Transport Layer

Dr. Keyur Parmar

S. V. National Institute of Technology (NIT), Surat

Outline

Transport Layer

Goals:

- Extending the network layer delivery service - The network layer facilitates communication between hosts, while the transport layer facilitates communication between processes on different hosts.
- Error checking

Which computational problems are solved by transport layer?

How to faciliate reliable data transfer between processes running on different hosts?

How can we avoid data loss and data corruption?

How to control the transmission rate?

How can we avoid and recover from network congestion?

Transport Layer Services

Transport-layer Protocols

- Between application layer and network layer.
- Extend the service of network layer.
- Available in hosts (and not in routers or switches).
- Encapsulate application layer “messages” inside transport layer “segments” and vice versa.

Transport-layer Protocols

- UDP (User Datagram Protocol)
  - Unreliable and connectionless
- TCP (Transmission Control Protocol)
  - Reliable and Connection-oriented

Network-layer Protocol

- IP - Internet Protocol
- Each host has a network layer address, IP address.
- Provide communication between hosts.
- The best-effort delivery service i.e., unreliable service.

Multiplexing and Demultiplexing

Multiplexing and Demultiplexing

- Multiplexing – When the transport layer receives the data from the sockets, appends the transport layer headers, and forwards the data to the network layer.
- Demultiplexing – When the transport layer receives the data from the network layer and forwards it to the corresponding socket.

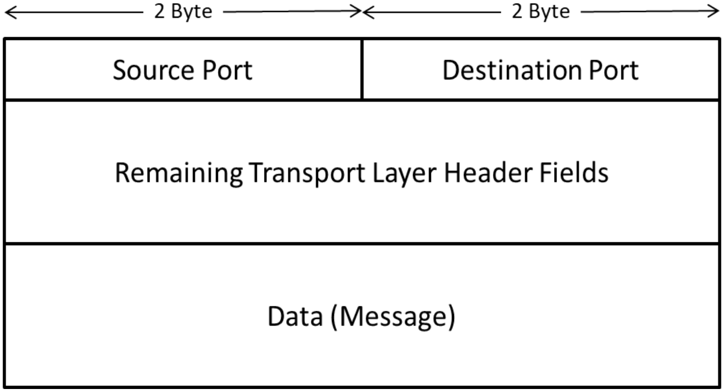
Transport Layer - Multiplexing and Demultiplexing

- A socket must have a unique identifier, port number.
- Each transport layer segment must have fields that identify the socket to which the segment must be delivered.

Port Numbers

- 16-bit number (Range: 0 to 65535)
- Reserved port numbers of servers - 0 to 1023
  - HTTP – 80
  - DNS – 53
  - SMTP – 25
- Port numbers assigned to the newly created sockets (un-reserved) of clients – 1024 to 65535.

Transport Layer Segment



Connectionless Multiplexing and Demultiplexing

- To identify a UDP socket
  - Destination IP address
  - Destination port number
- If two segments have different source IP addresses or port numbers, but the same destination IP address and the port number, then both the packets will be forwarded to the same socket.

Connection-Oriented Multiplexing and Demultiplexing

- To identify a TCP socket
    - Source IP address
    - Source port number
    - Destination IP address
    - Destination port
- If two segments have different source IP addresses or port numbers, but the same destination IP address and the port number, then both the packets will be forwarded to the different sockets.

Security – Port Scanning

- Scanning the ports
  - If the port is open, and if there is a known vulnerability related to the corresponding application, it can be exploited by the malicious adversary.
- Port Scanner
  - Nmap (`http://nmap.org`)

User Datagram Protocol (UDP)

Transport Layer Protocol - UDP

- User Datagram Protocol (UDP)
    - Defined in RFC 768.
    - Connectionless protocol (No handshaking)
    - Services
        - Multiplexing/Demultiplexing
        - Error checking
    - Example - DNS

Why UDP?

- Gives more control to applications, i.e., UDP immediately pass the data to the network layer without any congestion control .
- Real time applications – Delayed segment delivery may not be preferable. Such applications sometimes are tolerant to data loss.
- Reduces repetition if service is provided by other layers of the Internet protocol stack.

Why UDP?

- UDP does not maintain a connection state. Hence, no need to buffer data, store acknowledgment number, sequence number, etc. Supports multiple active connections on the server.
- Packet header - Requires only 8 bytes as compared to TCP which requires 20 bytes. Preferred to carry network management (reduce traffic when the network is already congested).

UDP Segment Structure

- UDP header
  - Length of UDP segment (2 bytes)
  - Checksum (2 bytes)
    - To determine if the bits in the segment have been altered en route.
    - Error checking (detection only, but no recovery. Erroneous segments are discarded with/without acknowledgment to the corresponding application).

How to Generate UDP Checksum

- Client – Generate a checksum.
  - Fragment segment into 16-bit chunks.
  - Add the 16-bit chunks.
  - Overflow – Wrap around.
  - Calculate 1's complement.

How to Verify UDP Checksum

- Server – Verifies the checksum.
    - Fragment segment into 16-bit chunks.
    - Add the 16-bit chunks and the checksum.
    - If answer is $11111111\ 11111111$, it represents the UDP segment without errors.
    - If one of the bits is 0 in the checksum, it represents error(s) in the segment.

How to Verify UDP Checksum

- UDP Segment

  0110011001100000
  0101010101010101
  1000111100001100

- Add all the chunks (wrap around if overflow).

  01001010 11000010

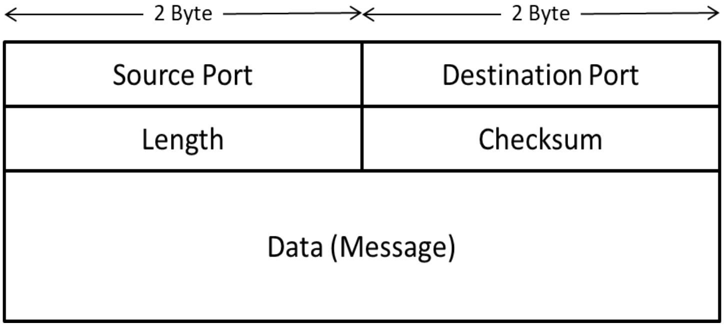
- Calculate 1's complement (Convert 0 to 1 and 1 to 0).

  10110101 00111101 – A checksum..!

Error Checking - Link layer and Transport layer

- Link layer – If one of the links in the path does not provide error checking, it will not ensure the end-to-end error checking.
- Link layer – Checks only those errors that are introduced during communication. Does not check errors introduced at intermediate nodes.
- Transport layer – Provides end-to-end (form source to destination) error checking.

UDP Segment


2 Byte
2 Byte
Source Port
Destination Port
Length
Checksum
Data (Message)

Reliable Data Transfer Problem

What is reliable data transfer?

How to implement a reliable data transfer between communicating processes of different hosts?

Reliable Data Transfer Problem

- Reliable Data Transfer between processes
  - Without data loss
  - Without data corruption
  - Without the modification of data sequence

How to provide reliable data transfer?

Use reliable data transfer protocols such as TCP.

Reliable Data Transfer Problem

- Reliable Data Transfer between processes
  - Without data loss
  - Without data corruption
  - Without the modification of data sequence
- Let's start with only the first problem.

What do you do to communicate reliably over an unreliable communication channel? E.g. calling someone..?

Reliable Data Transfer Problem

- Send data packets
- Send control packets

Let’s build a reliable data transfer protocol.

For simplicity, let's assume the unidirectional data transfer.

RDT 1.0

- Let’s assume that the communication channel is reliable.
- As the channel is reliable, all data communicated through the reliable channel will reach the destination reliably.

RDT 1.0 - State information

- Sender
  1. Waiting to send the packet.
- Receiver
  1. Waiting to receive the packet.

RDT 2.0

- Let's assume that the communication channel is not reliable, and bits in the packet may be corrupted.

How do you handle this scenario when talking to someone using a phone?

Acknowledgment Messages

- Acknowledgment Messages
    1. Positive acknowledgment, e.g., okay.
    2. Negative acknowledgment, e.g., pardon.
- Reliable data transfer protocols based on such retransmission are known as **Automatic Repeat ReQuest Protocols (ARQ)**.

How to handle the bit errors in a computer network?

RDT 2.0

- Error detection
  - How does the receiver detect errors? Needed a mechanism to detect the errors.
  - UDP – Checksum to detect errors.
  - Mechanisms used for error detection need senders to send extra bits (checksum) along with data.

RDT 2.0

- Receiver feedback
    - Receiver must send the feedback to the sender.
    - Positive acknowledgment (ACK) and negative acknowledgment (NAK)
    - How many bits are needed to send the feedback?

RDT 2.0

- Retransmission
    - Sender retransmits the packet for which it has received the NAK.

RDT 2.0 – State Information

- Sender
  - Waiting to send the data
  - Waiting for ACK or NAK
- Receiver
  - Waiting to receive the data

Stop-and-Wait Protocols

- Sender
    - Waits for the positive acknowledgment before sending the new data packets to the receiver.
    - Retransmits the same data packets, if the negative acknowledgment is received.

What if the ACK or NAK packet is erroneous (or corrupt)?

ACK or NAK Packets – Bit Errors

- Add checksum to ACK/NAK Packets.
  - To detect errors in the ACK/NAK packets.
- How do we recover from the bit errors of ACK/NAK packets?
  - Retransmit the data packets.

What is the problem with retransmission of data packets?

Retransmission of Data Packets

- Duplicate packets...!

If the ACK/NAK packets are erroneous, and receiver already has received the data packet correctly, then how does the receiver identify whether the incoming data packet is a new data packet or a retransmitted packet?

Duplicate Packets

- Number the data packets – Sequence number..!
- Send not only data but also the sequence number.
- A new protocol – RDT 2.1.
- For the stop-and-wait protocol (RDT 2.0), a 1-bit sequence number will be sufficient.

RDT 2.1 – State Information

- Sender
  - Waiting to send the data - 0
  - Waiting for ACK - 0 or NAK – 0
  - Waiting to send the data - 1
  - Waiting for ACK - 1 or NAK – 1

RDT 2.1 – State Information

- Receiver
    - Waiting to receive the data 0
    - Waiting to receive the data 1

Can we eliminate the NAK packets? How?

How to Eliminate NAK Packets

- Receiver
  - Send ACK packets.
  - If the sender receives the same ACK packet twice, it knows that the newly forwarded packet has not been received correctly by the receiver.

How do we identify that the ACK packet is same as the previously received ACK packet or a new ACK packet?

How to Eliminate NAK Packets

- Receiver
    - Add sequence number in ACK packets.
    - A new protocol – RDT 2.2.

How to detect packet loss?

How to recover from packet loss?

How to recover from a packet loss?

How to Recover From Packet Loss

- Checksum
- Sequence numbers
- ACK packets
- Retransmission

How to detect a packet loss?

How to Detect Packet Loss

- When the data packet or ACK packet will be lost, the sender will not receive the ACK packet.
- If there is a packet loss, ACK loss, or ACK overly delayed,
    - Sender retransmits the data packet.
- How long should the sender wait for an ACK packet?
    - Round Trip Time (RTT)
    - Packet processing time at the receiver

Ideally, we should recover from the packet loss as soon as possible.

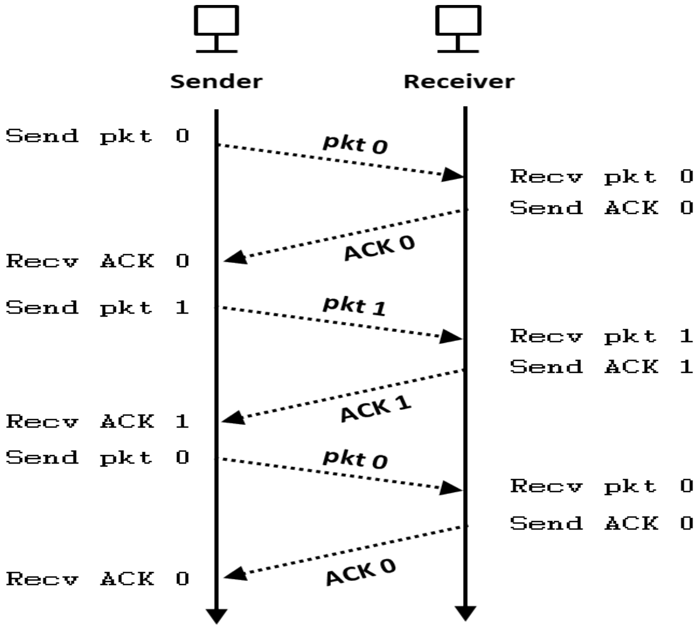
RDT 3.0

- Implement a countdown timer.
- If there is a packet loss, ACK loss, or overly delayed ACK, then the countdown timer will interrupt the sender for re-transmission.

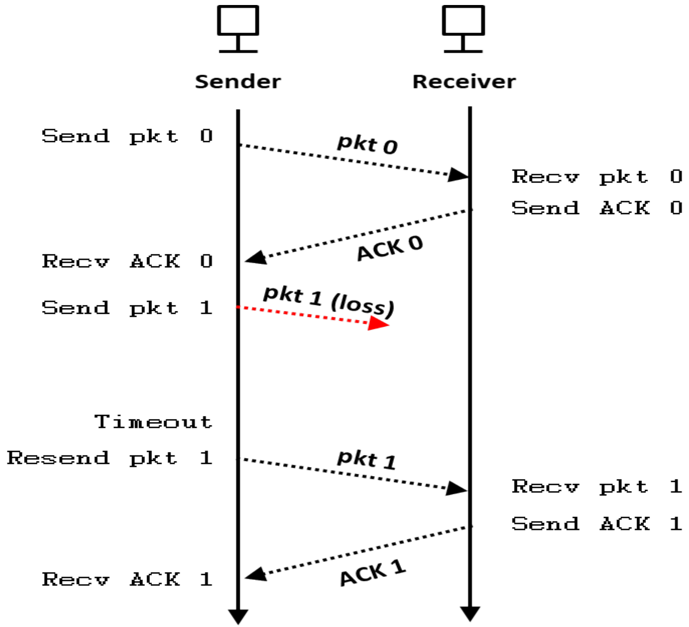
RDT 3.0

- Sender
  - Starts the timer when transmits the packet.
  - Retransmits the packet if interrupted by the timer.
  - Stops the timer when the ACK is received.

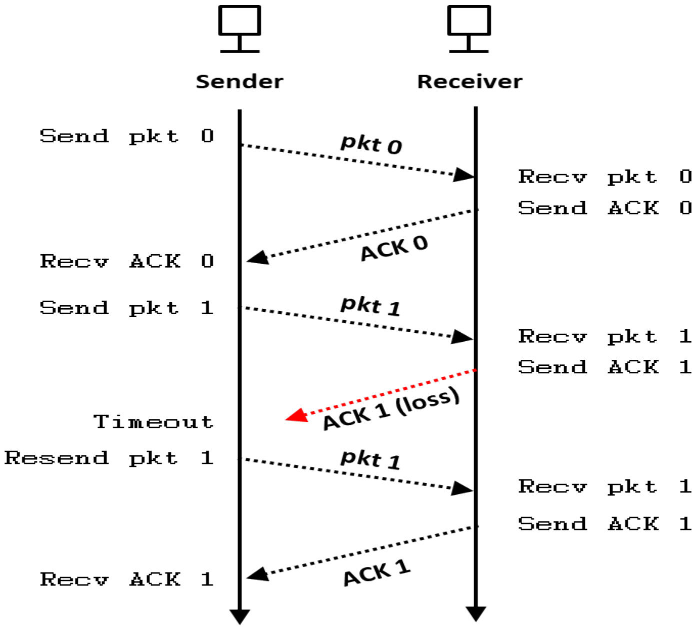
RDT 3.0


Sender
Receiver
Send pkt 0
pkt 0
Recv pkt 0
Send ACK 0
ACK 0
Recv ACK 0
Send pkt 1
pkt 1
Recv pkt 1
Send ACK 1
ACK 1
Recv ACK 1
Send pkt 0
pkt 0
Recv pkt 0
Send ACK 0
ACK 0
Recv ACK 0

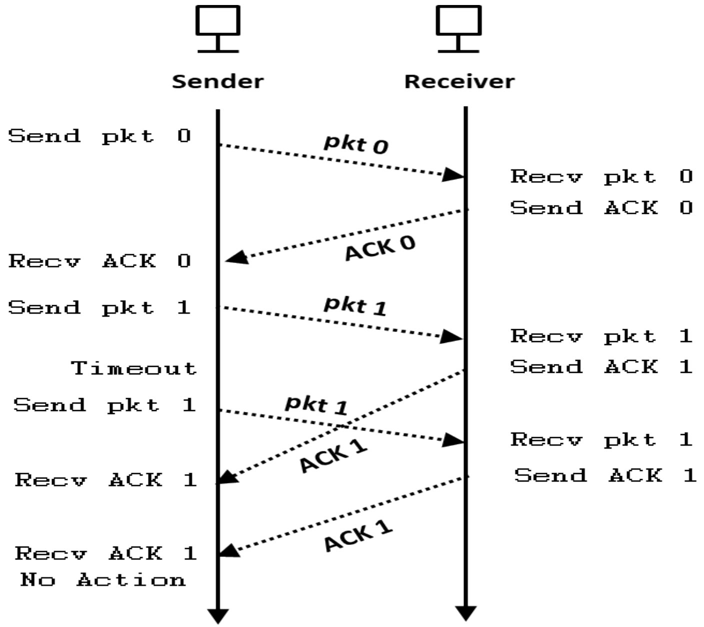
RDT 3.0


Sender
Receiver
Send pkt 0
pkt 0
Recv pkt 0
Send ACK 0
ACK 0
Recv ACK 0
Send pkt 1
pkt 1 (loss)
Timeout
Resend pkt 1
pkt 1
Recv pkt 1
Send ACK 1
ACK 1
Recv ACK 1

RDT 3.0


Sender
Receiver
Send pkt 0
pkt 0
Recv pkt 0
Send ACK 0
ACK 0
Recv ACK 0
Send pkt 1
pkt 1
Recv pkt 1
Send ACK 1
ACK 1 (loss)
Timeout
Resend pkt 1
pkt 1
Recv pkt 1
Send ACK 1
ACK 1
Recv ACK 1

RDT 3.0



Sender
Receiver
Send pkt 0
pkt 0
Recv pkt 0
Send ACK 0
ACK 0
Recv ACK 0
Send pkt 1
pkt 1
Recv pkt 1
Send ACK 1
Timeout
Send pkt 1
pkt 1
Recv pkt 1
ACK 1
Recv ACK 1
Send ACK 1
ACK 1
Recv ACK 1
No Action

Stop-and-Wait Protocol

- Performance
  - Delay
  - Very less effective throughput (unutilized communication channel)

Can we improve the performance of networking protocols that follow the stop-and-wait mechanism?

Pipelining

- Send multiple packets without waiting for acknowledgments.

How to modify RDT 3.0 to support pipelining?

RDT 3.0 and Pipelining

- Increase sequence number range
- Sender - Buffer the packets that are not acknowledged.
- Receiver – Buffer the packets. (discussed next..)
- The range of sequence numbers and buffer requirements depend on the way we handle data loss, data corruption, and packet delays.

Go-Back-N Protocol (Sliding Window Protocol)

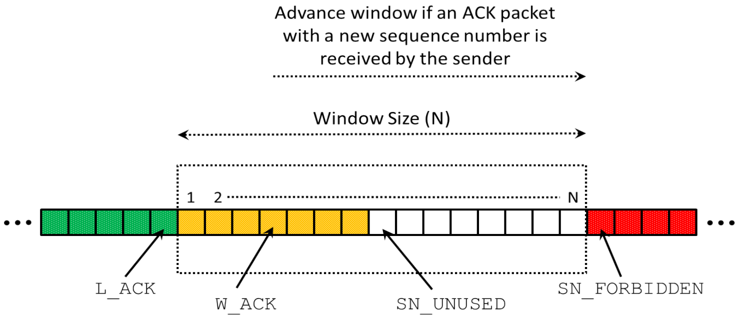
- Sender is allowed to send a threshold number (N) of packets without waiting for acknowledgments.
- GBN Animation

Why do we restrict the number of packets allowed to be sent by the sender?

Go-Back-N Protocol – Packet Restriction

- Flow control
- Congestion control

Sliding Window - Sender


Advance window if an ACK packet with a new sequence number is received by the sender
Window Size (N)
1
2
N
L_ACK
W_ACK
SN_UNUSED
SN_FORBIDDEN

Sliding Window - Sender

- `L_ACK` – Packets transmitted, and ACK received.
- `W_ACK` – Packets transmitted, but ACK not received.
- `SN_UNUSED` – New packets can use the available unused sequence numbers.
- `SN_FORBIDDEN` – Sequence numbers forbidden to be used unless new packet(s) are acknowledged.

Sliding Window Protocol

- Sequence number – Fixed length field in packet header.
- For example, k bit field, supports sequence numbers in the range 0 to $2^k - 1$.
- Consider the sequence numbers arranged in a circle (modular arithmetic), and after the last sequence number, start with 0 again.

Sliding Window Protocol

- Sender received a new packet (from Application Layer)
    - If all sequence numbers are already assigned (`SN_UNUSED` is not available), wait for new packet acknowledgment.
    - If `SN_UNUSED` is available, add the unused sequence number for the packet.

Sliding Window Protocol

- Sender received the acknowledgment
    - Cumulative acknowledgment – An ACK packet with a sequence number s indicates that all packets before the sequence number s have been successfully received by the receiver.

Sliding Window Protocol - Sender

- Timeout – When sender receives no acknowledgment
    - Go-back-N - Retransmits all packets sent after the packet N (the last acknowledged packet).
- A single timer
    - Timer of an oldest unacknowledged packet.
    - Restarts once the new ACK packet is received.
    - Stops if the sender is not waiting for any other ACK.

Sliding Window Protocol - Receiver

- If the last acknowledged packet is having the sequence number n,
  - and, if the packet with sequence number $n + 1$ is received, send the ACK for the packet having sequence number $n + 1$.
  - then, except the packet with sequence number $n + 1$, discard all other incoming packets, and send the ACK (of packet having sequence number n).

Sliding Window Protocol

- Sender maintains
    - `L_ACK`
    - `W_ACK`
    - `SN_UNUSED`
    - `SN_FORBIDDEN`
- Receiver maintains
    - `W_PKT` – A sequence number of the expected packet.

What is the problem with sliding window protocol?

When the window size is large and/or packet delay is also large, a single packet error can cause the sliding window protocol to retransmit the large number of packets. Can we reduce the number of packets required to be retransmitted?

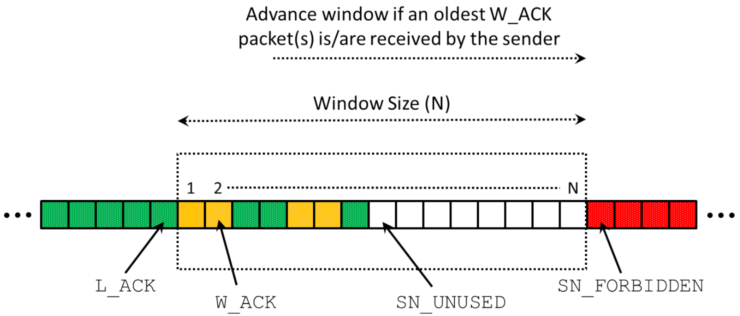
Selective Repeat (SR)

- Goal: To avoid unnecessary retransmissions of packets.
- Receiver – Acknowledge the correctly received packet irrespective of the packet order.
- Out of order packets (received) are stored in buffered while waiting for the missing packets.

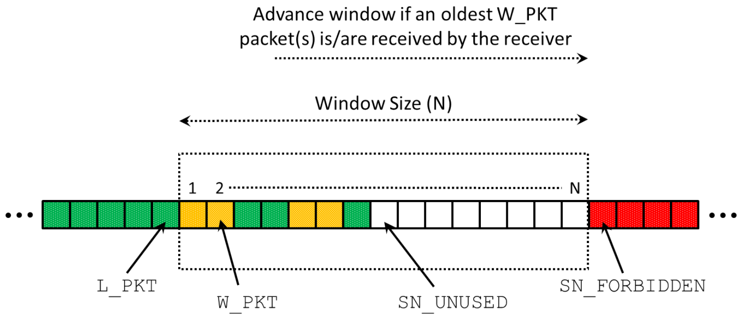
Selective Repeat (SR)

- Instead of a single timer as in Go-back-N protocol, different timers are used for different packets.
- Sender and receiver maintain the separate windows.
- Animation – SR Protocol

Selective Repeat (SR) – Sender Window

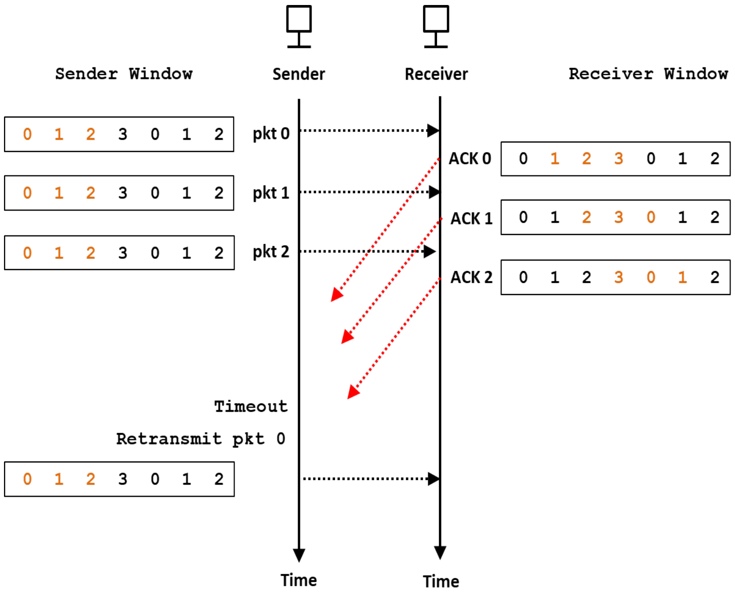

Advance window if an oldest W_ACK packet(s) is/are received by the sender
Window Size (N)
1
2
N
L_ACK
W_ACK
SN_UNUSED
SN_FORBIDDEN

Selective Repeat (SR) – Receiver Window

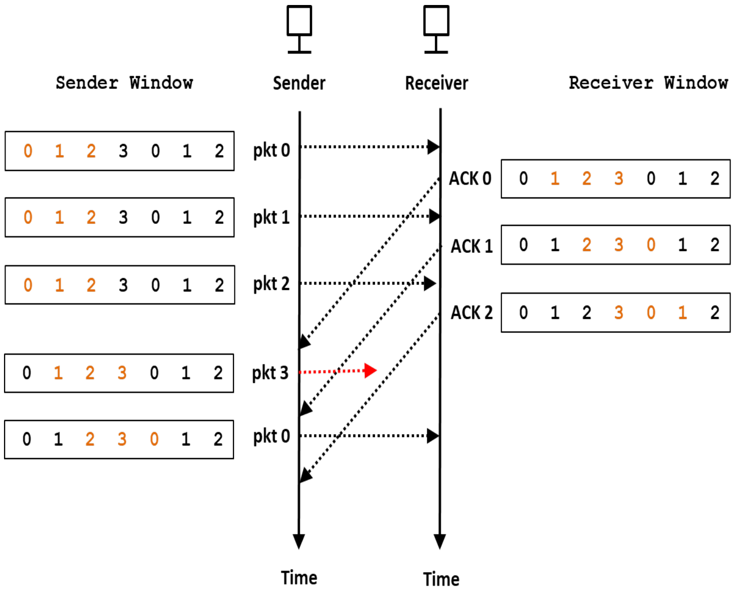

Advance window if an oldest W_PKT packet(s) is/are received by the receiver
Window Size (N)
1
2
N
L_PKT
W_PKT
SN_UNUSED
SN_FORBIDDEN

Are there any synchronization between the sender's window and the receiver's window?

Selective Repeat (SR) – Scenario 1



Selective Repeat (SR) – Scenario 2



Can the receiver distinguish between the retransmission of the first packet PKT 0 (Fig 1) and the transmission of the fifth packet PKT 0 (Fig 2)? What can we do to solve this problem?

Window size must be ___________ the size of total sequence numbers to avoid synchronization issues.

How do we handle the packets or ACKs that are received out of order? For example, assume that a very old packet may be received by the receiver having the sequence number same as one of the sequence numbers as in the window.

Packets or ACKs – Received Not as Per Order

- Avoid reusing the same sequence number until we are sure that the packet or ACK are not in the network.
- Set the expiry time of the packet in a network.
- Maximum packet lifetime in TCP – Approximately 3 Minutes.

Conclusions

Assignments

Assignment

References

References

- James Kurose and Keith Ross, Computer Networking: A Top-Down Approach - Textbook
- James Kurose, Computer Networking: A Top-Down Approach - Video Lectures
- Textbook Resources
- Internet Engineering Task Force

Thank You.