Network Technology and Programming Lab
Assignment 2

Stefanos Papadakis
Deadline: 3 April 23:59

March 19, 2024

General Information

The goal of this assignment is to become familiar with socket programming and the challenges of measuring the end-to-end performance of a network. The implementation language is up to your choice, but we strongly recommend to choose a low-level one (preferably C/C++ and POSIX sockets) that will allow you to get a deep understanding of network programming.

1. Introduction

In this assignment you have to implement a network measuring tool. The tool will measure the end-to-end throughput of the network, jitter and an estimation of the one-way latency.

The tool works in a client-server model. The server waits for connections and the clients connect to the remote server to begin the experiment. There are two communication channels between each client and the server.

- The signaling one, which is used for information exchange and signaling between the server and the client. This communication channel uses **TCP**
- The experiment channel. This channel transfers data over **UDP** that are part of the network measurement process

Your are free to create additional channels depending the runtime parameters of your program.

2. Parameters

Your program should be able to receive the following command line parameters. Note that you have to support exactly the syntax below.

Server/Client parameters:

- **-a**: In server mode, this argument specifies the IP address of the network interface that the program should bind. If none given the program binds in all available interfaces. In client mode this argument provides the IP address of the server to connect to
- **-p**: In server mode this argument indicates the listening port of the primary communication TCP channel of the server. In client mode, the argument specifies the server port to connect to
- **-i**: The interval in seconds to print information for the progress of the experiment. The information displayed maybe different depending on the experiment setup. You are free to produce any kind of meaningful result.

- **-f**: Specifies the file that the results will be stored. This option is for your own convenience. Use an output format that will help for plotting the results

  Server parameters

- **-s**: The program acts like server

  Client parameters

- **-c**: The program acts like client
- **-l**: UDP packet size in bytes
- **-b**: The bandwidth in bits per second of the data stream that the client should sent to the server. The program should take into consideration the overhead of the lower TCP/IP stack layers in order the measurement to be as accurate as possible
- **-n**: Number of parallel data streams that the client should create. For each one of these streams a new thread is assigned in both server and client
- **-t**: Experiment duration in seconds The client should send for the specified amount of time and the stop the data stream. Before it exits, it informs properly the server in order the later to print the measurement results. If this argument is not specified, the data stream should be continuously sending data until a user termination signal occurs
- **-d**: Measure the one way delay, instead of throughput, jitter and packet loss
- **-w**: Wait duration in seconds before starting the data transmission

3. Network Measurements

At each run the tool should perform either the one way delay or the throughput, goodput, jitter and packet loss measurement, depending on the argument specified. The measurement process is performed **at the server** and stops when the client sends appropriate message through the communication TCP channel. Then the results are printed to the server and are sent back to the client for printing to the terminal.

- **Average throughput:** The server measures the data that receives from the client until a termination signal from the control channel is received. When this happens it computes the effective throughput. As the tool can measure only the payload on the UDP packets, proper adjustments should be made to take into account the headers overhead of the different layers. Try these adjustments to be as accurate as possible.
- **Average goodput:** Same as throughput, but measuring only the bytes received from the UDP packets.

- **Packet loss percentage:** The tool should be able to report the percentage of lost packets. To achieve this each packet should contain a header with an appropriate counter. The server uses this counter to extract the packet loss. Special attention should be given in edge cases such as the change for packets to be lost at the end of the transmission. To overcome this problem, the client can send among with the termination signal and the counter of the last packet sent.

- **Average jitter:** Jitter is the difference between the inter-arrival times of consecutive packets.

- **Standard deviation of jitter:** Except the average, the tool should be able to specify the standard deviation of the jitter during the experiment.

- **One way delay:** In this type of measurement the tool exchange data in a different way. The most common technique to measure the one way delay is the RTT/2. We discussed in the class why this measurement is not so accurate. Extra bonus points will be given to teams that will try to estimate more accurately the one way delay using their tool.

4. Implementation considerations

Measuring accurately the time is quite critical in such type of applications. The timing function should provide enough precision and should be not subjected to timing adjustments from external sources (NTP, PTP, etc). Prefer monotonic reference clocks (e.g **clock_gettime(CLOCK_MONOTONIC, ...)**)

Remember that the goal of this tool, is to extract accurate results about the network. Filling the data buffers with random data, or allocate and de-allocate continuously memory can severely affect the precision of your measurements. Fill only the necessary minimal information and use pre-allocated buffers.

Another common issue that can destroy your measurements is the blocking nature of some network functions. You should identify these corner cases and deal with them. For example, you should not start measuring time before a blocking call, that you do not known for how much time will block.

4.1 Roadmap

Below there is a roadmap for your implementation. Try to follow it as much as possible, as it will help you to incrementally build the application.

1. Create a single executable file containing the **main()** of your program and the argument parsing

2. Create two different files one for the server and one for the client

3. Start the implementation of the TCP communication channel

(a) As you may recall, TCP is a stream oriented transport protocol and there is no notion of a packet from the application point of view. Therefore, you should implement a proper communication protocol between the two endpoints

(b) Start by defining a header of **constant** size. The constant sized header is crucial for the receiver FSM. The rest of the application layer packet can be variable

(c) Populate the necessary fields into this header. For example, message type, message length etc.

(d) Make sure that the two endpoints can communicate properly and exchange all the necessary information without issues

4. Start with the implementation of the UDP data transfer. At this point, you should not yet integrate it into the rest of your application, just to keep it simple. Test the data transfer as stand alone routines and without any throttling. Assume that port number and IP of the remote host are known

5. Implement the throughput/goodput measurement mechanism

6. Without any throttling make sure that the results of your measurement are in par with those reported by tools like **iftop**

7. Start implementing the throttling mechanism. Again use **iftop** to check if the reported and the actual throughput matches the intended one

8. Implement the rest of the measurements required

9. Start the integration of the UDP experiments into the main application. Parameters, start/stop and anything else should now be instructed by the TCP communication channel. The server and the client should exchange vital information for establishing the UDP transfer

10. Support parallel streams, by properly assigning each stream to a separate thread

11. Conduct the experiments, plot the results and document them into your report

5. Experiments

Make some experiments similar to the experiments that you have done in the assignment 1 in order to demonstrate the full functionality of your tool. You can use the lab hosts or your own. If you use the lab, do not forget to allocate a timeslot!

Make a report showing the plots of your experiments and if you spot any abnormal behavior try to provide a reasonable explanation.

Oral Examination

All the students who have submitted their exercises are requested to attend the oral exam session, in order to present their solutions. A short quiz will also take place during that time. You will need to choose a timeslot for the oral exam using Doodle. More details will be sent to you via email.

Attention

- Each team will only be examined during the timeslot choosed.
- During this session both the Assignments 1 and 2 will be examined.
- Both the timely submission and the oral exam session will contribute to the grading of the assignments.