Εργαστήριο Δικτύων ΙΙ – Άνοιξη 2025

2ο Project – SpeedTest with Unix Sockets

1. Project Overview

In this project you will implement a SpeedTest application using TCP Unix Sockets in C. SpeedTest applications typically measure throughput and latency at both Downlink and Uplink directions. Your SpeedTest App will measure only Downlink TCP throughput, using a client / server architecture, running over Unix Sockets.

a. SpeedTest Implementation Overview

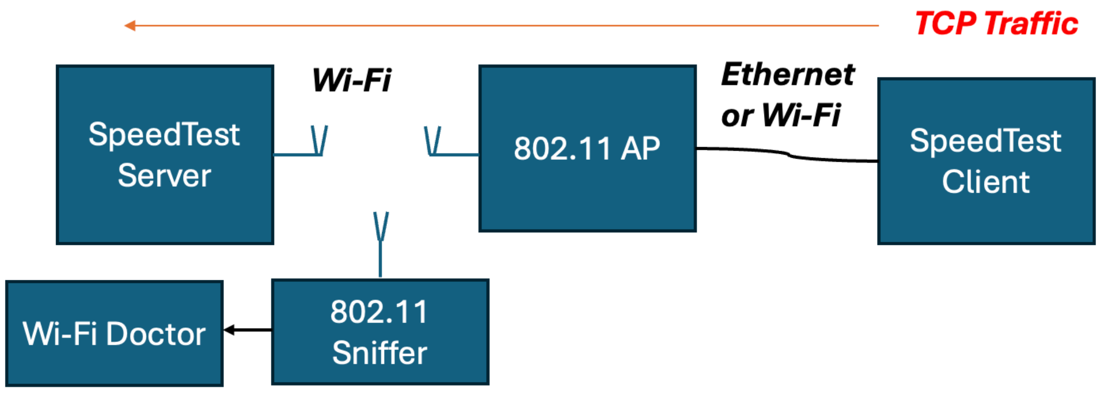
You will implement a SpeedTest application using a client / server architecture and TCP Unix Sockets in C, as discussed in the class. The server (server.c) will always run at a configured TCP and port listening for clients' connections. The client (client.c) will connect to server's IP/port, and create full buffer traffic for 30 secs. It will print the data sent rate (Mbps) in 2-second time windows and exit. The server will print the received throughput (Mbps) in 2-second windows and the aggregated throughput (Mbps) over the 30-second experiment.

Implementation notes:

- Your server will serve one client at a time. There is no need to use multi-threaded code or "*select*".
- The server should never exit. It should always run to accept new client connections.
- Your server (as discussed in the next section) should not run at localhost (127.0.0.1).
- Each SpeedTest experiment will run for 30 seconds. The server will print both the aggregated throughput (Mbps) and the runtime throughput (Mbps) every 2 seconds.
- Use prints at every step to illustrate the progress of your SpeedTest App at runtime.

b. SpeedTest App Evaluation

You will test your SpeedTest Application using the following topology.



The TCP traffic will be Downlink from the client to the server. The server will be connected to the network through an 802.11 Access Point (AP). The SpeedTest client will connect to the AP through Ethernet or Wi-Fi networks (or both). To limit the wireless impact to the last mile (AP to server), it is preferable (not required though) the client - AP connection to involve only wired links. The throughput will be measured at the server side. A Wi-Fi sniffer operating at the same channel as the AP, will monitor the wireless performance.

<u>Evaluation settings</u>: You will evaluate the above topology in 5 scenarios:

1. SpeedTest server device close to the AP (high RSSI) at 2.4 GHz.
2. SpeedTest server device far from the AP (low RSSI) at 2.4 GHz.
3. SpeedTest server device close to the AP (high RSSI) at 5 GHz.
4. SpeedTest server device far from the AP (low RSSI) at 5 GHz.
5. Mobility scenario where the SpeedTest server device is moving in pedestrian speed far from the AP.

For each scenario you need to calculate:

- An overall throughput over the 30-second time window.

- Timeseries analysis of the throughput. Specifically you will break the time in 2-second time windows, and you will plot the throughput vs time.

For each scenario, please provide a plot with throughput timeseries and a throughput distribution (max/min/mean/median/75perc/95perc) in your report.

Throughput verification: You will validate the accuracy of your App's throughput estimation, by comparing its performance with ***iperf*** tool. iperf is a free open source tool used to measure network performance. It relies on a client-server architecture similar to your SpeedTest. For more information see [1]. For your test, configure your iperf to run in TCP mode for 30 secs, similar to your SpeedTest App and monitor the throughput every 2 seconds. Your SpeedTest and iperf tests need to be conducted back-to-back to ensure similar network conditions. Compare both the aggregate and the timeseries throughput. Illustrate through a single plot (x-axis: time, y-axis: throughput in Mbps) the comparison between your SpeedTest and iperf.

Wi-Fi Doctor Analysis: You will use the Wi-Fi Doctor from project 1 to analyze the performance of the wireless link. Your analysis should include the following metrics:

- Throughput as estimated by the Wi-Fi Doctor.
- Data Rate.
- Frame Loss.
- RSSI.
- Rate Gap.

Monitor the SpeedTest App throughput and the above metrics in 2-second time windows. Create the plots where on x-axis there is time and on y-axis there is the throughput and the metric(s). In your report you need to compare the throughputs in all the above 5 scenarios and explain the throughput gaps you observed using the Wi-Fi Doctor metrics. For example, throughput of scenario A is greater than scenario B because of congestion.

<u>Revise Wi-Fi Doctor Throughput Estimation</u>: Wi-Fi doctor estimates the theoretical wireless throughput as

- *Throughput = Data_Rate * (1-Frame_Loss_Rate)*

How can you improve your theoretical throughput formula, to be closer to the SpeedTest App measured throughput? Revise the above formula and compare it (use a timeseries plot) with the SpeedTest App throughput. Check the literature for related papers on theoretical throughput estimation.

<u>Note</u>: You can consider channel utilization in the above formula. I am more interested in the discussion and your thinking process, rather than identify a more accurate formula.

Deliverables

The project deliverable is a zipped folder with:

- Your code along with a README file to explain what each file of the code is doing.
- A report describing your SpeedTest system and your results.

The report will be a .doc or .pdf, in single column format, up to 10 pages. It can be either in Greek or in English. Its structure will be similar to an academic research paper, with the following sections.

- **Title**
- **Introduction**: Includes the goal of the project, a short description of your system and a short description of your results and findings.
- **Design**: It should provide an overview of your system, including important implementation details.
- **Evaluation**: Presents the evaluation results as discussed in Section 1(b).
- **References**: List of references (if any).

Longer reports are not necessarily better. Be precise!

2. Grading

The exam of the project will have 60% weight, while the report will have 40%. The questions in the exam require a clean report to be answered properly. You will be graded based on your code completeness and correctness, the completeness of your evaluation, the quality of your analysis and visualizations (tables, plots).

3. References

[1] https://iperf.fr/