Experiment No. -08

Experiment Title: Comparative Analysis of Flow Control Protocols in Data Communication

Objective:
The primary objective of this experiment is to study and compare different flow control protocols in the context of data communication and computer networks. Flow control plays a crucial role in ensuring efficient and reliable data transfer between devices in a network. By conducting this experiment, we aim to gain insights into the performance characteristics, advantages, and limitations of various flow control protocols.

Equipment and Materials:

Computers with network interfaces
Network simulation software (e.g., Cisco Packet Tracer, GNS3)
Ethernet cables
Switches and routers (if not using simulation software)
Data communication devices (such as PCs or laptops)
Protocol analyzer tool (Wireshark)
Experimental Setup:

Establish a simple network topology using either physical devices or network simulation software. Ensure that the network consists of at least two nodes capable of data communication.

Select and implement different flow control protocols for the experiment. Common flow control protocols include:

What is flow control?
Flow control is a technique used to regulate data transfer between computers or other nodes in a network. Flow control ensures that the transmitting device does not send more data to the receiving device than it can handle. If a device receives more data than it can process or store in memory at any given time, the data is lost and needs to be retransmitted.

The purpose of flow control is to throttle the amount of data transmitted to avoid overwhelming the receiver's resources. This is accomplished through a series of messages that the receiver transmits to the sender to acknowledge if frames have been received. The sender uses these messages to determine when to transmit more data. If the sender does not receive an acknowledgement (ACK), it concludes that there has been a problem with the transmission and retransmits the data.

Flow control is implemented in different ways, depending on how the sender and receiver handle messages and track data frames. There are two basic approaches to flow control: stop and wait and sliding window. The stop-and-wait approach is the simplest to implement, but it is not as efficient as sliding window, which delivers better network performance and utilizes network resources more effectively.

Stop-and-Wait flow control

In the stop-and-wait approach, the sender segments the data into frames and then transmits one frame at a time to the receiver, which responds to each frame with an ACK message. This process occurs through the following steps:

1. The sender transmits a data frame to the receiver.

2. The sender waits for the receiver to respond.
3. Upon receiving the frame, the receiver transmits an ACK to the sender.
4. Upon receiving the ACK, the sender sends the next frame to the receiver and waits for the next ACK. If the sender does not receive an ACK within a defined time limit, known as a *timeout*, the sender retransmits the same frame.
5. The process continues until the sender has finished transmitting all the data to the receiver.

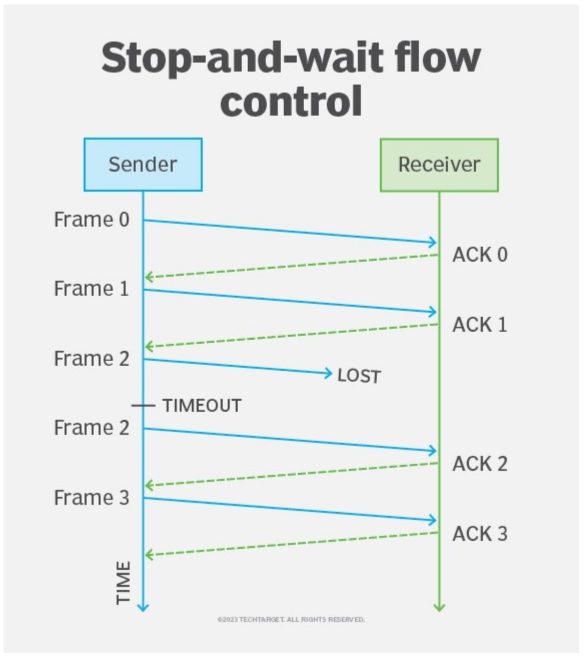

Figure 1. How stop-and-wait flow control works

Stop-and-Wait Protocol:
Figure 1 illustrates how this exchange works. In this case, the sender starts by transmitting Frame 0 and then waiting for the ACK. When Frame 0 reaches its destination, the receiver sends ACK 0 to the sender.

After receiving ACK 0, the sender transmits Frame 1 and waits for ACK 1. When that arrives, the sender transmits Frame 2 and waits again. This time, however, the sender does not receive ACK 2 before the timeout occurs, so it retransmits Frame 2. The frame now arrives at its destination, so the receiver sends ACK 2. When the sender receives ACK 2, it transmits Frame 3, which is also acknowledged by the receiver.

Stop and wait belongs to a category of error control mechanisms called *automatic repeat requests* (ARQs), which rely on the use of ACKs to determine if a data transmission was successful or if retransmission is needed. Other ARQs include Go-Back-N ARQ and Selective Repeat ARQ, both of which use the sliding window protocol.

Stop and wait is simpler to implement than sliding window. It is also fairly reliable because the sender receives an ACK for each frame successfully transmitted to the receiver. These qualities, however, also make data communications much slower, which can be exacerbated by long distances and heavy traffic. The stop-and-wait approach also tends to underutilize network resources.

Sliding Window (Selective Repeat and Go-Back-N)

The sliding window approach addresses many of the issues that come with stop and wait because the sender can transmit multiple frames at once without having to wait for an ACK for each frame. However, this approach also comes with additional complexity.

When first connecting, the sender and receiver establish a window that determines the maximum number of frames the sender can transmit at a time. During the transmission, the sender and receiver must carefully track which frames have been sent and received to ensure that all the data reaches its destination and is reassembled in the correct order.

Sliding window flow control can be implemented using one of two approaches: Go-Back-N and Selective Repeat. With the Go-Back-N approach, the sender can send one or more frames but never more frames than the window allows. As the receiver acknowledges the frames, the sender moves to the next batch, or window, of frames that can now be sent. If there is a problem with a transmitted frame, the sender retransmits all the frames in the current window.

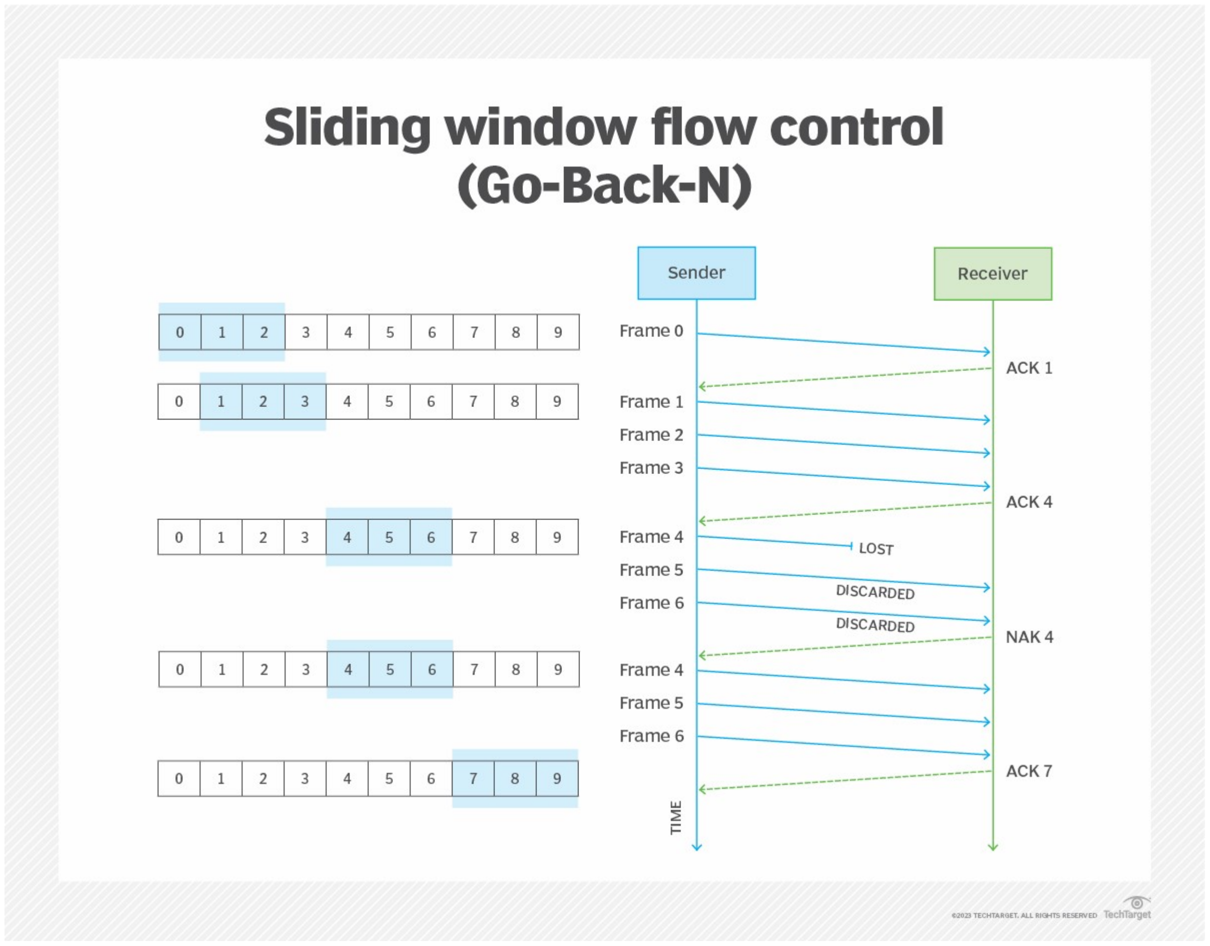

Figure 2. How Go-Back-N type of sliding window flow control works

Figure 2 shows an example of how Go-Back-N works. In this case, the window consists of only three frames -- initially, Frames 0 through 2. The sender begins by transmitting Frame 0 to the receiver. Upon receiving Frame 0, the receiver sends an ACK that specifies the next frame to send (Frame 1), rather than specifying the frame that has just been received.

When the sender receives ACK 1, it moves the window over by one position, dropping Frame 0 and adding Frame 3. The sender then transmits Frames 1, 2 and 3, which represent the window's entire contents. The sender does not necessarily need to send Frame 0 first, followed by Frames 1 through 3. This is illustrated in Figure 2 to demonstrate how the process works.

Upon receiving the three frames, the receiver sends a cumulative ACK that specifies the next frame to send, which is Frame 4. The ACK indicates that the receiver now has all the preceding frames (0 through 3).

When the sender receives ACK 4, it adjusts the window so that it now includes Frames 4 through 6 and then transmits those frames. This time, however, Frame 4 gets lost in the transmission, while Frames 5 and 6 reach their destination. Upon receiving Frame 5, the receiver detects that Frame 4 is missing and sends a negative acknowledgement (NAK) that specifies Frame 4. At the same time, the receiver discards Frames 5 and 6.

When the sender receives the NAK, it retransmits Frames 4 through 6 and waits for the ACK. The frames arrive with no errors the second time around, so the receiver returns an ACK indicating that the sender can now transmit Frame 7. The sender adjusts the window accordingly and transmits the next set of frames, starting with Frame 7.

The Selective Repeat approach is similar to Go-Back-N. The primary difference is that Selective Repeat does not retransmit the entire window if there is an error, only the individual frame in dispute. Selective Repeat does not support cumulative ACK messages like Go-Back-N, so each ACK is specific to the frame that was just received, which is what enables the sender to identify the precise frame that needs to be retransmitted.

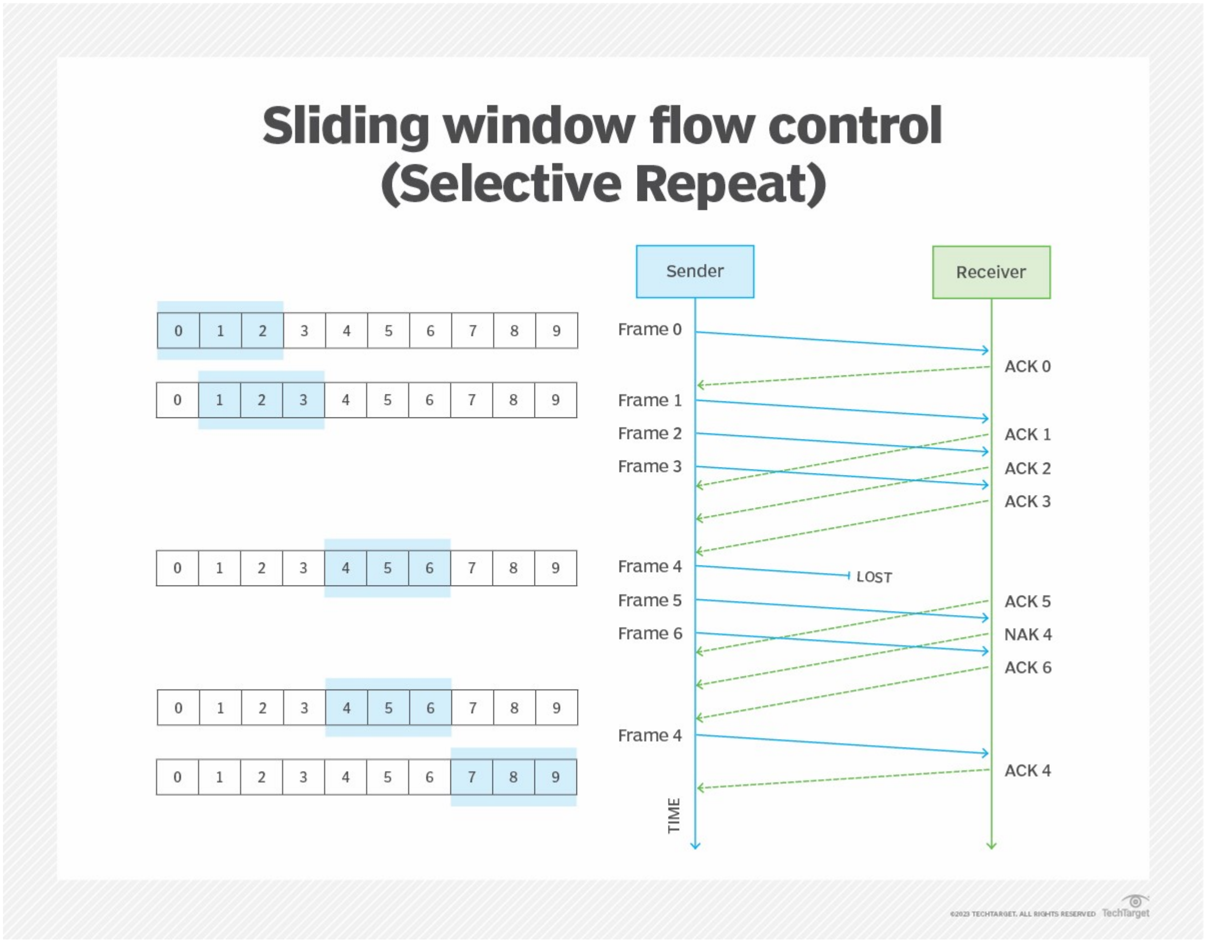

Figure 3. How Selective Repeat sliding window flow control works

Figure 3 illustrates an example of the Selective Repeat process. After transmitting Frame 0, the sender receives an ACK, so it transmits Frames 1 through 3 and receives an ACK for each one. The sender then transmits Frames 4 through 6. When Frames 5 and 6 arrive at the receiver, but not Frame 4, the receiver sends ACK 5 and ACK 6, along with NAK 4. The sender responds to the NAK by retransmitting Frame 4. Upon receiving Frame 4, the receiver sends an ACK. The sender then adjusts the window and transmits the next three frames, starting with Frame 7.

Both Selective Repeat and Go-Back-N are more efficient than the stop-and-wait approach, but there are important differences between the two sliding window approaches. The Go-Back-N approach can consume more bandwidth because all the frames in a window are retransmitted if an error occurs. However, it is not as complex to implement as Selective Repeat and does not require the same amount of system resources. Selective Repeat comes with greater management overhead because the frames must be tracked and sorted throughout the data transmission.

Summary of Sliding Window Protocol (Selective Repeat and Go-Back-N):
The Sliding Window protocol is an advanced flow control mechanism that allows for the concurrent transmission of multiple frames, enhancing data transfer efficiency. Two variations of this protocol are Selective Repeat and Go-Back-N.

Selective Repeat:

1. **Sender (S) Side:**
    - S divides the data stream into a sequence of frames and sends a window of frames to R without waiting for individual acknowledgments.

- S maintains a record of which frames were sent in the current window.
- Upon receiving acknowledgments, S updates its record, and any unacknowledged frames are resent.
- The receiver (R) individually acknowledges correctly received frames.

2. **Receiver (R) Side:**
   - R receives a window of frames and individually acknowledges each correctly received frame.
   - If a frame is damaged or lost, R requests retransmission only for that specific frame.
   - Out-of-sequence frames are buffered until the missing frame is received.

Go-Back-N:

1. **Sender (S) Side:**
   - S sends a continuous stream of frames without waiting for individual acknowledgments.
   - S maintains a "window" of frames awaiting acknowledgment.
   - If an acknowledgment is not received within a specified timeout, S retransmits all frames in the current window.
2. **Receiver (R) Side:**
   - R receives frames and individually acknowledges each correctly received frame.
   - If a frame is damaged or lost, R discards all subsequent frames until the correct one is received.
   - R sends cumulative acknowledgments, indicating the highest correctly received frame.

Leaky Bucket Algorithm

The Leaky Bucket Algorithm is a traffic shaping mechanism used to control the rate at which data is sent into a network. It is often employed in scenarios where there is a need to smooth out bursty traffic and ensure a consistent flow of data. In this algorithm, the "bucket" has a finite capacity, and data is added to it at a constant rate. If the bucket overflows, excess data is discarded or marked for slower transmission. This helps in controlling the rate at which data is released into the network, preventing congestion. The diagram below illustrates the concept of the Leaky Bucket Algorithm, where incoming data is added to the bucket, and the network allows a controlled output rate.

Leaky Bucket Algorithm:

The Leaky Bucket Algorithm is a traffic shaping mechanism used to control the rate at which data is sent into a network. It is commonly employed in scenarios where there is a need to smooth out bursty traffic and ensure a consistent flow of data. The algorithm is named after its analogy with a bucket that has a leak.

Basic Operation:

Bucket Structure:

The "bucket" has a finite capacity, representing the maximum amount of data that can be transmitted in a given time period.
Incoming data is added to the bucket at a constant rate.
Leakage:

The bucket has a leakage mechanism, allowing it to release data at a controlled rate.
If the incoming data rate exceeds the leak rate, the bucket may overflow.
Data Transmission:

Data is transmitted into the network at the rate determined by the leaky bucket, preventing bursts of data that could lead to network congestion.
If the bucket is full, excess data is either discarded or marked for slower transmission.
Configure the network parameters, such as data transfer rate, latency, and error rates, to create realistic network conditions.

Design a data transfer scenario where a significant amount of data needs to be transferred between the nodes in the network.

Conduct the following experiments for each flow control protocol:
a. Measure and compare the throughput of data transfer under normal conditions.
b. Introduce network congestion or errors and observe how each flow control protocol handles these situations.
c. Analyze the impact of varying parameters like window size in sliding window protocols on the overall performance.

Use a protocol analyzer tool (e.g., Wireshark) to capture and analyze the network traffic during the experiments. This will provide insights into the efficiency of each flow control protocol.

Data Collection and Analysis:

Record the throughput, delay, and any retransmission events for each flow control protocol in different scenarios.

Analyze the captured network traces to identify the behavior of each protocol under various conditions.

Compare the performance of different flow control protocols based on the collected data and draw conclusions regarding their suitability for specific network conditions.

Conclusion:
Summarize the findings of the experiment, highlighting the strengths and weaknesses of each flow control protocol. Provide recommendations for selecting the most appropriate flow control protocol based on specific network requirements and conditions. Discuss potential areas for further research and improvement in flow control mechanisms.