**Computer Communications and Networks (COMN)**

**2022/23, Semester 1**

**Assignment 2 Worksheet**

|  |  |
| --- | --- |
| **Forename and Surname:** | Robin Jehn |
| **Matriculation Number:** | S2024553 |

**Question 1** – Number of retransmissions and throughput with different retransmission timeout values with stop-and-wait protocol. For each value of retransmission timeout, run the experiments for **5 times** and write down the **average number of retransmissions** and the **average throughput**.

|  |  |  |
| --- | --- | --- |
| **Retransmission timeout (ms)** | **Average number of**  **retransmissions** | **Average throughput**  **(Kilobytes per second)** |
| 5 | 4773 | 65 |
| 10 | 1677 | 64 |
| 15 | 324 | 62 |
| 20 | 94 | 56 |
| 25 | 99 | 51 |
| 30 | 100 | 51 |
| 40 | 84 | 50 |
| 50 | 96 | 46 |
| 75 | 93 | 40 |
| 100 | 93 | 36 |

**Question 2** – Discuss the impact of retransmission timeout value on the number of retransmissions and throughput. Indicate the optimal timeout value from a communication efficiency viewpoint (i.e., the timeout that minimizes the number of retransmissions while ensuring a high throughput).

Retransmission timeouts are designed to ensure that lost or significantly delayed packets are resent. However, if the timeout value is set too low, even packets that are successfully delivered may trigger unnecessary retransmissions. For example, with a one-way delay of 5 ms, any timeout shorter than about 10 ms plus a small processing overhead will often cause a timeout even when the packet and its acknowledgment are not lost. This explains why timeout values of 5 ms and 10 ms result in many retry attempts—and even at 15 ms, premature retransmissions can occur. It seems like we aren’t capped by the bandwidth.

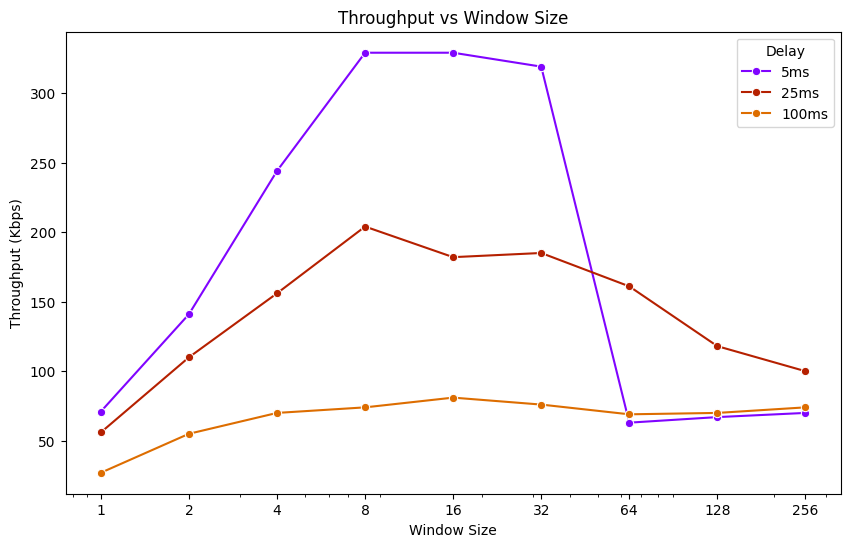
When the timeout is set above this threshold, retransmissions mainly occur when a packet or its acknowledgment is actually dropped. With a 5% drop probability in each direction, roughly 10% of packets will need to be resent, equating to around 90 packets in this scenario. Since bandwidth isn’t a limiting factor here, using a lower timeout can increase throughput because faster retransmission of lost packets accelerates data recovery without a significant penalty when packets are delivered successfully.

For our specific setup, a retransmission timeout of around 20 ms appears optimal, as it minimizes unnecessary retries while maximizing throughput. A timeout of 15 ms could still be acceptable depending on the precise network conditions. In general, for networks with variable delay, it is best to choose a timeout that exceeds the majority of round-trip times by a small margin to account for processing overhead.

**Question 3** – Experimentation with Go-Back-N. For each value of window size, run the experiments for **5 times** and write down the **average throughput**.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Average throughput (Kilobytes per second)** | | |
| **Window** **Size** | **Delay = 5ms** | **Delay = 25ms** | **Delay = 100ms** |
| 1 | 71 | 56 | 27 |
| 2 | 141 | 110 | 55 |
| 4 | 244 | 156 | 70 |
| 8 | 329 | 204 | 74 |
| 16 | 329 | 182 | 81 |
| 32 | 319 | 185 | 76 |
| 64 | 63 | 161 | 69 |
| 128 | 67 | 118 | 70 |
| 256 | 70 | 100 | 74 |

Create a graph using the results from the above table (empty example graph shown below):



**Question 4** – Discuss your results from Question 3.

Based on the experimental results from above, I set the retransmission timeout (RTO) using the formula: RTO = 2 × (Propagation Delay) + 10 ms. This strategy ensures that the timeout is long enough to avoid premature retransmissions but not so long that it unnecessarily delays recovery.

Across all propagation delays (5ms, 25ms, and 100ms), increasing the sender’s window size initially improves throughput. A larger window allows more packets to be in transit simultaneously. However, after a certain point, further increasing the window size results in a performance plateau and eventually a decline. This drop-off likely occurs because the sender transmits all packets as quickly as possible without adequate spacing. Consequently, the receiver or the underlying network buffers become overwhelmed, leading to additional packet drops.

Unsurprisingly, lower propagation delays (e.g., 5ms) yield better throughput compared to higher delays (25ms and 100ms) since shorter delays result in quicker acknowledgments and more efficient utilization of the network capacity.

**Question 5** – Experimentation with Selective Repeat. For each value of window size, run the experiments for **5 times** and write down the **average throughput**.

|  |  |
| --- | --- |
|  | **Average throughput (Kilobytes per second)** |
| **Window Size** | **Delay = 25ms** |
| 1 | 76 |
| 2 | 106 |
| 4 | 139 |
| 8 | 193 |
| 16 | 285 |
| 32 | 493 |

**Question 6** - Compare the throughput obtained when using “Selective Repeat” with the corresponding results you got from the “Go Back N” experiment and explain the reasons behind any differences.

When comparing the throughput of Selective Repeat with Go Back N, several differences emerge based on window size. For smaller window sizes (1, 2, 4, and 8), Go Back N slightly outperforms Selective Repeat because the receiver isn’t overwhelmed and the cost of retransmitting packets is relatively low. In these cases, resending a group of packets works well, and any inefficiencies in the implementation are minimal. However, as the window size increases, the limitations of Go Back N become more apparent. Since Go Back N retransmits the last N packets following an error—even if only one packet was lost—it can lead to unnecessary retransmissions and congestion. In contrast, Selective Repeat only retransmits the specific packets that were lost or received in error. This approach significantly improves efficiency at larger window sizes, leading to better overall throughput.

**Question 7** – Experimentation with *iperf*. For each value of window size, run the experiments for **5 times** and write down the **average throughput**.

|  |  |
| --- | --- |
|  | **Average throughput (Kilobytes per second)** |
| **Window Size (KB)** | **Delay = 25ms** |
| 1 | 85 |
| 2 | 123 |
| 4 | 196 |
| 8 | 142 |
| 16 | 139 |
| 32 | 157 |

**Question 8** - Compare the throughput obtained when using “Selective Repeat” and “Go Back N” with the corresponding results you got from the *iperf* experiment and explain the reasons behind any differences.

Selective Repeat achieves higher throughput because it retransmits only lost packets, while Go-Back-N requires retransmitting a lost packet along with all subsequent packets in the window. In our iperf experiments using TCP, the behavior is closer to Go-Back-N due to TCP’s cumulative acknowledgements. Additionally, TCP introduces extra overhead from larger headers and the connection setup handshake, which further reduces throughput. Additionally, we observe a large standard deviation for our iperf measurements. Using a window size of 16 I sometimes see 70 kB/s and other times 300 kB/s. While I am not entirely sure why this happens, I suspect that there could be individual packets during the handshake that create a bottleneck if dropped. Hence, we get low numbers when they are dropped and high numbers if not. To investigate the performance further we should transmit smaller and bigger files to analyse the impact of the handshake overhead.