Performance Evaluation

EITN30: Internet Inside

Moa Eberhardt - mo5385eb-s
Abdalsalam Muhialdeen - ab5015mu-s

Date: 2024-05-17

LUNDS UNIVERSITET
Lunds Tekniska Högskola

1 Implementation

To test the system, we implemented two applications based on the server-client diagram to measure the throughput and latency. The general idea is to have the server listing for requests sent by the client and answer with helpful information about the packet traversal over the system. This would mean that when the client sends a stream of packages at a certain speed our servers could respond with the practical throughput at which the packages arrive or the round-trip-time representing the latency of the connection.

When implementing the tests a python wrapper class of the iperf3 client and server were used to generate the throughput result data. Whilst the socket python package was used to create a UDP connection in which the round-trip-time was measured.

To run the tests, start the corresponding python server for that test on the base station:

```
'tests/iperf3_server.py' -- throughput
'tests/udp_server.py'    -- latency
```

and then run the bash script with the right option on the client:

```
'./run.sh tTest' -- throughput
'./run.sh lTest' -- latency
```

2 Test Results

To find the maximum capacity of our system, we stress test it by running the above scenario in a loop by sweeping over a range of speeds. More generally a subjective lowest speed of 2 Kbps and a speed above our maximum speed. The results are shown in Figure 1 where it's clear that we achieved a max speed of 400 Kbps and an average latency of 30 ms for a none-overloaded system ($\rho < 1$).

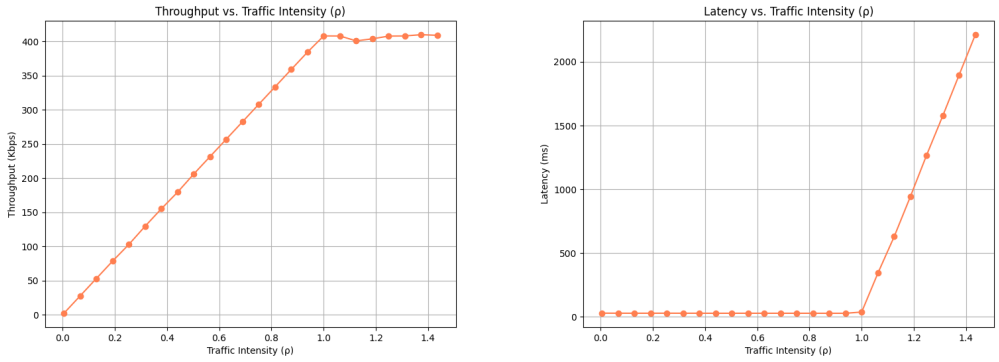

Figure 1: Resulting plots of the Throughput and the Latency tests

3 Discussion of the result

The results shown in figure 1 are accurate considering the boundary of overload is $\rho \geq 1$. In the throughput graph this showcases the turning point and shows what the maximum throughput the system can handle. Similarly, latency starts to increase once ρ have reached the value of 1.

In earlier states of the implementation, latency started increasing before $\rho > 1$. By calibrating the delay time, to become only 250 μs instead of $10 \cdot 250$ μs, and use 2 retries instead of 15 the turning point changed to $\rho = 1$ resulting in a less increasing latency for traffic intensities near 1.

Overall the tests and their results can be seen as reliable given that we use well-defined python packages to perform the test for each individual speed. Our implementation only iterates over the range of speeds we are interested in and performs the tests so that we can plot the data.