INF-2300: Computer Communication

Assignment 3 - Reliable Transport Protocol

Sera Madeleine Elstad

UiT id: sel063@uit.no

GitHub user: SeraMadeleine

October 22, 2023

Introduction

This project aims to create a thorough grasp of the algorithms designed to facilitate reliable data transfer from the transport layer of the OSI-stack model to the application layer above. In this project, one of two critical algorithms, Go-Back-N (GBN) or Selective-Repeat, had to be implemented. The chosen algorithm for the current assignment is GBN, which is well-known in the field of dependable data transfer.

When sending data through a network, three significant issues can arise. These include packet loss, latency, and the chance of data corruption during transmission. The chosen algorithm is critical to overcoming these problems and ensuring that data is delivered and received accurately and reliably.

Technical background

The OSI-stack model

The OSI ((Open Systems Interconnection) model, consisting of seven layers, facilitates computer data transmission, with layers managing different aspects of communication. [1]. Physical connectivity and synchronization are managed by the Physical Layer. The Data Link Layer manages framing and access control as well as ensuring error-free data delivery. The Network Layer is responsible for routing data between networks and assigning IP addresses. The Transport Layer connects networks and is responsible for data segmentation, integrity, and delivery. Communication sessions are managed by the Session Layer. The Presentation Layer is responsible for data format transformation. The Application Layer interacts with programs by acting as a user interface for data access and display. All the layers are displayed in figure 1.

While the OSI model provides a comprehensive reference framework for network communication, the Transmission Control Protocol (TCP) model is the actual framework often used on the internet [1].

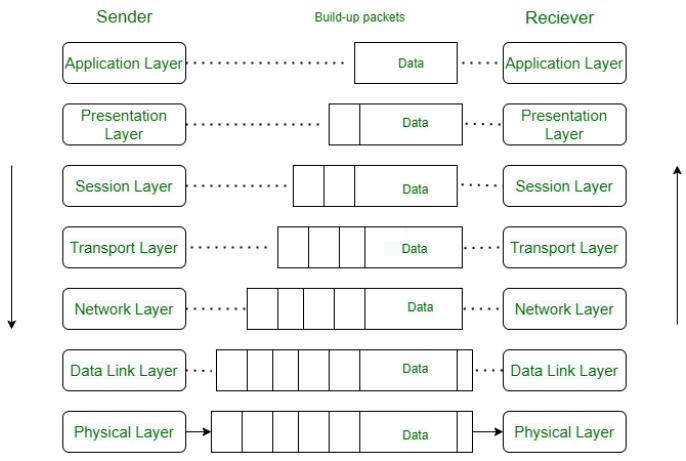

Figure 1: OSI-Model, [1]

Transmission Control Protocol (TCP)

TCP (Transmission Control Protocol) is a key part of the Internet protocol suite, ensuring reliable data delivery between devices. It breaks data into small packets for transmission and guarantees they are reassembled correctly. TCP maintains order and reliability, supports full-duplex communication, and includes features for flow control, error control, and congestion control [2].

Go-Back-N (GBN) and Selective-Repeat

The Go-Back-N protocol is a sliding window protocol employed in computer networks for reliable data transfer. It operates as a sender-based protocol, allowing the sender to transmit multiple packets without waiting for individual acknowledgments. In Go-Back-N, the receiver sends cumulative acknowledgments, indicating the last successfully received packet. If any packet is lost or corrupted, the receiver sends a negative acknowledgment (NACK) for the missing packet. In response, the sender retransmits all the packets in the window, starting from the lost packet. Go-Back-N employs a sliding window mechanism, sequence numbers, cumulative acknowledgments, timeouts, and a NACK mechanism, making it relatively simple to implement [3].

The Selective Repeat protocol is another sliding window protocol used for reliable data transfer in computer networks. Unlike Go-Back-N, it operates as a receiver-based protocol, allowing the receiver to individually acknowledge each packet. In Selective Repeat, the sender sends packets within a window and waits for acknowledgments for each packet. If a packet is lost or corrupted, the receiver sends a NACK for that specific packet, and the sender retransmits only the lost packet. This protocol maintains a buffer to store out-of-order packets and handles them efficiently. However, Selective Repeat is more complex and requires additional memory and processing power compared to Go-Back-N [3].

Both Go-Back-N and Selective Repeat protocols utilize a sliding window mechanism, sequence numbers, and timers to manage lost or corrupted packets. However, they differ in several key aspects. In Go-Back-N, if any packet within the window is lost or corrupted, the sender retransmits all packets from the lost one to the last one sent. In contrast, Selective Repeat retransmits only the damaged packet. While Go-Back-N is less complex and doesn't require sorting, Selective Repeat is more intricate and necessitates sorting for out-of-order packets. Another significant difference is that Go-Back-N discards out-of-order packets, causing retransmission of the entire window if a corrupt packet is received. In contrast, Selective Repeat accepts out-of-order packets and retransmits only the suspected packet. Therefore, the choice between these protocols depends on the specific network requirements and constraints [3].

Design and Implementation

From app

This method is in charge of transmitting data from the application layer to the network layer. It aims to send data from the application layer (Alice) to the network layer. It separates data into packets, assigns sequence numbers, computes checksums for data integrity, manages the sliding window, and ensures that data is transmitted to the network layer on time while providing means to deal with any timeouts.

From network

The method is in charge of receiving data from the network layer and routing incoming packets, whether they are acknowledgment packets (ACKs) or data packets, to their proper destinations.

When a data packet arrives, the procedure routes it to Bob, who is in charge of data packet processing. This procedure includes a vital step of computing and confirming the checksum, which serves as a safeguard against potential data corruption. If the data is unchanged and matches the sequence number Bob expected, an acknowledgment (ACK) is sent back to acknowledge the successful reception of the data. The data is then sent to the application layer for additional processing. If Bob gets a data packet for which he has previously provided an acknowledgment, it is possible that the acknowledgment was lost in transit to Alice. To fix this, Bob sends a new ACK to confirm the message. Furthermore, if Bob receives a data packet with a sequence number that exceeds his expectations, suggesting a packet from the future, he ignores the unexpected packets.

If the incoming packet is an acknowledgment (ACK), the procedure sends it to Alice for processing. The focus here is on assuring the correct receiving of the acknowledgment and validating its alignment with the predicted sequence number. If she receives data with a greater ack-number than predicted, she knows Bob received the previous packets, and she can shift her window such that it no longer contains their packets.

These methods collectively empower the transport layer to adeptly handle the reception of data packets and acknowledgments, while seamlessly managing the sliding window. The strategy employed ensures the integrity of the data through checksum validation and guarantees the reliable delivery of data, even in cases of packet delay or loss. This approach ensures the efficient functioning of the communication system, maintaining data integrity and reliability.

Timer

The transport layer's management of timeout events is critical to ensuring the reliability of data transfer. A timeout indicates that an acknowledgment for a submitted packet was not received within the expected timeframe.

When a timeout event happens, one of the methods takes action. It retransmits unacknowledged packets as a safety net to ensure critical data delivery. This strategy mitigates the impact of unacknowledged packets in the network by commencing retransmissions.

The other way ensures that the timer used to handle timeouts is reset with care, preventing the generation of duplicate timer threads, which could possibly overload the system. Importantly, the timer is reset only when there are still unacknowledged packets within the packet window. This strategic approach optimizes the timeout procedure, which contributes greatly to data transmission reliability in the transport layer.

These techniques work in unison to efficiently handle timeout events. When a timeout occurs, the handle timeout method responds by retransmitting unacknowledged packets. The reset timer technique ensures that the timer is properly handled by avoiding the generation of duplicate timer threads and restarting the timer only when there are unacknowledged packets in the window. This reliable timeout technique contributes to the transport layer's data delivery reliability.

Chechsum

The integrity of sent data is a primary problem in data communication. The transport layer addresses this by using a process known as "Checksum." This method computes a checksum for a given dataset by doing a bitwise XOR operation on each byte.

This checksum is an important tool for validating data integrity and detecting any corruption during transmission. The transport layer can verify whether data has been altered throughout its transit through the network by comparing the calculated checksum at the receiving end to the original checksum supplied with the data.

Testing

The program has undergone comprehensive testing to see how well it performs under diverse scenarios. The tests show that the application constantly completes, even whether there are packet drops, corruption, or delays, all of which occur with probabilities as high as 0.9. It is important to be aware that the program may occasionally stop at packet 9. However, restarting the software has the desired result. When subjected to a combination of drop, corruption, and delay situations, the program displays robustness with a probability up to 0.9. Even under these adverse conditions, it maintains its dependability and integrity.

Discussion

During testing, it became evident that the program sometimes appears to halt at packet 9. The program does not finish as intended. One workaround is to close the terminal or forcefully stop the program and then restart it, allowing it to eventually reach a "finished" state.

Another noticed issue is that, even after the program reaches a "finished" state, it continues to display messages suggesting that Alice is sending packets. This problem appears to be in the precode. It declares itself "finished" as soon as Bob receives the last packet, even if Alice has not yet gotten the associated acknowledgement (ACK). As a result, the program continues to run until Alice receives the missing ACK.

While this issue might potentially be addressed inside the program's logic, it appears that a purposeful decision was made not to account for it. Nonetheless, it raises concerns regarding program behavior and possible improvements in dealing with certain situations more gently.

Conclusion

Finally, this project gave a thorough understanding of trustworthy data transmission techniques, as well as their implementation and testing. It emphasized the significance of these algorithms in maintaining data integrity and dependable communication in complicated network contexts. While difficulties arose, they served as opportunities for future enhancements, strengthening the program's robustness and usability.

References

[1] What is OSI Model Layers of OSI Model, September 2023. [Online; accessed 19. Oct. 2023].

[2] What is Transmission Control Protocol TCP, April 2023. [Online; accessed 19. Oct. 2023].

[3] Difference Between Go Back N and Selective Repeat Protocol, March 2023. [Online; accessed 19. Oct. 2023].