Below is the information pertaining to the contributions made by each team member and the test cases ran on the client and server.

**Overview:**

For this project both team members collaborated equally on all parts of the project. Each person contributed to the README and code/network simulation. Below is the implementation and screenshots of the simulation with each of the cases outlined within the problem-set prompt.

A screenshot of a computer

Description automatically generatedSACK – Case 1:

A screenshot of a computer

Description automatically generated

SACK – Case 2:

A screenshot of a computer

Description automatically generated

SACK – Case 3:

Vegas – Case 1:

A screenshot of a computer

Description automatically generated

Vegas – Case 2:

A screenshot of a computer

Description automatically generated

Vegas – Case 3:

A screenshot of a computer

Description automatically generated

Per the question posted at the end of the assignment document, after running the code and analyzing the results of the two TCP flavors (SACK and Vegas) under the three different case scenarios we were able to understand the relationship between the throughput and RTT.

Essentially, higher RTT can lead to a lower throughput due to TCP using the RTT to determine the appropriate window size for data transmission. Additionally, higher delays can result in longer run times between the transmission of a packet and its acknowledgment – causing TCP to reduce its sending rate. Based off our results for both SACK and Vegas, as the end-to-end delay increased from case 1 to case 3, the throughput decreased. This reflects consistency with the expected behavior as higher delays typically negatively impact TCP throughput.

**CASE 1 Throughput Performance Comparison (SACK v. Vegas):**

In case one, the end-to-end delays were the lowest and both TCP flavors showed higher throughput compared to the other cases. SACK generally performs better than Vegas in case 1. This could be due to the differences in how the algorithms handle packet loss and recovery. Additionally, SACK allows receivers to inform the sender about which packets were received, helping to recover from packet loss efficiently.

**Explanation of SACK’s better performance in CASE 1:**

SACK’s improved performance in case 1 could be attributed to its ability to recover from packet loss more effectively. By having lower delays, the impact of loss is less severe, and SACK’s selective acknowledgement helps it to quickly recover from any losses – which also leads to a higher throughput. Vegas on the other hand may have a more conservative approach to congestion control, which results in a slightly lower throughput in cases with lower delays.

**Final Thoughts:**

Both SACK and Vegas display decreasing throughput as the end-to-end delay increases. The relative performance of SACK and Vegas vary depending on network conditions and the specific characteristics of the scenarios.