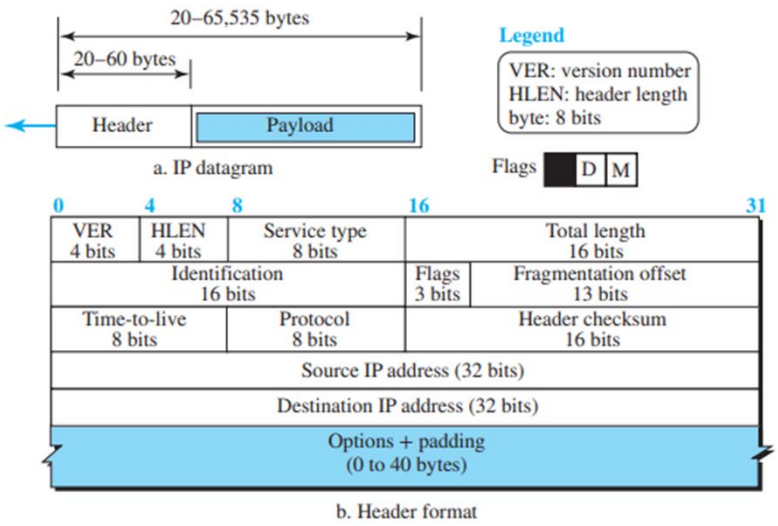
Fundamentals of Computer Networks

Understand computer network technology

Lesson 4

Network layer – IP (recall)



Field	Size (bits)	Description
Version (VER)	4	Indicates the IP version. For IPv4, this is always **4**.
Header Length (HLEN)	4	Specifies the length of the header in 32-bit words. Minimum value is 5 (20 bytes).
Service Type	8	Originally used to indicate priority. Now used as **DSCP** for Quality of Service (QoS).
Total Length	16	Length of the entire packet (header + data), in bytes. Maximum is 65,535.
Identification	16	A unique value used for identifying fragments of the same original packet.
Flags	3	Controls fragmentation. Includes **DF (Don't Fragment)** and **MF (More Fragments)** flags.
Fragmentation Offset	13	Used to reassemble fragmented packets in the correct order.
Time to Live (TTL)	8	Limits the packet's lifespan. Each router decrements it; packet is dropped when TTL reaches 0.
Protocol	8	Specifies the next-level protocol (e.g., **1 = ICMP**, **6 = TCP**, **17 = UDP**).
Header Checksum	16	Error-checking of the IPv4 header only (not the payload).
Source IP Address	32	The IPv4 address of the sender.
Destination IP Address	32	The IPv4 address of the intended recipient.
Options (optional)	Variable	Additional control information; rarely used. Header length increases if present.
Padding	Variable	Extra bits added to make header length a multiple of 32 bits.

https://datatracker.ietf.org/doc/html/rfc791

*** RFC stands for Request for Comments, a series of documents that are the foundation of Internet standards and protocols.**

Checksum

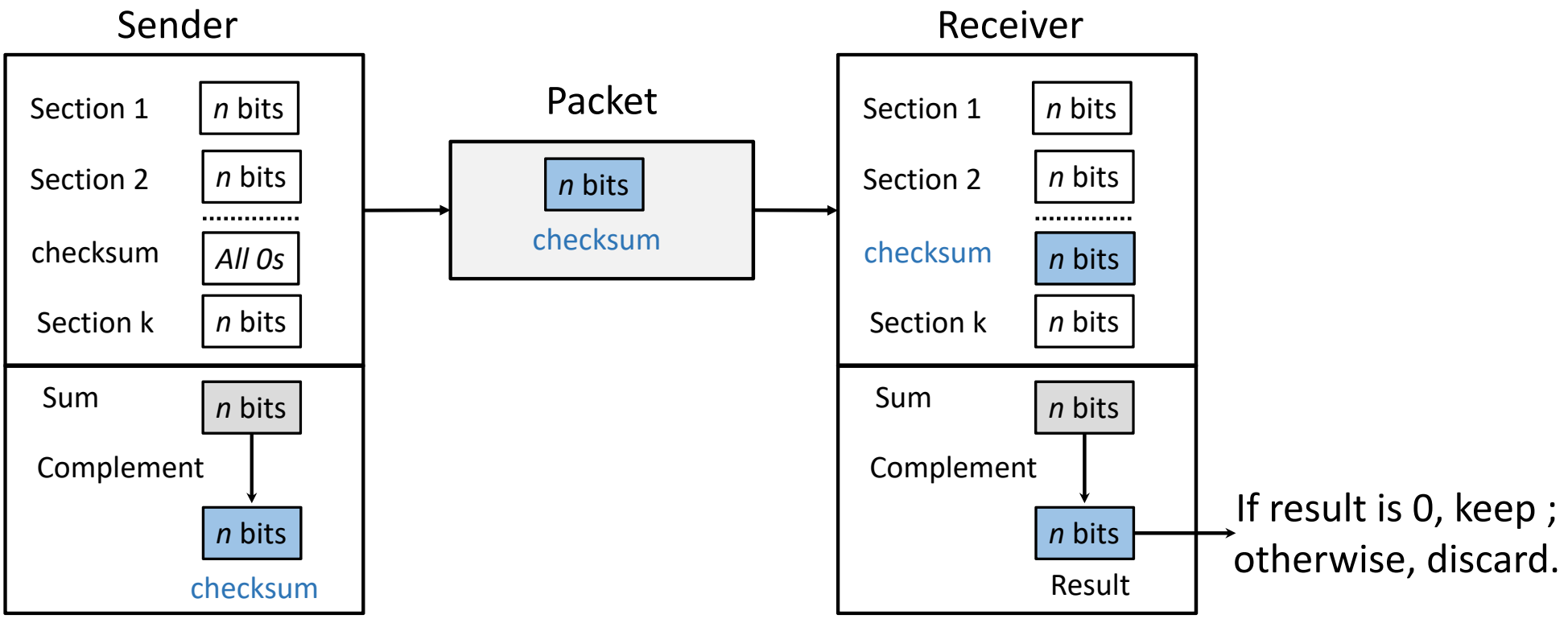
Checksum

- Error detection method used by most TCP/IP protocols
- The checksum protects against the corruption that may occur during the transmission of a packet. It is a redundant information added to packet.
- It is calculated at the sender and the value obtained is sent with the packet. The receiver repeats the same calculation on the whole packet including the checksum. If the result is correct, the packet is accepted, otherwise it is rejected.

Checksum

- Checksum calculation at the sender:
  - Packet is divided into k sections of n bits each.
  - All sections are added together using one's complement arithmetic
  - Final result is complemented to make the checksum
- Checksum calculation at the receiver:
  - Received packet is divided into k sections of n bits each.
  - All sections are added together using one's complement arithmetic
  - Result is complemented.
  - If final result is 0, the packet is accepted, otherwise rejected.

Checksum


Sender
Section 1
n bits
Section 2
n bits
checksum
All 0s
Section k
n bits
Sum
n bits
Complement
n bits
checksum
Packet
n bits
checksum
Receiver
Section 1
n bits
Section 2
n bits
checksum
n bits
Section k
n bits
Sum
n bits
Complement
n bits
Result
If result is 0, keep ;
otherwise, discard.

Checksum – binary addition

A	B	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1 (carry)

Example:

```
          1101
          1011
Result:  11000     (starts from the LSB on the right)
```

Checksum – one's complement arithmetic

- Similarly to normal binary addition except when there's a carry out of the most significant bit (MSB), the carry is wrapped around and added back into the least significant bit(LSB).

Example:

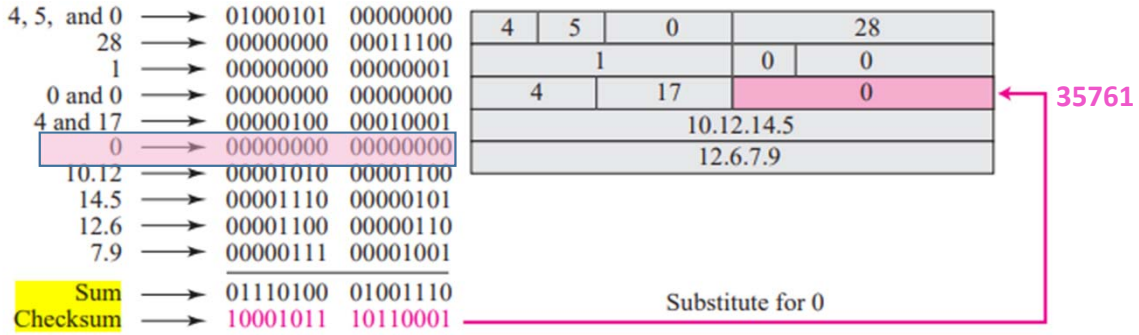
```
        Word 1:   1111 1111 1111 1111 (0xFFFF)
+       Word 2:   0000 0000 0000 0001 (0x0001)
              1 0000 0000 0000 0000 (17 bits – overflowed 16 bits)
  Final result:   0000 0000 0000 0001 (drop carry bit and add to result)
```

Checksum – IP header

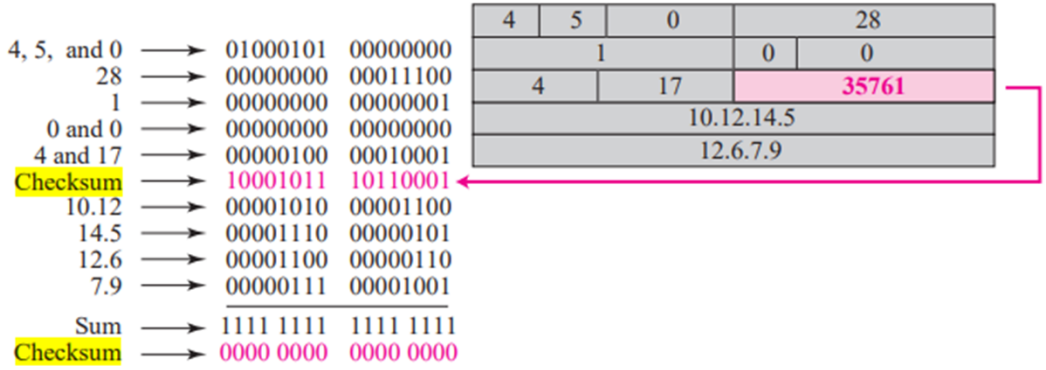
- Checksum in IP covers only the header, not the data.
- All higher-level protocols that encapsulate data in the IP datagram have a checksum fields that covers the whole packet. Therefore the checksum for IP datagram does not have to check the encapsulated data.
- The header of the IP packet changes with each visited router, but the data do not. So the checksum includes only the part that has changed.
- If data were included, each router would need to recalculate the checksum for the whole packet which will increase in processing time.

Checksum – IP header example

Checksum calculation at the sender



Checksum calculation at the receiver

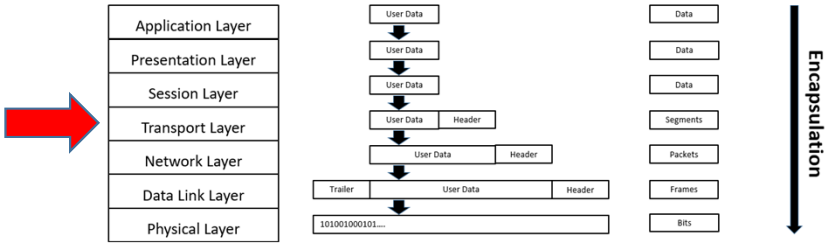


Transport Layer

Protocols

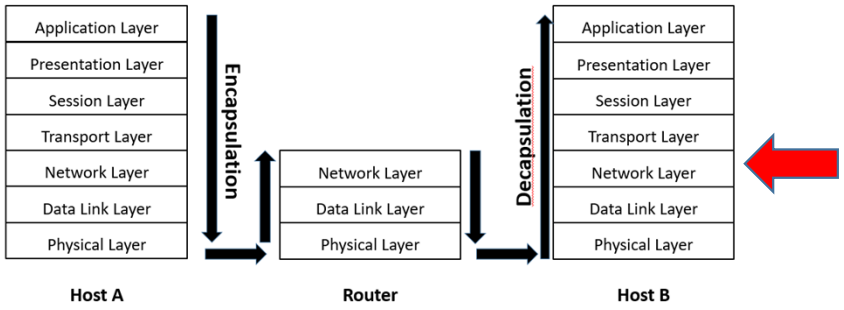
End-to-End Data Flow (Host A Source) (Recall)

- At the transport layer, the data is segmented and a header is added to each segment of the data.
- This header contains transport control information such as the sequence number and acknowledgement number.
- The segment along with its header is passed down to the network layer.
- The header added by the transport layer is meant to be read by the transport layer at the receiving end.



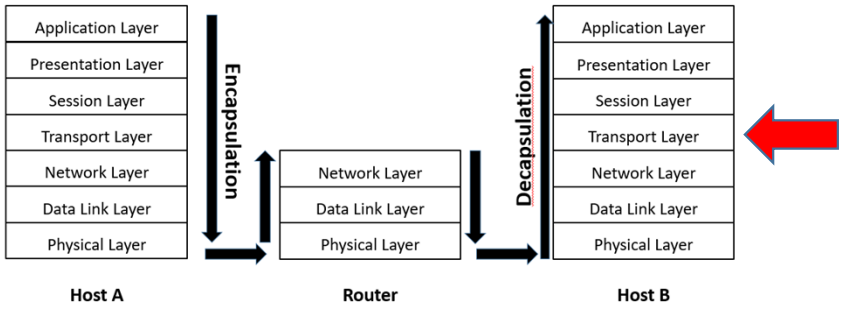
End-to-End Data Flow (Host B Dest) (Recall)

- The header of the packet is read to determine if this is the correct destination for the packet and other network control information is also taken from the network layer header. The header is then removed and the rest of the data is elevated to the transport layer as segments.



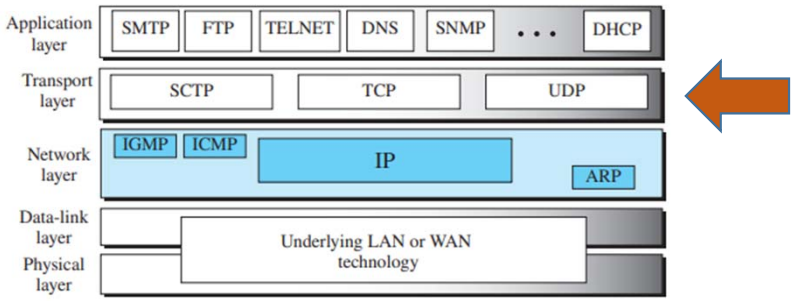
End-to-End Data Flow (Host B Dest) (Recall)

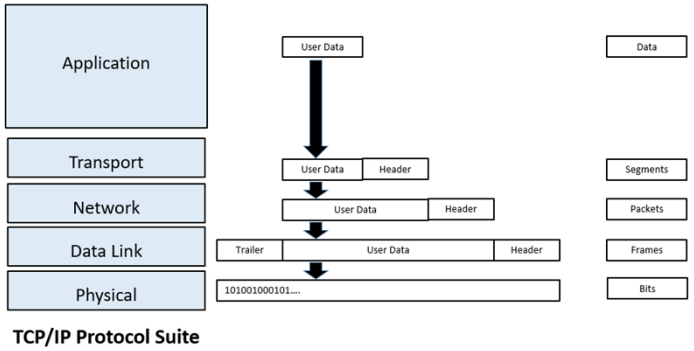
- The header of each segment is read to determine the sequence number and arrange the segments in their correct order. Then, the header is also removed and the rest of the data is elevated to the session layer.
- The transport header also contains information of which session this data is going to. This information is passed to the session layer with the data.



Transport layer (Recall)

- The transport layer in the TCP/IP model has similar purposes to that of the OSI model. It is designed to give the source and destination the ability to have end-to-end conversation.
- There are two defined protocols that can operate in this layer; Transmission Control Protocol (TCP) and User Datagram Protocol (UDP). These two protocols provide connection-oriented and connectionless communications.





Transport Layer

- The main purpose of transport layer is to ensure process-to-process delivery of the entire message between applications on different host.
- Service-Point Addressing (Port Addressing)
  - Each program (process) on a host has a unique port number
  - Transport layer uses port numbers to direct data to the correct application.
- Segmentation and Reassembly
  - Large messages are **split into smaller segments** for transmission.
  - Each segment is given a **sequence number**.
  - At the destination, the transport layer **reassembles** the segments in the correct order.
  - If segments are missing, they can be **identified and retransmitted**.

Transport Layer

- Connection Control – Connection-oriented (TCP)
  - Establishes a session before data transfer.
  - Ensures all data arrives in sequence.
  - Terminate connection after transfer.
- Connection Control – Connectionless (UDP)
  - No prior connection setup
  - Each packet is sent independently
- Flow Control
  - Prevents sender from overloading receive by regulating the rate of data transmission
- Error Control
  - Detects and corrects errors using acknowledgement and retransmission

Transport Layer - protocol

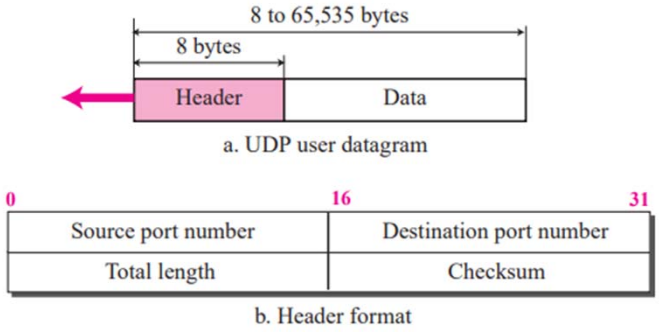
- User Datagram Protocol (UDP) -connectionless, unreliable communication
  - Faster and typically used for transferring small amount of data
- Transmission Control Protocol (TCP) – connection-oriented, reliable communication
  - Slower and used for transferring large amounts of data to ensure it is not sent again

Transport Layer – UDP

- User Datagram Protocol (UDP) is a connectionless, unreliable and lightweight transport-layer protocol. It is a simple protocol using minimum overhead.
- Used when speed is more important than reliability.
- Characteristics:
  - Connectionless – No handshake before communication starts
  - Unreliable – No guarantee of delivery or order
  - Fast and efficient – It has a minimal 8 byte header

Transport Layer – UDP

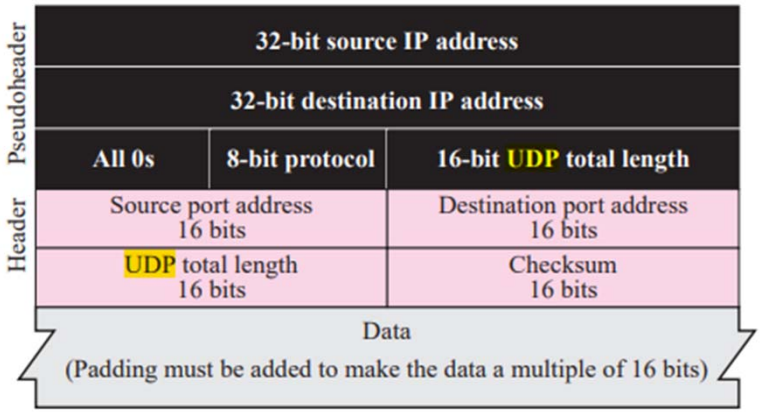
- UDP packets, called user datagrams, have a fixed-size header of 8 bytes.



Field	Size (byte)	Description
Source Port	2	Port of sending application
Destination Port	2	Port of receiving application
Length	2	Length of UDP header + data
Checksum	2	Used for error-checking of header and data

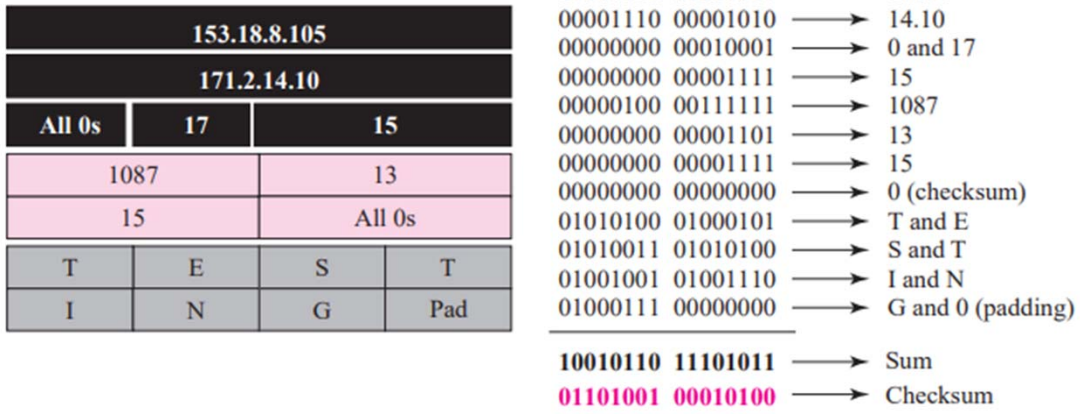
Checksum – UDP header

- UDP checksum calculation is different from IP.
- It includes 3 section: pseudoheader + UDP header + data
- Pseudoheader is part of the IP packet header



Checksum – UDP header example

Example of checksum calculation at the sender



Transport Layer – UDP

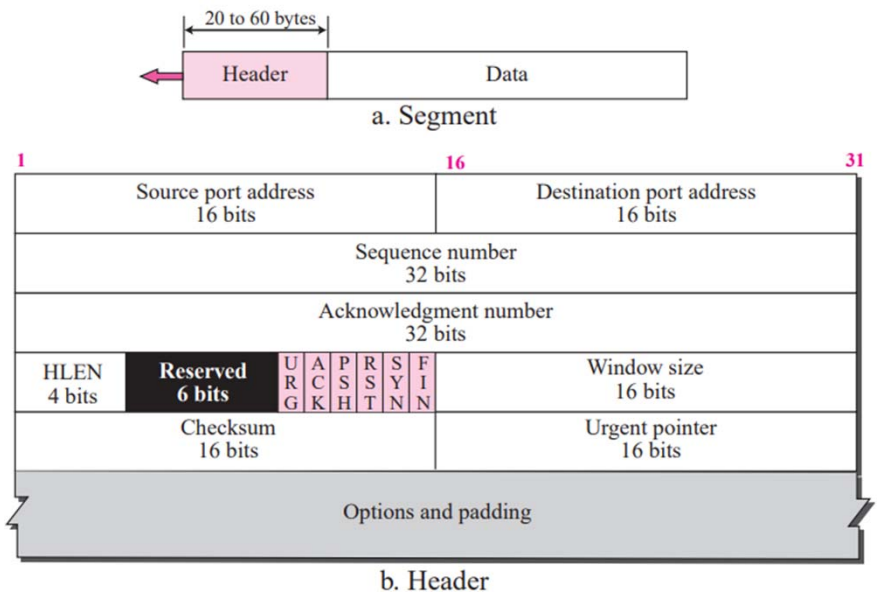
- Some Applications of UDP (will be in lesson 4)
- Real-time applications (fast , low latency).
  - VoIP (Voice over IP)
  - Online gaming
  - Streaming media
- Lightweight services
  - *Domain Name System (DNS), port 53*
  - *Dynamic Host Configuration Protocol (DHCP), port 67(server) /68 (client)*
  - Trivial File Transfer Protocol (TFTP), port:69
- Broadcast/Multicast
  - Routing Information Protocol (RIP), port 520
  - Multicast DNS (mDNS), port 5353

Transport Layer – TCP

- Transmission Control Protocol (TCP) is a connection-oriented protocol that provides reliable, ordered, and error-checked delivery of data between applications over a network.
- Characteristics:
  - Connection-oriented – before any data can be transmitted, a reliable connection must be obtained and acknowledged.
  - Reliable delivery – ensure data integrity (no loss or corruption). Uses acknowledgements and timeouts
  - Error checking – use checksum to detect error
  - Flow control – prevent overwhelming of the receiver
  - Congestion control – manage traffic load to avoid congestion
  - Ordered delivery – reassembled in correct order with sequence number

Transport Layer – TCP

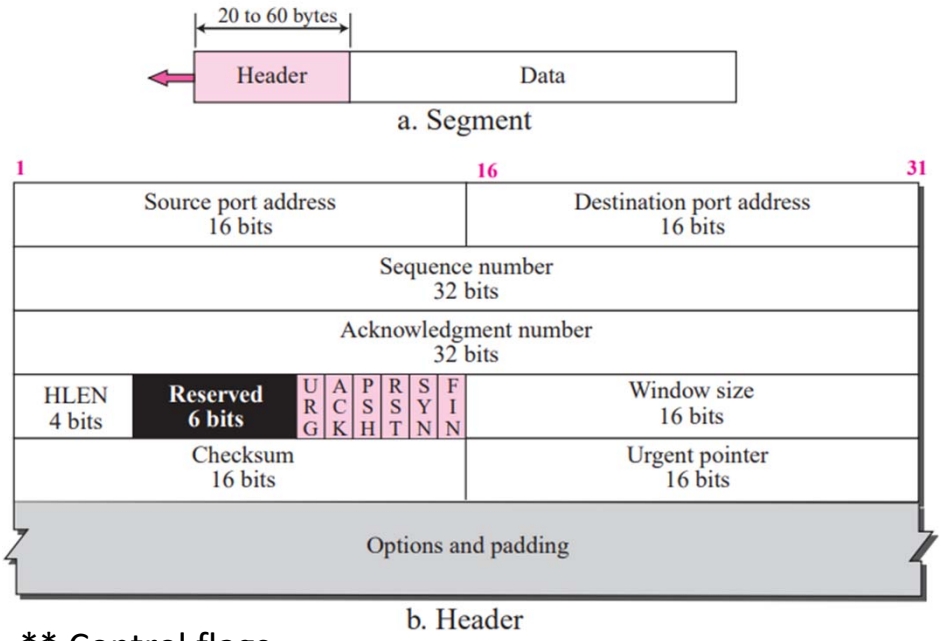
TCP packets (segments).



Field	Size (bits)	Description
Source Port	16	Port of sending application
Destination Port	16	Port of receiving application
Sequence Number	32	Number assigned to the first byte of data contained in this segment
Acknowledge Number	32	Number of the next data byte that the receiver expects to received from the sender.
Header Length	4	Number of 4-byte words in TCP header
Reserved	6	Reserved for future use

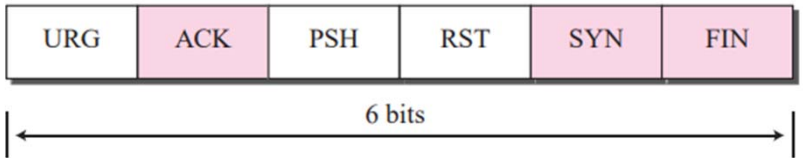
TCP sees user data as a stream of bytes.
Each byte in this stream has a number.

Transport Layer – TCP



** Control flags

URG: Urgent pointer is valid
ACK: Acknowledgment is valid
PSH: Request for push
RST: Reset the connection
SYN: Synchronize sequence numbers
FIN: Terminate the connection



Field	Size (bits)	Description
Control **	6	6 different control bits/flags. One or more of these bits can be set at a time.
Window size	16	Number of bytes the sender of this segment is willing to accept.
Checksum	16	Mandatory checksum of header + data
Urgent pointer	16	Offset number that must be added to the sequence number to obtain the number of the last urgent byte in the data section of the segment. Only valid if the urgent (URG) flag is set.
Options	40 x 8 bits	Optional information in TCP header

Transport Layer – TCP

- TCP connection management phases:
  - Connection establishment
  - Data transmission
  - Connection termination

Transport Layer - TCP

- Connection establishment – uses a three-way handshake process to establish connection. It ensures both sender and receiver agree on the connection state.
- Connection termination – uses a three-way handshake process or a 4 step termination process (half-close) to terminate the connection. It ensures that all data has been delivered and acknowledged before a connection is closed.

Transport Layer - TCP

- Mechanisms used to ensure reliable data transmission:
  - Sequence Numbers – every byte of data is assigned a sequence number. It allows the receiver to reorder segments and detect missing data.
  - Acknowledgements (ACK) – Receiver sends an ACK with the next expected byte number. It confirms that data was received correctly and in order.
  - Retransmission – Sender retransmits the data if it does not receive an ACK within a time.
    - TCP sets a retransmission timer for each sent segment. This ensures that lost ACKs or segments are detected and resent.
    - TCP has mechanism that allows faster recovery from packet loss by allowing sender to retransmit after receiving three duplicate ACK.

Transport Layer - TCP

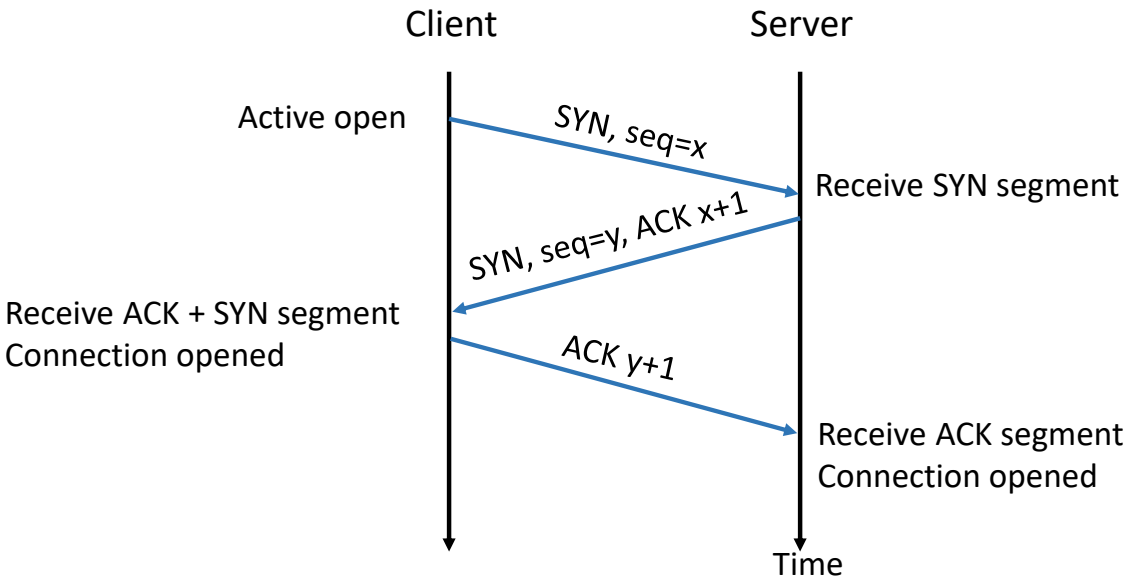
- Mechanisms used to ensure reliable data transmission:
  - Checksum –Each TCP segment includes a 16-bit checksum covering the header and data.
    - It detects corruption during transmission. If a segment is corrupted (invalid checksum), the segment is discarded by the destination TCP and is considered as lost.
  - Flow Control – Receiver advertises a window size to control how much data the sender can transmit. It prevents buffer overflow at the receiver.
  - Ordered delivery – Receiver stores out-of-order segments and waits to deliver them in sequence to the application. It ensures in-order delivery to the application layer.

TCP – Establish Connection

- Establish connection – TCP uses a three-way handshake process.
- Three messages are exchanged that allow each side to agree to form a connection and know that the other side has agreed
- Client sent a SYN segment for synchronization of sequence number – the number is called the initial sequence number (ISN).
- Client chooses a random number as the first sequence number.

TCP – Establish Connection

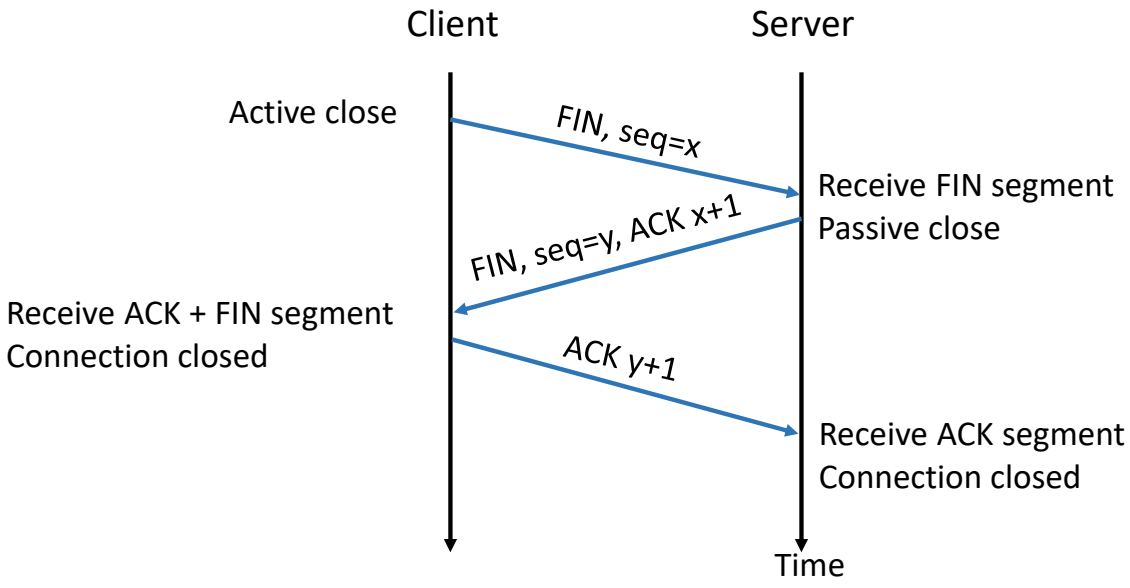
- Establish connection with a three-way handshake process.



1. **SYN**: Client sends synchronization request (seq no=x) to server.
2. **SYN+ACK**: Server acknowledges (seq no = x+1) and sends own synchronization (seq no = y)
3. **ACK**: Client acknowledges back to server (seq no=y+1)

TCP – Terminate Connection

- Terminate connection – TCP uses a three-way handshake process.



1. **FIN**: Client sends FIN segment (seq no=x) to server.

2. **FIN+ACK**: Server sends FIN+ACK segment to confirm the receipt of the FIN segment from the client and announce the closing of the connection. ACK (seq no = x+1) and FIN (seq no = y)

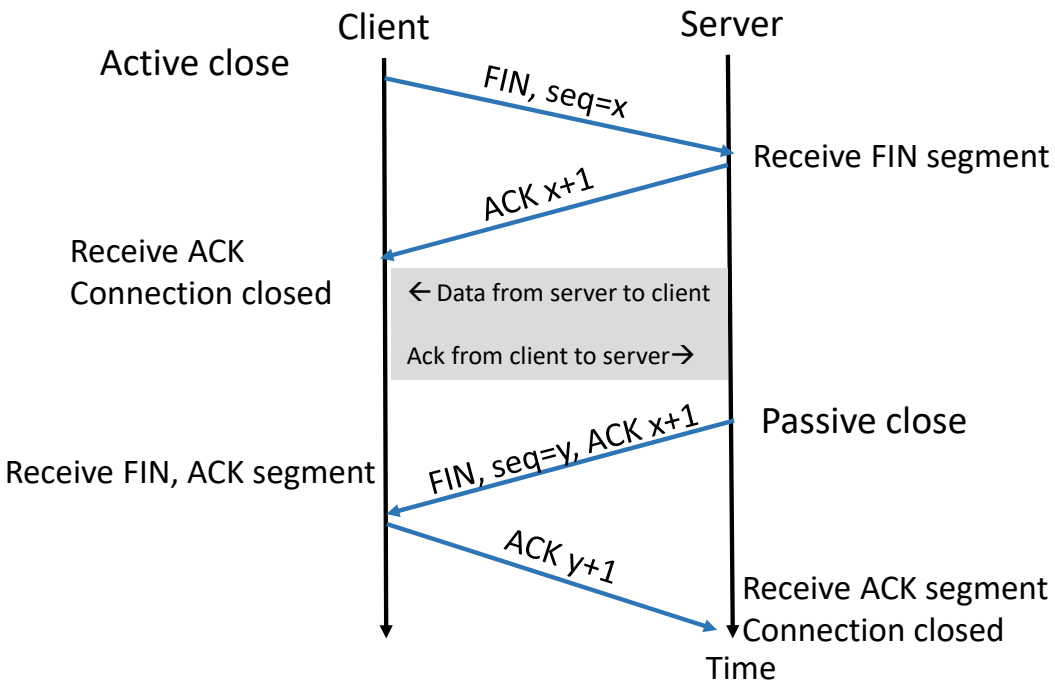
3. **ACK**: Client acknowledges the receipt of the FIN segment (seq no=y+1)

TCP – Terminate Connection

- Terminate connection – 4-step teardown process (half-close)
- This is known as a half-close request where one end can stop sending data while still receiving data.
- Either server or client can issue the request.
- Second FIN segment is not generated immediately

TCP – Terminate Connection

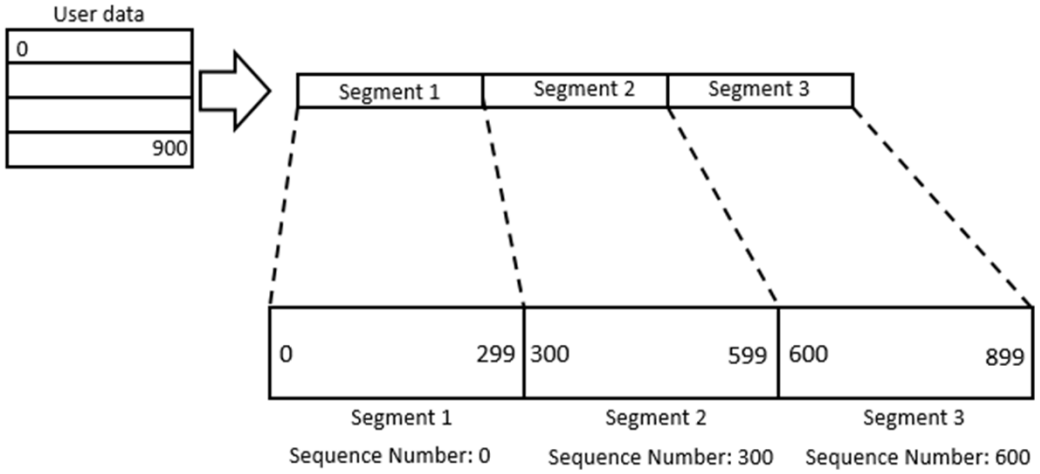
- Terminate connection with 4-step teardown process (half-close)



1. **FIN**: Client requests for connection closure (seq no =x)
2. **ACK**: Server acknowledges (seq no = x+1). Client cannot send any more data to server.
3. **FIN**: Server sends own FIN (seq no=y) with ACK (seq no=x+1)
4. **ACK**: Client acknowledges closure (seq no = y+1)

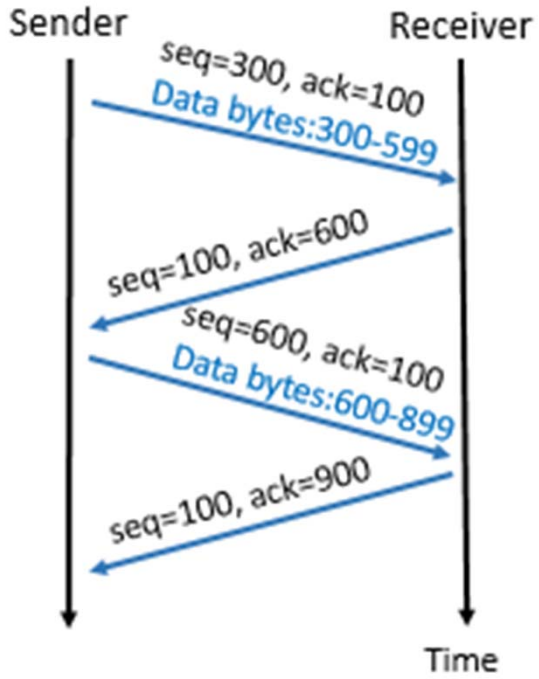
TCP – Sequence Number

- Each segment, from a large group of segments composing the user data, is assigned a sequence number.
- This number represents the sequence number of the first byte of user data carried in this particular segment.



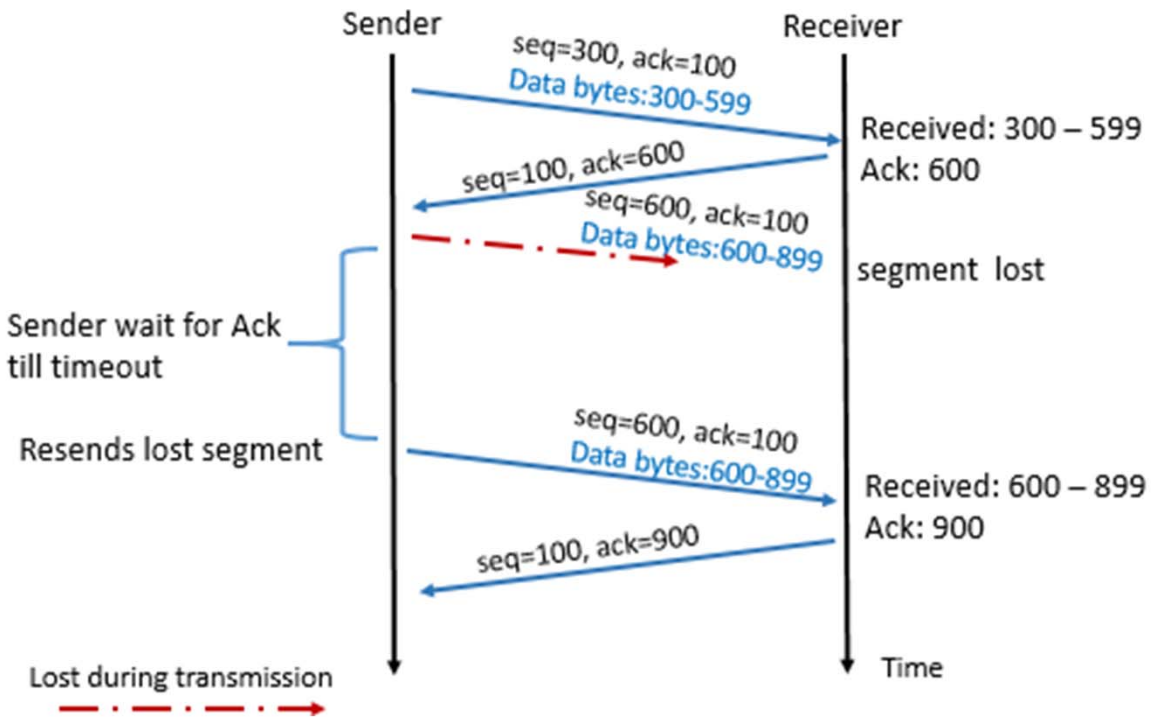
TCP – Acknowledge Number

- A simple example of the concept of sequence number and acknowledgement number
- shows that each side of the communication has its own sequence numbers for the data it wants to transfer.
- Sender sends the acknowledgement number that is the sequence number of the last byte received from receiver plus one.
- Receiver sends acknowledgement number that is the sequence number of the last byte received from sender plus one.



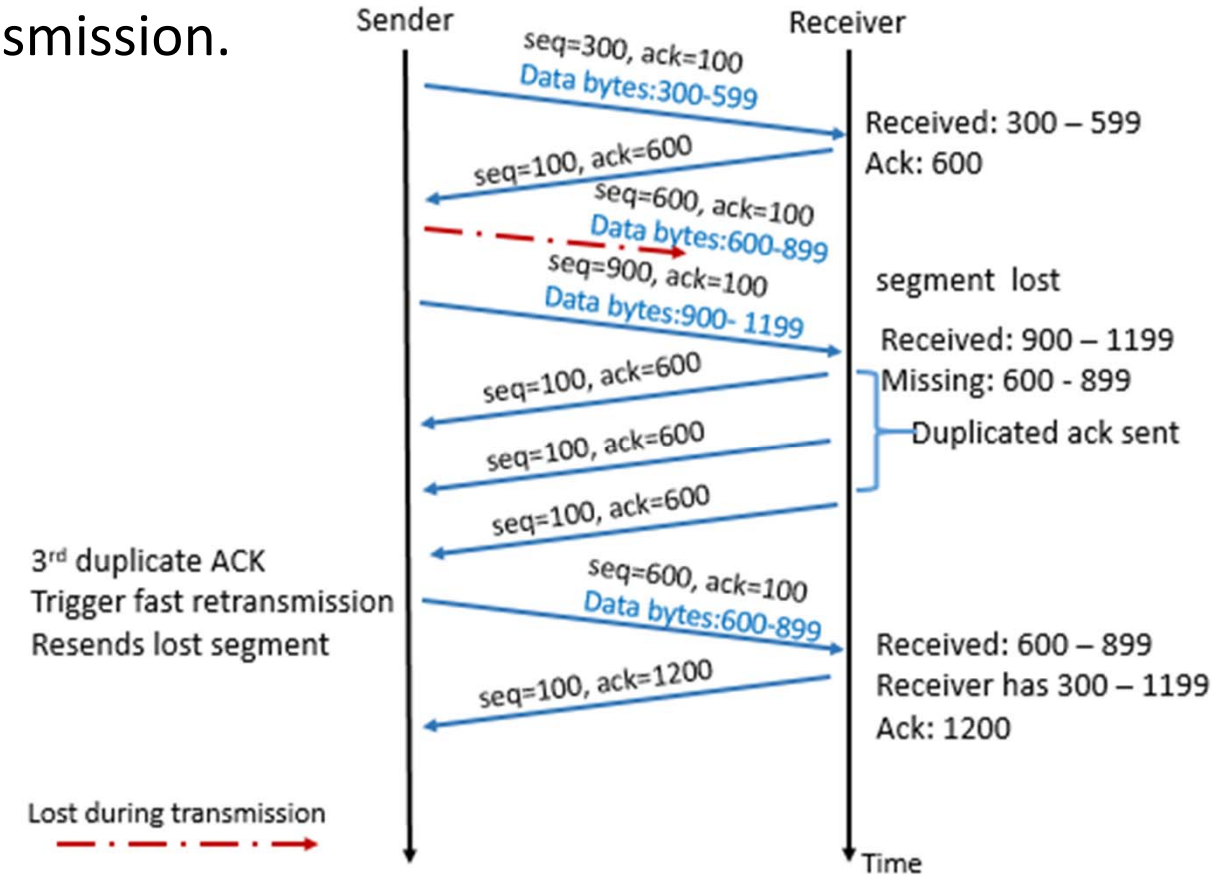
TCP – Retransmission

An example of lost segment and retransmission due to timeout.



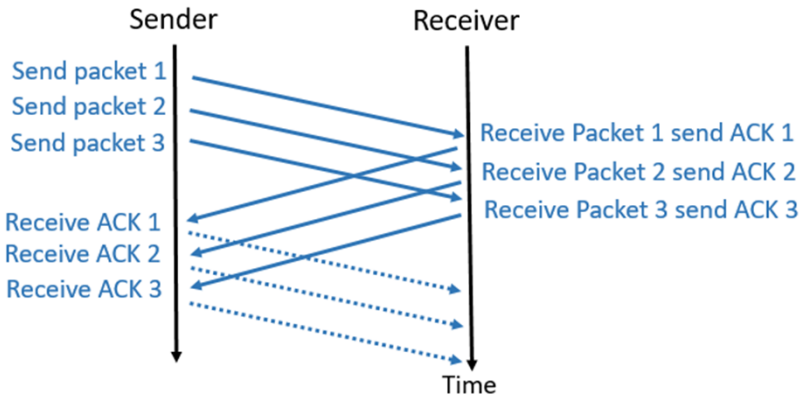
TCP – Retransmission

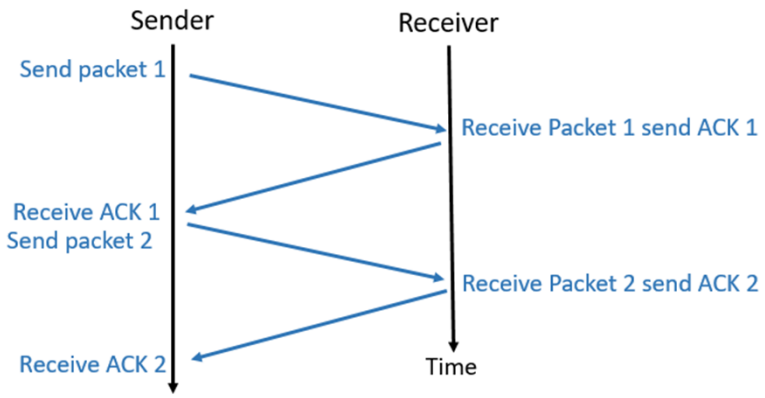
An example of lost segment and sending of 3 duplicated ACK that triggers a fast retransmission.



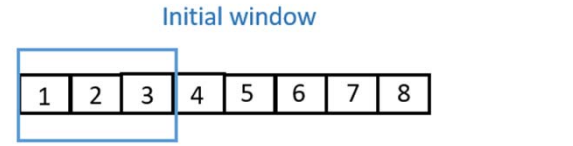
TCP – Sliding Window

- Mechanism to improve overall throughput.
- A simple acknowledgement protocol wastes a substantial amount of network capacity because it must delay sending a new packet until it receives an acknowledge for the previous packet.
- Key idea: allows a sender to transmit multiple packets (window size) before waiting for an acknowledgement.

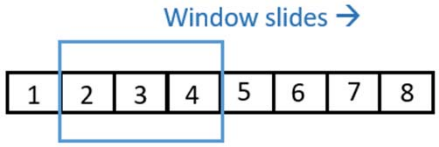




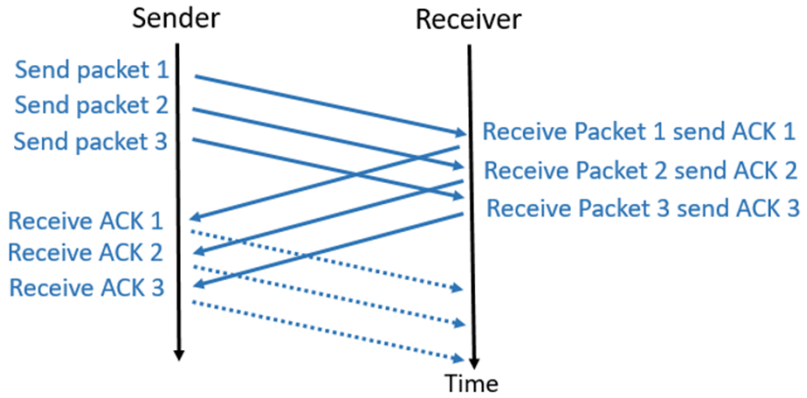
TCP – Flow control :Window size



A sliding window with three packets in the window.



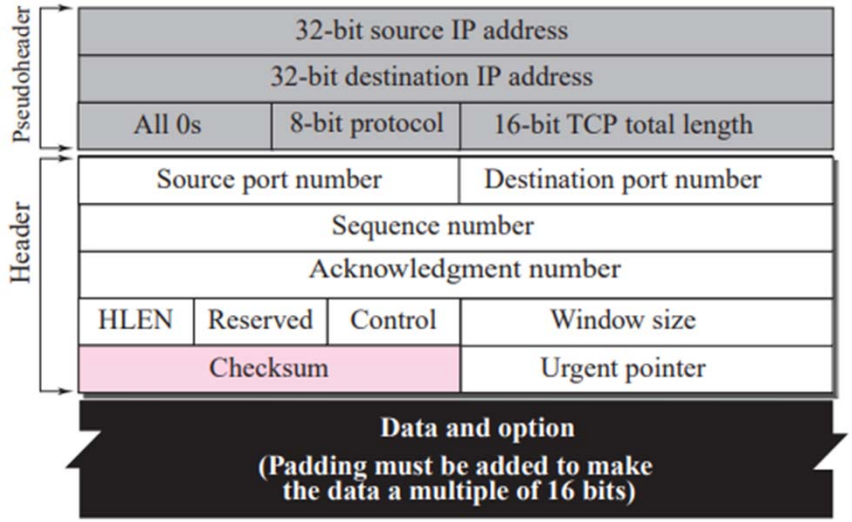
The window sliding – packet 4 can be sent after an acknowledgement has been received for packet 1.



- TCP sliding window mechanism operates at the octet level, not at the packet or segment level.

TCP - Checksum

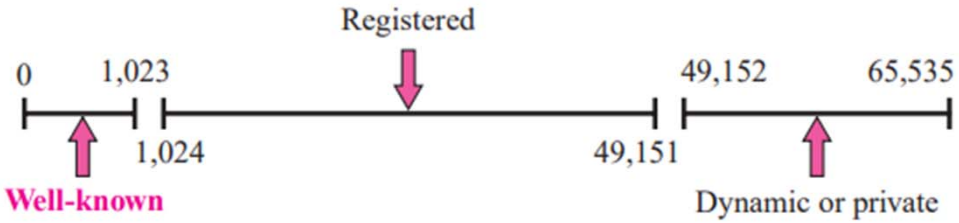
- TCP checksum calculation follows the same procedure as the one described for UDP.
- It includes 3 section: pseudoheader + TCP header + data
- Pseudoheader is part of the IP packet header



TCP – Flow control

- Window size changes over time.
- Each ACK contains a receiver window size that specifies how many more octet it can receive without overflowing its buffer.
- Sender uses a sliding window to determine how many octet it can send without waiting for an ACK.
- Window size increase when receiver process the data and is able to free up buffer
- Window size decrease when receiver buffer fills up.
- Window size of zero is possible if receiver want the sender to stop all transmissions.

Transport Layer - Port numbers



Name	Purpose
Well-known/System Ports	The ports ranging from 0 to 1,023 are assigned and controlled by Internet Assigned Numbers Authority (IANA). Eg. TCP port for HyperText Transfer Protocol (HTTP)
Registered/User Ports	The ports ranging from 1,024 to 49,151 are not assigned or controlled by IANA. They can only be registered with IANA to prevent duplication. Eg. PostgreSQL uses 5432
Dynamic/Private Ports	The ports ranging from 49,152 to 65,535 are neither controlled nor registered. They can be used as temporary or private port numbers. The original recommendation was that the ephemeral port numbers for clients be chosen from this range. However, most systems do not follow this recommendation

Addresses

Feature	MAC Address	IP Address	Port Address
Also called	Physical address / hardware address	Logical address	Service address
Used at	Data Link Layer (Layer 2)	Network Layer (Layer 3)	Transport Layer (Layer 4)
Purpose	Identifies a device on a local network	Identifies a device on a global network	Identifies a **specific process/service** on a device
Uniqueness	Unique to each network interface	Unique per device (in a given network)	Not globally unique; reused across devices
Format	48-bit, e.g., 00:1A:2B:3C:4D:5E	IPv4: 192.168.1.1 IPv6: 2001:db8::1	16-bit, e.g., 80 for HTTP, 53 for DNS
Assigned by	Manufacturer (burned into NIC)	Network administrator or DHCP	Application or system
Changes over time	Usually fixed	Can change (e.g., DHCP)	Changes as apps open/close
Scope	Local Network only (LAN)	Network-wide / Internet-wide	Local to device (used for multiplexing)

End