CSN-341
Computer Networks
Assignment - 4

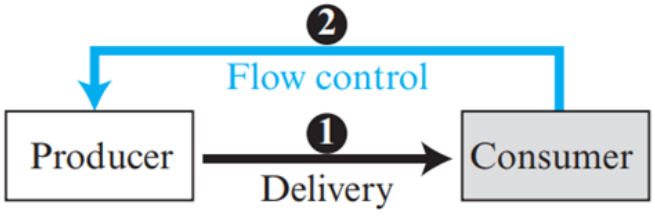
Group-6

Name	Enrollment No
Meet Sindhav	22114053
Mohammed Haaziq	22114055
Aditya Mundada	22114058
Nayan Kakade	22114060
Sarvesh Baheti	22114087
Roopam Taneja	22125030

Question 1: Why is flow control important in the transport layer, and how does TCP implement flow control through mechanisms like sliding windows? Discuss how improper flow control can lead to issues like buffer overflow and network congestion, and explain the potential impact on overall network performance.

During the process to process communication between transport layers sometimes items(packets) are produced faster than they can be consumed, the consumer can be overwhelmed and may need to discard some items. Flow control solves this issue by preventing loss of data items at the consumer site.

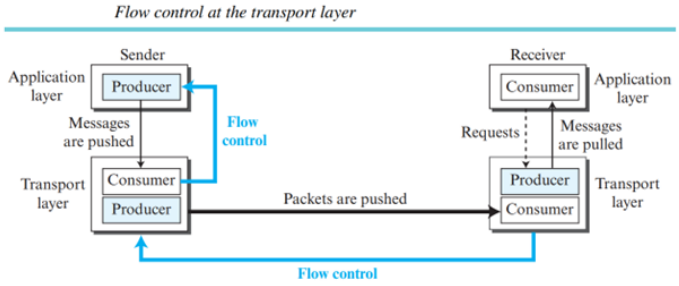
Flow control is especially required where pushing is used for delivery of items i.e. If the sender delivers items whenever they are produced without a prior request from the consumer the delivery is referred to as pushing.



a. Pushing

In communication at the transport layer, we are dealing with four entities: sender process, sender transport layer, receiver transport layer, and receiver process. The sending process at the application layer is only a producer. It produces message chunks and pushes them to the transport layer. The sending transport layer has a double role: it is both a consumer and a producer. It consumes the messages pushed by the producer. It encapsulates the messages in packets and pushes them to the receiving transport layer. The receiving transport layer also has a double role, it is the consumer for the packets received from the sender and the producer that decapsulates the messages and delivers them

to the application layer. The last delivery, however, is normally a pulling delivery; the transport layer waits until the application-layer process asks for messages. We need at least two cases of flow control: from the sending transport layer to the sending application layer and from the receiving transport layer to the sending transport layer.



Implementation of flow contro

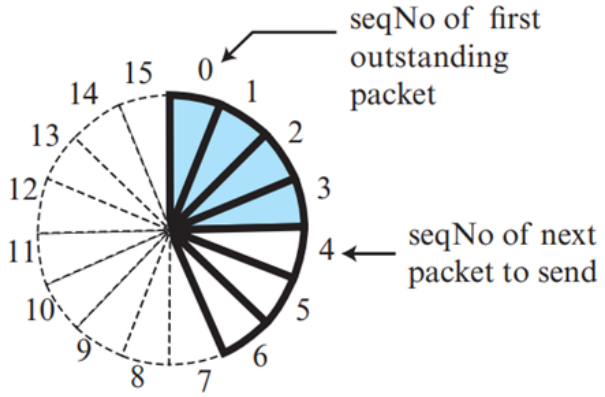
Flow control can be implemented in several ways,few methods are :

1. **Buffers:** One of the solutions is normally to use two buffers: one at the sending transport layer and the other at the receiving transport layer. A buffer is a set of memory locations that can hold packets at the sender and receiver. The flow control communication can occur by sending signals from the consumer to the producer. When the buffer of the sending transport layer is full, it informs the application layer to stop passing chunks of messages; when there are some vacancies, it informs the application layer that it can pass message chunks again. When the buffer of the receiving transport layer is full, it informs the sending transport layer to stop sending packets. When there are some vacancies, it informs the sending transport layer that it can send packets again

   This method is then modified as per requirement to make transport layer protocols that have flow control like Stop-and-Wait Protocol etc.

   Flow control along with error control can be solve by using sequence numbers and sliding window.

2. **Sliding window:** Since the sequence numbers used modulo 2^m, a circle can represent the sequence numbers from 0 to $2^m - 1$ .The buffer is represented as a set of slices, called the sliding window, that occupies part of the circle at any time. At the sender site, when a packet is sent, the corresponding slice is marked. When all the slices are marked, it means that the buffer is full and no further messages can be accepted from the application layer. When an acknowledgment arrives, the corresponding slice is unmarked. If some consecutive slices from the beginning of the window are unmarked, the window slides over the range of the corresponding sequence numbers to allow more free slices at the end of the window.



a. Four packets have been sent.

Improper flow control in a network context can lead to critical issues like buffer overflow and network congestion, which can degrade overall network performance.

1. **Buffer Overflow:** This often occurs when the receiving end cannot signal the sender to slow down, the buffer fills up, leading to an overflow. the buffer (temporary storage) in a receiver or intermediate network device (such as a router) becomes full because the incoming data is faster than it can be processed or forwarded.
2. **Network Congestion:** Flow control also helps prevent network congestion by ensuring that traffic is spread evenly across the network. If flow control mechanisms are faulty, and multiple sources send data at rates that exceed the network's capacity to handle it, congestion builds up in the form of packet collisions, queuing delays, and packet drops.

Thus the overall performance of the network is affected due to **increased latency**, packet loss as excess data is either dropped or overwritten, leading to loss of important information, which needs to be re transmitted, security risk as it may overwrite adjacent memory, including important control information like return addresses, function pointers, or other sensitive data. and **decreased throughput**.

Question 2: Why are port numbers significant in the transport layer, and how do they facilitate multiplexing and demultiplexing of data streams? Provide examples of specific port numbers associated with well-known services, such as HTTP and DNS, and discuss the potential issues that can arise due to port number conflicts or misuse.

A port is a virtual point associated with a specific process or service. A port number is the logical address of each application or process that uses a network or the Internet to communicate. A port number uniquely identifies a network-based application on a computer. Each application/program is allocated a 16-bit integer port number. This number is assigned automatically by the OS, manually by the user or is set as a default for some popular applications.

1. **Multiplexing:** Gathering data from multiple application processes of the sender, enveloping that data with a header, and sending them as a whole to the intended receiver is called multiplexing. When multiple applications on a device want to send data over the network, each application is assigned a unique port number. The transport layer uses these port numbers to combine (or multiplex) the data from various applications into a single outgoing stream. This allows multiple applications to share the same network connection.
2. **Demultiplexing:** Delivering received segments at the receiver side to the correct app layer processes is called demultiplexing. When data arrives at the receiving device, the transport layer looks at the port numbers in the packets to determine which application the data belongs to. This process ensures that the data is correctly delivered to the right application by separating (or demultiplexing) it based on the port numbers.

Some popular port numbers are:

- HTTP : 80
- TCP : 20/21
- DNS : 53
- SMTP : 25
- SSH : 22
- POP3 : 110
- IMAP : 143

The potential issues that can arise due to port number conflicts or misuse:

- **Service Interruption:** If two applications try to use the same port number, they may conflict and result in one or both failing to work properly. For example, if two web servers attempt to use port 80 on the same machine, only one will be able to bind to it, causing the other to fail.

- **Unauthorized Access (Port Hijacking):** Malicious applications can exploit well-known ports (e.g., port 80 for HTTP) by pretending to be legitimate services. This can result in unauthorized access or exploitation, especially if there is a security vulnerability in the service operating on that port.
- **Performance Degradation:** Port conflicts can cause resource contention, slowing down the performance of the affected applications. Repeated retries to establish connections or the unavailability of a service can degrade overall system performance
- **Data Leakage:** Misuse of ports, especially in shared environments, can lead to sensitive data being sent to the wrong destination, particularly if the wrong application receives data intended for another service.
- **Security Vulnerabilities:** Misconfigured or open ports can expose systems to attacks. For example, if unnecessary services are running on default ports (e.g., an open SSH port 22), attackers might target those for brute-force or other security exploits.

Question 3: Compare and contrast the Selective-Repeat and Go-Back-N protocols in the context of reliability and efficiency. How does piggybacking in bidirectional communication improve efficiency, and what are the practical considerations when implementing piggybacking in real-world applications?

Go Back N:
The Go back N protocol has a receive window size of 1 and a sending window size of maximum $2^m - 1$. The Acknowledgement numbers are cumulative and declare the sequence number of next expected packet.

It maintains a timer that ensures reliability is maintained. When the first packet is sent, the timer is started.

The send window contains the copy of all packets currently outstanding, and as soon as an acknowledgment arrives for an outstanding packet, all outstanding packets before it are marked as acknowledged, and the send window moves forward. Also the timer is restarted. A corrupted acknowledgment is discarded. If no acknowledgment is received until a timer expires, then the timer is reset, and all outstanding packets are resent.

The receive window is always looking for a particular packet. We wait until that packet arrives. If an unexpected packet arrives it is simply discarded and we send back the acknowledgment corresponding to the expected packet.

- **Reliability:** Thus, discarding out-of-order packets and error prone packets and acknowledgments as well as resending packets not acknowledged by maintaining a timer ensure that all packets are delivered to the receiving application in order and corrupt data is correctly discarded. Also, duplicate data is correctly identified and discarded. This makes the protocol reliable.
- **Efficiency:** It has a low efficiency because lets say we are looking for a packet with sequence number 2 and we receive packets in order 3,4,5,2 then the packets 3,4,5, are all discarded and on expiration of the timer at the sender side, they must be resent. This is because the receiving layer doesn't have any mechanism to store out of order packets temporarily, order them and then send them to the application layer.

Selective Repeat:
The Selective Repeat protocol has a receive window size equal to send window size of maximum $2^{(m-1)}$. The Acknowledgement numbers are the actual sequence numbers of each arrived packet.

This too maintains a timer to ensure that reliability is maintained. The send window contains the copy of all packets currently outstanding, and as soon as an acknowledgment arrives for an outstanding packet, that packet is marked as appropriately received. A corrupted acknowledgment is discarded. Once the timer expires, all the packets which are not acknowledged in the window are resent and the timer is reset. The window slides over a contiguous segment of packets already acknowledged at the start of the window and if this happens, the timer is restarted.

The receiving window now stores packets even if they arrive out of order and sends their acknowledgment. It slides only when a contiguous segment of packets at the start of the window are already received and hence delivered to the application layer above in order.

- **Reliability:** We now shift the out-of-order concern to the receive layer and thus packets are ultimately delivered to the application in order. Duplicate packets are simply discarded, and resending unacknowledged packets again after timer expiry ensures that all packets are sent and none is left unreceived.
- **Efficiency:** This is more efficient because all we must sent is unacknowledged packets again and even if the packets are received at receiving layer out of order it is ensured that they need not be resent as receiving layers stores them and sends their acknowledgements.

Thus, Go-Back-N is simpler and works efficiently in low-error networks but suffers in bandwidth utilization when errors are frequent.

Selective repeat is more efficient and reliable in error-prone environments, though it comes with higher complexity due to the need for more sophisticated buffering and handling of individual acknowledgments.

Piggybacking:
In order to ensure bidirectional communication between two devices A and B both must simultaneously send data packets as well as acknowledgments in both the directions.

When a packet is carrying data from A to B, it can also carry acknowledgment feedback about arrived packets from B; when a packet is carrying data from B to A, it can also carry acknowledgment feedback about the arrived packets from A. This optimisation is called piggybacking.

This ensures efficiency as:

- Protocol overhead reduces with number of packets.
- Efficient bandwidth utilisation as total number of packets to be transmitted is reduced.
- Latency decreases due to quicker data exchange.

Some practical considerations are:

- It increases implementation complexity as devices must manage the combining of ACKs with data and handle potential issues such as packet loss or reordering.
- Delayed Acknowledgments: Since ACKs are delayed until the receiver has data to send, there is a risk of increased latency if the receiver does not have immediate data to transmit. This can be mitigated by implementing a timeout mechanism to ensure that ACKs are sent within a reasonable time frame.
- Error Handling: Combining data with ACKs can complicate error detection and correction processes, as the loss or corruption of a piggybacked packet affects both the data and the acknowledgment, potentially requiring more sophisticated error handling mechanisms.

Question 4: In scenarios where low latency is critical, how would you evaluate the trade-offs between using TCP and UDP? Consider the impact on reliability, error correction, and the specific needs of applications: a) online gaming b) video conferencing c) stock trading.

a) Online Gaming

Requirements:

- Low latency and fast data transmission.
- Real-time interactions between players.

UDP

- **Pros:** Offers lower latency since it avoids connection setup and does not require acknowledgments for received packets. Suitable for real-time interactions where a few lost packets do not significantly affect gameplay (e.g., character position updates).
- **Cons:** No built-in reliability or error correction, leading to potential loss of important game data. Developers often implement their own mechanisms to handle critical data (e.g., health status).

TCP

- **Pros:** Provides reliability, ensuring that packets arrive in order and without errors. Ideal for non-time-sensitive data, such as initial game state loads or chat messages.
- **Cons:** Higher latency due to connection establishment, acknowledgments, and retransmissions. Not suitable for real-time game state updates where speed is crucial.

Conclusion: UDP is typically preferred for real-time data, while TCP can be used for less time-sensitive communication.

b) Video Conferencing

Requirements:

- Low latency for smooth audio/video transmission.
- Tolerant of some data loss (e.g., dropped frames).

UDP

- **Pros:** Allows for quicker transmission of audio and video packets, essential for real-time communication. Small amounts of packet loss may result in minor artifacts (e.g., a brief pause in video) but typically won't disrupt the overall call.
- **Cons:** Without built-in reliability, there is a risk of losing important packets, which can affect quality if packet loss is significant.

TCP

- **Pros:** Guarantees delivery and order of packets, which can be useful for text chat or important data transfers during calls.
- **Cons:** Higher latency due to retransmissions and connection overhead, which can hinder real-time performance.

Conclusion: UDP is generally favored for the primary media stream, while TCP can be used for supplementary features like text chat.

c) Stock Trading

Requirements:

- Extremely low latency to ensure timely execution of trades.
- High reliability to prevent loss of critical transaction data.

UDP

- **Pros:** Can be used for fast, frequent updates (e.g., market data feeds) that benefit from low latency.
- **Cons:** Lack of reliability means that critical transaction data may be lost, which is unacceptable in a trading environment.

TCP

- **Pros:** Ensures that all transactions are received and processed reliably, maintaining data integrity crucial for financial transactions. It provides order and guarantees against lost packets.
- **Cons:** Potentially higher latency, but often still acceptable in trading systems where the reliability of data is paramount.

Conclusion: TCP is preferred for actual transaction processing, while UDP might be used for high-speed data feeds where slight data loss is tolerable but still needs careful management.

Question 5: Discuss the importance of sequence numbers and acknowledgments in TCP. How do these mechanisms ensure reliable data transmission and prevent issues of: a) packet duplication b) out-of-order delivery c) Provide an example of how TCP recovers from lost or out-of-order packets.

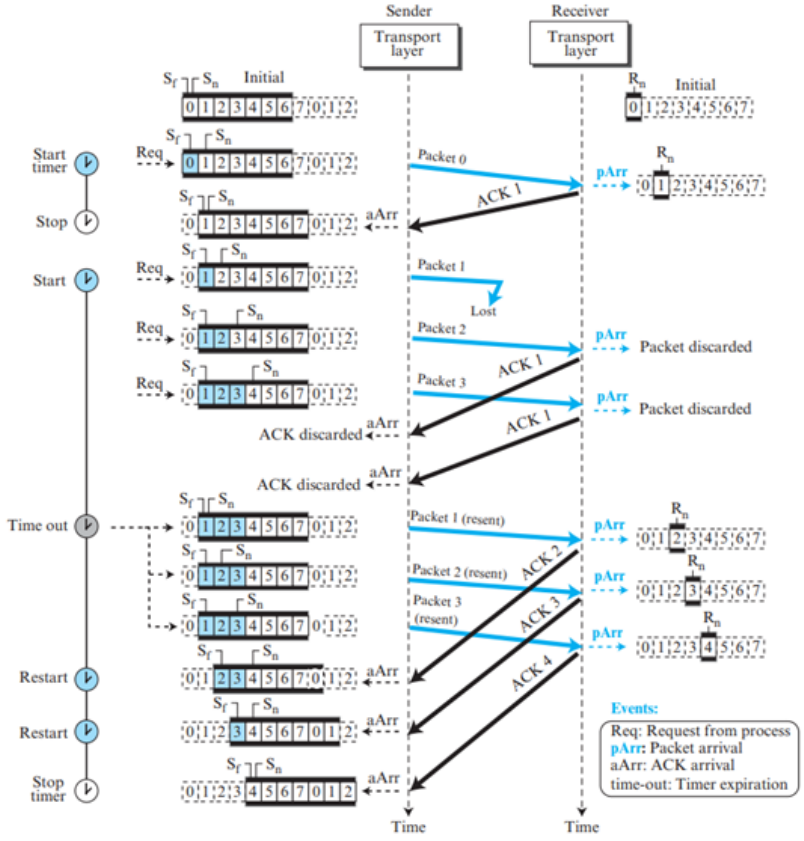
Sequence Number:
Each packet which is transferred between the sending transport layer and receiving transport layer is assigned a number. This number is known as sequence number and is stored in the packet header. The header decides the number of bits (m) which the sequence number will be. The sequence numbers range from 0 to $2^m - 1$ that is the sequence numbers are modulo 2^m. For example, if the value of m is 3, then the sequence numbers are 0,1,2,3,4,5,6,7,0,1,2,3 ...

Acknowledgements:
When the receiving transport layer receives a packet, then it sends a message to the sending transport layer indicating that the packet has arrived safe and sound. This message is known as acknowledgement.

Issue of Packet Duplication:
Sender sends packet and starts a timer. When the receiver receives a packet uncorrupted, it sends the acknowledgement (ACK) to the sender and marks at its end that it has received packet with so and so sequence number. If this ACK is received uncorrupted to the sender before the timer stops, then this information is sufficient for the sender to understand that packet has arrived safe at the destination. If the ACK is not received or received in corrupted format, then the sender re-sends the packet and restarts the timer. Now when this packet arrives at the receiver's end, then there is problem of duplicate packets. Since the receiver marks at his end that he has received the packet in the first attempt, it silently discards the same packet sent a second time.

Issue of Out of Order Delivery:
Depending on the protocol type, out of order delivery can be handled in different ways. In Go-Back N protocol, if out of order packet arrives then it silently discards the packet and repeatedly sends the acknowledgement for the in-order packet. In Selective Repeat Protocol, if out of order packet arrives then it stores the packet at its end and sends the corresponding acknowledgement for it.

Example of how TCP recovers from lost or out-of-order packets: Packets 0, 1, 2, and 3 are sent. However, packet 1 is lost. The receiver receives packets 2 and 3, but they are discarded because they are received out of order (packet 1 is expected). When the receiver receives packets 2 and 3, it sends ACK1 to show that it expects to receive packet 1. However, these ACKs are not useful for the sender because the ackNo is equal to Sf, not greater that Sf. So, the sender discards them. When the time-out occurs, the sender resends packets 1, 2, and 3, which are acknowledged.


Sender
Transport layer
Receiver
Transport layer
Initial
Start timer
Stop
Start
Time out
Restart
Restart
Stop timer
Req
Packet 0
ACK 1
pArr
aArr
Packet 1
Lost
Packet 2
Packet discarded
Packet 3
ACK discarded
Packet 1 (resent)
Packet 2 (resent)
Packet 3 (resent)
ACK 2
ACK 3
ACK 4
Events:
Req: Request from process
pArr: Packet arrival
aArr: ACK arrival
time-out: Timer expiration
Time
Time