**ECE 545 – Modern Internet Technologies**

**Project Report**

**In-Depth Investigation of TCP via NS3 Simulation**

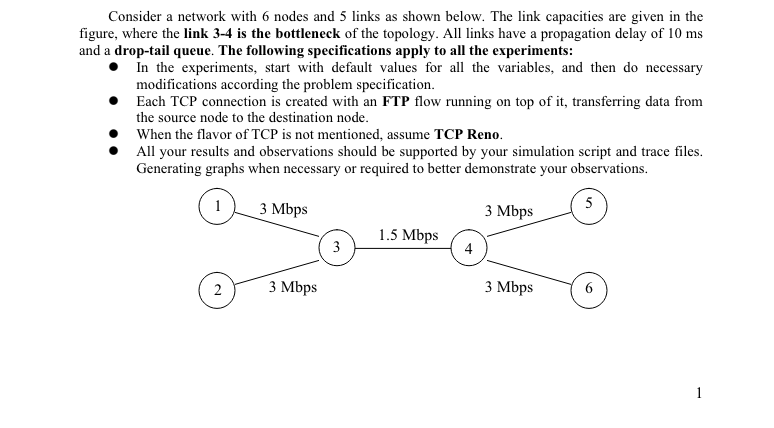
**Course Instructor : Prof. Yu Cheng**

**Submitted by**

**Samman Chouhan**

**A20561414**

Description: NS3 is a widely used discrete event simulator for network analysis. In this project, you will study the implementation of the TCP protocol using ns3 simulation. You will study the various flavors of TCP and how the behavior of TCP changes according to the background traffic. You may need to find and read related RFCs or other references to learn the implementation details of the TCP protocols (i.e.,Reno, Westwood) considered in this project.

****

Part 1: Efficiency of the TCP congestion control, flow control, and reliable data transfer protocols In the first part, create a single TCP connection between node-1 (source) and node-5 (destination), and no other background traffic is considered. Do experiments to observe the operations of TCP. 1. Use the default parameters, and run your simulation from 0 to 10 seconds. Analyze your simulation results and answer the following questions. What is the total number of segments and the number of bytes successfully transmitted during the 10 seconds? What is the average throughput achieved? How does this compare with the bottleneck bandwidth in your topology? Then, change the queue size used at node 3 to a value much smaller than the default value in one case, and to a value much larger than the default value in the other case. Run simulations to obtain the average throughput in these two cases, respectively. Do you get different average throughputs compared to that under the default queue size? Explain your observations?

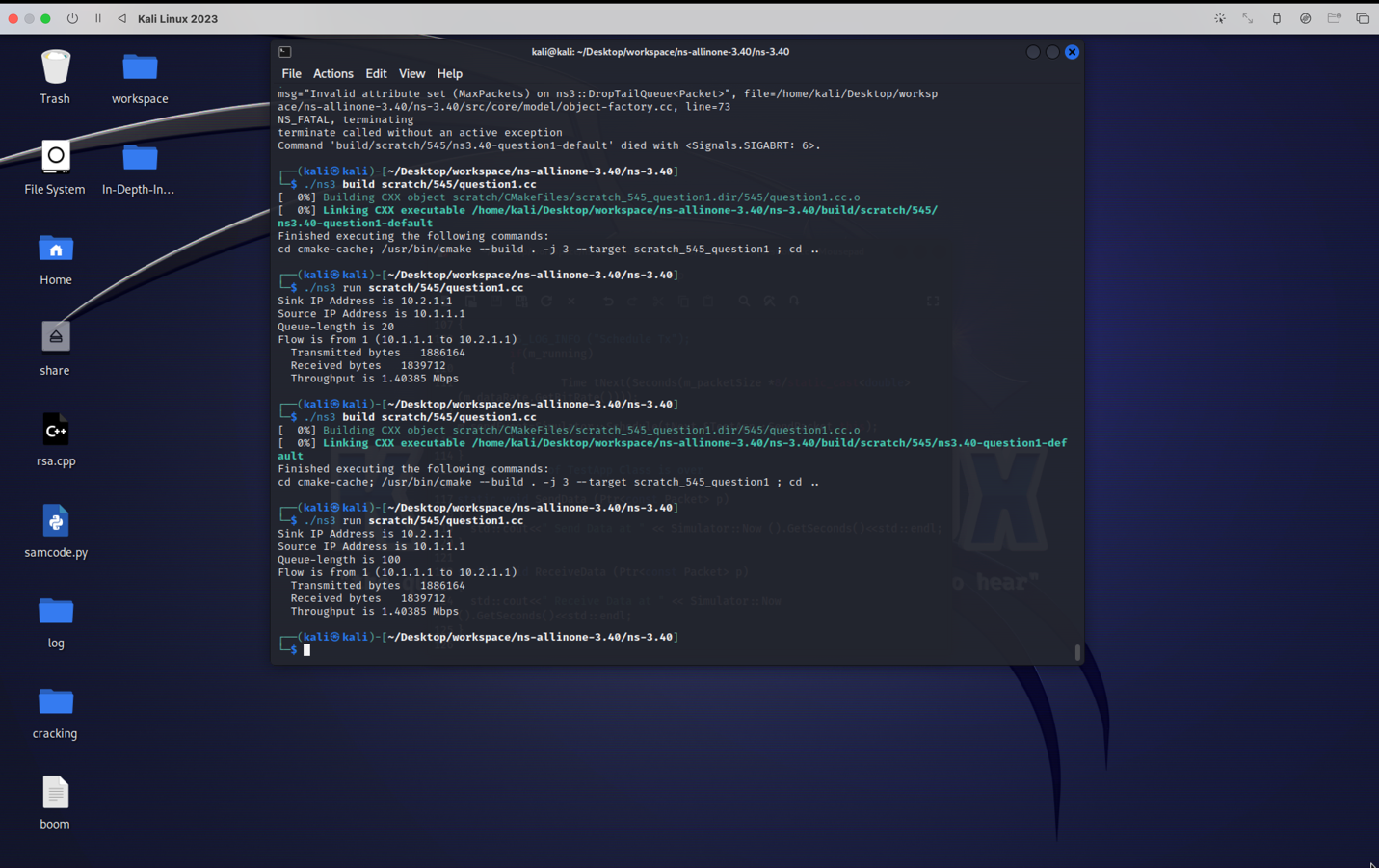
**A screenshot of a computer

Description automatically generated**

The Throughput as shows in the output is 1.40385 MBPS, the number of bytes successfully transmitted during 10 secs is 1886164 bytes

The average throughput is 1.4 mbps and the main reason for that is the bottleneck between the nodes 3 and 4 with a link speed of 1.5 mbps

Hence the throughput can never be more than 1.5 in this case

****

And surprisingly even with the queue set with 100 packets , the throughput is still the same

2nd

Plot the congestion window (cwnd) as a function of time. Use the graphs generated from one or multiple simulation runs to demonstrate the options of slow start, congestion avoidance, the reaction to triple duplicate ACK, and the reaction to timeout.

A graph with a line graph

Description automatically generated

3RD RTT and Estimated RTT vs time

**A screen shot of a graph

Description automatically generated**

**4th**

**Average Throughput when packet loss rate at 10^-2**

**A screenshot of a computer program

Description automatically generated**

**Similarly we can find all the other throughput**

**Part 2**

TCP + UDP: Flow-1 is TCP and flow-2 is UDP attached with a CBR source. Run the experiment multiple times by modifying the traffic generation rate of the UDP flow, and observe the throughput achieved by the two flows. Plot a graph showing the UDP throughput on the x-axis and TCP throughput on the y-axis. Determine the UDP throughput for which the two flows achieve a fair share of the link. Fix the UDP rate at the fair-share rate you just found, and run simulations to calculate the loss rate of the TCP connection. Replace the drop-tail queue with another queuing scheme provided by ns3 and run simulation again. Do you get different throughput under a new queuing scheme? If yes, do some extra reading to find possible reasons leading to the TCP throughput increase or decrease under a certain queuing scheme.

A screen shot of a computer

Description automatically generated

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer screen

Description automatically generated**

To calculate loss rate tcp

Transmitted bytes 630444. Recived bytes 337620 = 46 % , as it can be seen in the graph

TCP + TCP: Both flows are TCP, with a maximum window size (MWS) of 30 packets for flow-1 and 6 packets for flow-2, respectively. Plot, in one graph, the throughput of both flows vs. time. Do the flows get a fair share of the link? Explain.

A screenshot of a computer screen

Description automatically generated

Since its clear that flow 1 takes the most bandwidth , so the throughput of flow-1 is more than that of flow-2

2.3

To achieve this optimization and traffic management, TCP uses a separate machine, the Combocorder, state machines for congestion and flow control. One of the primary goals is to prevent congestion and make the bandwidth of all flows fair. For example, regarding our flows:

Congestion window growth. Flows 1 and 2 start with the same initial cwnd size. Indeed, according to various TCP congestion control models (e.g., TCP Reno and TCP Cubic), the cwnd grows by one MSS every RTT before a loss. Since the MSS of Flow 1 is larger so that a whole packet would be 1000 bytes, this window will grow in bytes faster compared to the window with the MSS equal to 500 bytes.

Bandwidth sharing. As long as the two flows doubled their window sizes by how many bytes they can send, that is, a whole packet, Flow 1 would try to claim more bandwidth in each RTT cycle than Flow. Therefore, without any additional mechanisms, changes or adjustments, mechanisms or any additional phenomena, Flow 1 will capture more than half of the traffic as at the beginning.

AIMD (Additive Increase/Multiplicative Decrease): TCP’s AIMD algorithm is designed to provide long-term fairness, where each flow adjusts its rate by increasing linearly during no congestion and decreasing multiplicatively when congestion happens (usually by halving the window size). However, "fairness" in TCP is typically defined in terms of equal opportunity to send packets rather than equal throughput in bytes. This means that while each flow may get to send a similar number of packets over time, the flow with larger packets (Flow 1) may end up with a higher throughput in bytes. Link Utilization

and RTT Variability: Given the same MWS and assuming the link capacity and RTT are constant for both flows, the utilization of the link would initially favor the flow with the larger packets. Over time, as congestion events adjust their window sizes, the share might become more balanced in terms of packets sent, but not necessarily in bytes.

2.4

A more advanced congestion control implementation, TCP Westwood is ideally suitable for networks with high bandwidth-delay products and random loss:

Estimation Bandwidth: Westwood characterizes three primary differences relative to Reno: the method in which bandwidth is assessed, its reaction to estimated bandwidth, and its reaction to real-world bandwidth. The congestion window and the slow-start threshold are set by Westwood following a congestion event based on the estimate of the ideal scenario’s bandwidths, which may lead to faster recovery than Reno. Thus, the estimations of actual bandwidth use and where deficiencies in the available bandwidth are straight.