

Assignment 1: UDP/TCP socket programming

UDP/TCP socket programming

The first assignment is dedicated to UDP/TCP socket programming with Java and testing your programs in a virtual networking environment. You will use provided starter code for UDP echo server and client, improve it and test your implementation in a setting where server and client programs are executed on different machines connected in a network.

Important reference material

- [Ubuntu terminal usage documentation](#)
- [Notes on VM usage](#)
- [Oracle VirtualBox documentation](#)
- [Java networking tutorial](#)
- [Official Java documentation](#)
- [Java concurrency tutorial](#)
- [Wireshark tutorial](#)

Required downloads

- [Virtual Box](#) (Don't forget Extension Pack)
- [Ubuntu virtual machine image](#), [alternative link](#) (may require registration)
- [Starter code for UDP echo server / client](#)
- [Wireshark download](#)

Problem 1

You will start with setting up the virtual networking environment. Follow the guidelines [on this page](#).

After completing the installation, check networking with “ping -c 5” program from one machine with IP address of the other machine. Include resulting **screenshot** in your report.

Problem 2

This part of assignment involves programming. Start with downloading provided UDP server and client programs. Execute Java classes / JAR files of server program on your virtual machine and execute client program from the host machine.

The next feature to implement is configuration of client buffer size (in bytes) and messages transfer rate (in messages per second). Transfer rate value 0 must result in client sending message only once. It must be possible to specify these values as client command-line parameters in the same way as IP and port address. Try sending 5 messages / second (a 1-second long simulation is enough), include resulting **screenshot** in your report. For now you can leave the buffer size at a default value.

A little clarification on client buffer size: the value refers to size of byte array used as buffer, not to Socket class methods that manipulate underlying OS mechanisms. For example, in Problem 3 with TCP you are supposed to use `InputStream.read(byte[])` method instead of `InputStream.read()` that would fetch one byte at a time, and the size of array used as buffer for this task is supposed to be configured.

Improve programs by adding exception handling and error messages output. Your task is to handle **all** possible errors. *Hint:* for example, what is considered a valid IP? What message size can lead to a program failure? **List the exceptions/errors** you handle in the report.

VG-task 1: Let's assume messages transfer rate $\geq 10/\text{sec}$. As the client-server operations, such as sending and receiving messages, take some amount of time, it will actually take a bit more than a second to send such amount of messages. Your task is to make this mechanism work properly, i.e. to send the specified amount of messages in exactly one second. If transfer rate is too big (e.g. $10\,000/\text{sec}$), you must send as much messages as possible during a second and notify user about the amount of remaining messages.

VG-task 2: Implement the networking layer for UDP as an abstract class, reuse methods of this class in TCP (problem 3) accordingly.

Problem 3

Continue with your implementation of UDP echo server / client. Now the task is to implement the same echo server / client functionality (including client buffer size, message transfer rate and error handling), but using TCP instead of UDP. You can reuse parts of code from task 1.

However, the server must support multiple client connections. To achieve this, you have to modify server code to use Java threads: upon accepting connection from client, new thread with request handling code must be created and executed. After sending response (echo), the thread execution should stop. The main server thread, as previously, is supposed to run in a loop until manual termination. Include resulting **screenshot** with multiple clients in your report.

Try to set client buffer size to some small value (e.g., 64) and run programs with message of bigger size (e.g., 100). Then repeat this procedure with your UDP code from problem 2. What is the difference and why? Include resulting TCP/UDP **screenshots** (or paste the terminal session as text) and answer this question in your report.

Problem 4


This part of assignment involves using Wireshark software. Install this tool and run it (you will need Administrator / root privileges). The good idea is to read one of dedicated tutorials, see "reference material". Plenty of tutorials are also available on the Internet.

Choose the virtual networking interface on your host machine (check VirtualBox settings to find out its title). Start capturing the traffic and repeat an experiment with a small buffer size (Problem 3) for both UDP and TCP. Include resulting **screenshot of Wireshark** in your report and explain the results. *Hint:* try reading about "handshake". *Hint #2:* explaining results means explaining everything you see in Wireshark during network communication, i.e. what is ACK, SYN, PSH? Where is the difference between TCP and UDP?

Submission

Read the general [submission instructions](#).

Submission status

Submission status	No attempt
Grading status	Not graded
Due date	Tuesday, 14 February 2017, 11:55 PM
Time remaining	21 days 10 hours
Last modified	-
Submission comments	 Comments (0)

Add submission

.