communication protocol for process interaction using TCP or/and UDP sockets.
- You can develop an object-oriented node of a distributed application with a simple but yet informative and responsive (graphical) user interface, taking into account communication latency in a distributed application.
- You can use concurrent threads in nodes of a distributed application in order to improve scalability and performance (e.g. response time), and to hide communication latency.

### 2 Grading

The grading is as follows:

**Not accepted** Your work has not been accepted, and you have no score.

**0 points** Your work has been accepted.

**1 point** Your work has been accepted before or on due date.

**2 points** Your work has been accepted before or on due date. Also, it has an acceptable layered architecture and is well designed. This means it follows the guidelines of the lecture on architecture, and of the programming examples on the course web.

### 3 Auto-Generated Code and Copying

**You must be able to explain and motivate every single part of your code. You are *not* allowed to copy entire files or classes from the example programs on the course web, even if you understand it and/or change it.** However, you are allowed to write code which is very similar to the example programs on the course web. You are also allowed to use GUI builders and other tools that generate code.



## 4 Tasks

You are to solve *one* of the following two tasks. You do not get any extra points from solving both.

### Task 1, The Hangman Game

Develop a client-server distributed application in Java for the "Hangman" word guessing game (described in Wikipedia at [https://en.wikipedia.org/wiki/Hangman\\_\(game\)](https://en.wikipedia.org/wiki/Hangman_(game))). As a source of words, you can use the file `words.txt` on the course web.

#### Rules of The Game

Table 1 shows a sample execution of the game. In brief, the game is played as follows. The server chooses a word from a dictionary, and the client (the player) tries to guess the chosen word by suggesting letters occurring in the word (one letter at a time), or by suggesting the whole word. The client is only allowed as many failed attempts as there are letters in the word. A failed attempt is a suggestion of a letter not occurring in the word, or of the wrong word. If the client suggests a letter that occurs in the word, the server places the letter in all its positions in the word; otherwise the number of allowed failed attempts is decreased by one. At any time the client is allowed to guess the whole word. The client wins when the word is completed using single letters, or the whole word is guessed correctly. The client loses when the counter of allowed failed attempts reaches zero.

The client is able to play multiple such games, and the server records the total score. The score is initially zero, if the client wins the score is increased by one, if the client loses the score is decreased by one. This means the score can be negative.

#### Requirements on Your Program

All of the following requirements must be met in order for your solution to be accepted.

- Client and server must communicate by sending messages over a TCP connection, using blocking TCP sockets.
- The client must not store any data. All data entered by the user must be sent to the server for processing, and all data displayed to the user must be received from the server.
- The client must have a responsive user interface, which means it must be multithreaded. The user must be able to give commands, for example to quit the program, even if the client is waiting for a message from the server.
- The server must be able to handle multiple clients playing concurrently, which means it must be multithreaded.



- The user interface must be informative. The current state of the program must be clear to the user, and the user must understand what to do next.

User's action	User interface shows			Word chosen by server
	Word	Remain- ing failed attempts	Score	
Connect to server	no value	no value	0	no word chosen
Start game	_____	7	0	hangman
guess p	_____	6	0	hangman
guess a	_a__a_	6	0	hangman
guess m	_a__ma_	6	0	hangman
guess t	_a__ma_	5	0	hangman
guess gangman	_a__ma_	4	0	hangman
guess hangman	hangman	no value	1	no word chosen
Start game	----	3	1	dog
guess a	----	2	1	dog
guess e	----	1	1	dog
guess n	----	0	0	no word chosen

Table 1: A sample execution of the client-server hangman game.

### What is NOT Required of Your Program

Below is an explanation of things that do not affect your score.

- Minor changes or misunderstandings of the rules are allowed, as long as your program does not become notably simpler than a program implementing the correct rules.
- You are not required to create a graphical user interface. A command line UI is sufficient.

### Task 2, The Rock-Paper-Scissors Game

Develop a distributed peer-to-peer application for the multi-player rock-paper-scissors game. (Wikipedia page at <https://en.wikipedia.org/wiki/Rock-paper-scissors>).

#### Rules of The Game

Each player runs a node with a user interface that allows to play the game and show scores. Table 2 shows a sample execution of a game with three nodes. In a round of the



game, each player chooses one of rock, scissors, or paper. Then the nodes communicate their choices to each other in order to compare the choices to see who won. Rock wins over scissors, scissors wins over paper, and paper wins over rock. Points are awarded as follows. Assume  $n > 1$  players. Award  $m$  players  $(n - m)$  points each if they choose the same gesture and beat the other  $(n - m)$  players. This rule implies that if all  $n$  players choose the same gesture (i.e. if  $m = n$ ), points are not awarded. If a player beats all the others (i.e. if  $m = 1$ ), the winner is awarded  $(n - 1)$  points. When a round is over, the players may decide to play another round or to quit the game. The user interface must reflect the score of the last round and the total score.

### Requirements on Your Program

All of the following requirements must be met in order for your solution to be accepted.

- Nodes must communicate by sending messages over TCP or UDP, using blocking sockets. Note that you may face problems running UDP multicast over KTHOPEN or Eduroam. Therefore, you are advised not to use multicast.
- Use only one peer node for each player, do not use an additional coordinator node. Each node must run exactly the same program, there may not be any “master” node.
- Your program must allow at least three nodes to participate in the game.
- The node must have a responsive user interface, which means it must be multithreaded. The user must be able to give commands, for example to quit the program, even if the node is waiting for a message from another node.
- The user interface must be informative. The current state of the program must be clear to the user, and the user must understand what to do next.

### What is NOT Required of Your Program

Below is an explanation of things that do not affect your score.

- Minor changes or misunderstandings of the rules are allowed, as long as your program does not become notably simpler than a program implementing the correct rules.
- You are not required to create a graphical user interface. A command line UI is sufficient.

User A's action	User B's action	User C's action	User A's view	User B's view	User C's view
start game			round: 0 total: 0	none	none
		start game	round: 0 total: 0	none	round: 0 total: 0
	start game		round: 0 total: 0	round: 0 total: 0	round: 0 total: 0
	rock		round: 0 total: 0	round: 0 total: 0	round: 0 total: 0
paper			round: 0 total: 0	round: 0 total: 0	round: 0 total: 0
		paper	round: 1 total: 1	round: 0 total: 0	round: 1 total: 1
		paper	round: 0 total: 1	round: 0 total: 0	round: 0 total: 1
	scissors		round: 0 total: 1	round: 0 total: 0	round: 0 total: 1
paper			round: 0 total: 1	round: 2 total: 2	round: 0 total: 1
rock			round: 0 total: 1	round: 0 total: 2	round: 0 total: 1
		rock	round: 0 total: 1	round: 0 total: 2	round: 0 total: 1
	rock		round: 0 total: 1	round: 0 total: 2	round: 0 total: 1

Table 2: A sample execution of the peer-to-peer rock-paper-scissors game with three players.