# Transmission Control Protocol (TCP)

# Today: Transport Layer Protocols



- Transmission Control Protocol (TCP): Reliably sending packets

# Review: IP Reliability



- **Reliability** ensures that packets are received correctly

- IP is **unreliable** and only provides a **best effort** delivery service, which means:
    - Packets may be lost (“dropped”)
    - Packets may be corrupted
    - Packets may be delivered out of order

- It is up to higher level protocols to ensure that the connection is reliable

# Scratchpad: Let’s Design It Together



- Problem: IP packets have a limited size. To send longer messages, we have to manually break messages into packets
  - When sending packets: TCP will automatically split up messages
  - When receiving packets: TCP will automatically reassemble the packets
- Problem: Packets can arrive out of order
  - When sending packets: TCP labels each byte of the message with increasing numbers
  - When receiving packets: TCP can use the numbers to rearrange bytes in the correct order
- Problem: Packets can be dropped
  - When receiving packets: TCP sends an extra message acknowledging that a packet has been received
  - When sending packets: If the acknowledgement doesn’t arrive, re-send the packet

# Transmission Control Protocol (TCP)

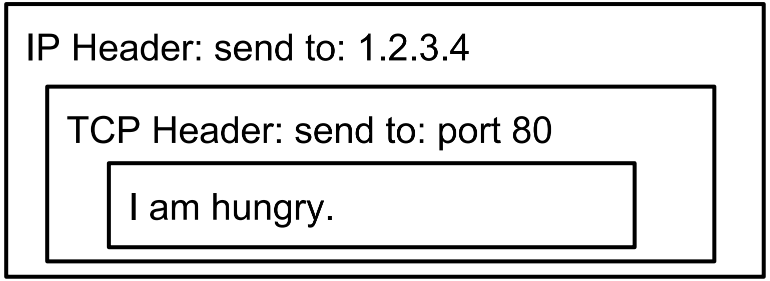


- Provides a byte stream abstraction
  - Bytes go in one end of the stream at the source and come out at the other end at the destination
  - TCP automatically breaks streams into **segments**, which are sent as layer 3 packets
- Provides ordering
  - Segments contain sequence numbers, so the destination can reassemble the stream in order
- Provides reliability
  - The destination sends **acknowledgements** (ACKs) for each sequence number received
  - If the source doesn't receive the ACK, the source sends the packet again
- Provides ports
  - Multiple services can share the same IP address by using different ports

# Ports

ITIS 6200 / 8200

- **Ports** help us distinguish between different applications on the same computer or server
  - On private computers, port numbers can be random
  - On public servers, port numbers should be constant and well-known (so users can access the right port)
- Remember: TCP is built on top of IP, so the IP header (and therefore the IP address) is still present



# Establishing Sequence Numbers



- Each TCP connection requires two sets of sequence numbers
  - One sequence number for messages from the client to the server
  - One sequence number for messages from the server to the client
- Before starting a TCP connection, the client and server must agree on two **initial sequence numbers** (ISNs)
  - The ISNs are different and random for every connection (for security reasons)

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| H | e | l | l | o | | s | e | r | v | e | r |
| 50 | 51 | 52 | 53 | 54 | 55 | 56 | 57 | 58 | 59 | 60 | 61 |

Messages from the client are numbered starting at 50.

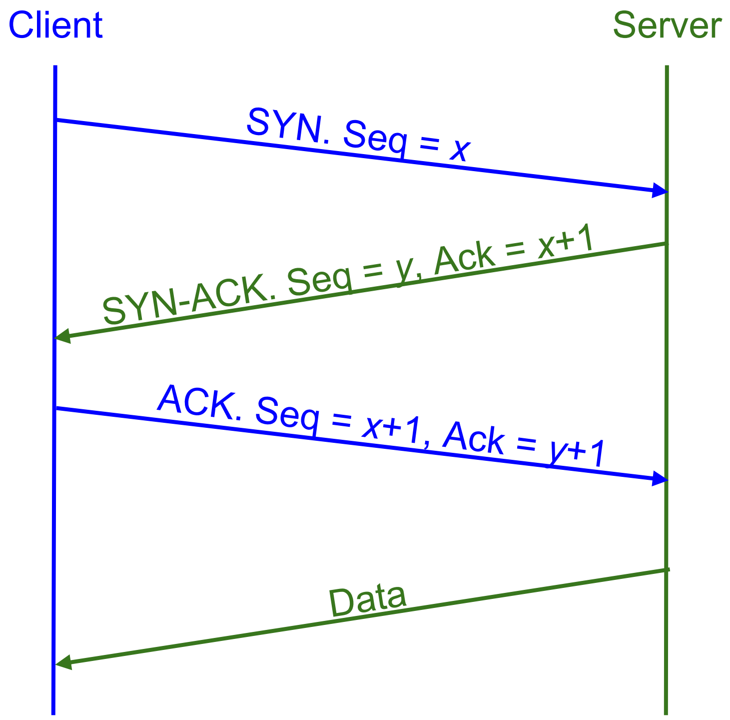
| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| H | e | l | l | o | | c | l | i | e | n | t |
| 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 | 33 | 34 | 35 | 36 |

Messages from the server are numbered starting at 25.

# TCP: 3-Way Handshake



1. Client chooses an initial sequence number *x* its bytes and sends a SYN (synchronize) packet to the server
2. Server chooses an initial sequence number *y* for its bytes and responds with a SYN-ACK packet
3. Client then returns with an ACK packet
4. Once both hosts have synchronized sequence numbers, the connection is "established"



# TCP: Sending and Receiving Data



- The TCP handlers on each side track which TCP segments have been received for each connection
  - A connection is identified by these 5 values (sometimes called a 5-tuple)
    - Source IP
    - Destination IP
    - Source Port
    - Destination Port
    - Protocol
- Data from the bytestream can be presented to the application when all data before has been received and presented
  - Recall: TCP presents data to the application as a bytestream, so the order must be preserved from one end to the other, even if packets are received out of order

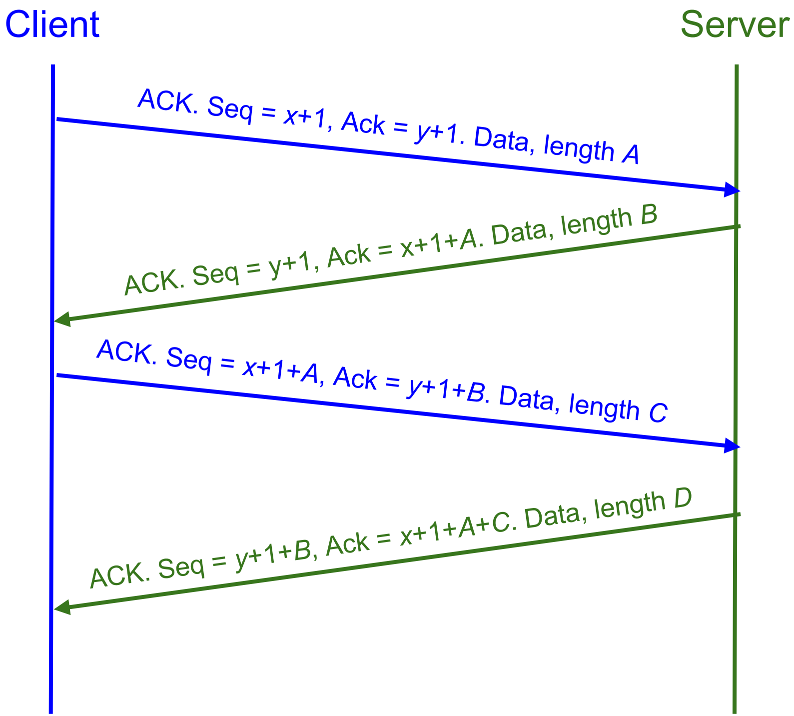
# TCP: Sending and Receiving Data

ITIS 6200 / 8200

- Byte *i* of the bytestream is represented by sequence number $x + i$
  - The first byte is byte $i = 1$, since sequence number $x$ was used for the SYN packet and $y$ for the SYN-ACK packet
- A packet's sequence number is the number of the first byte of its data
  - This number is from the sender's set of sequence numbers
- A packet's ACK number, if the ACK flag is set, is the number of the byte immediately after the last received byte
  - This number is from the receiver's set of sequence numbers
  - This would be (sequence number) + (length of data) for the last received packet

# TCP: Sending and Receiving Data





# TCP: Retransmission



- If a packet is dropped (lost in transit):
  - The recipient will not send an ACK, so the sender will not receive the ACK
  - The sender repeatedly tries to send the packet again until it receives the ACK
- If a packet is received, but the ACK is dropped:
  - The sender tries to send the packet again since it didn't receive the ACK
  - The recipient ignores the duplicate data and sends the ACK again
- When packets are dropped in TCP, TCP assumes that there is congestion and sends the data at a slower rate

# TCP: Ending/Aborting a Connection



- To **end** a connection, one side sends a packet with the FIN (finish) flag set, which should then be acknowledged
  - This means "I will no longer be sending any more packets, but I will continue to receive packets"
  - Once the other side is no longer sending packets, it sends a packet with the FIN flag set
- To **abort** a connection, one side sends a packet with the RST (reset) flag set
  - This means "I will no longer be sending nor receiving packets on this connection"
  - RST packets are not acknowledged since they usually mean that something went wrong

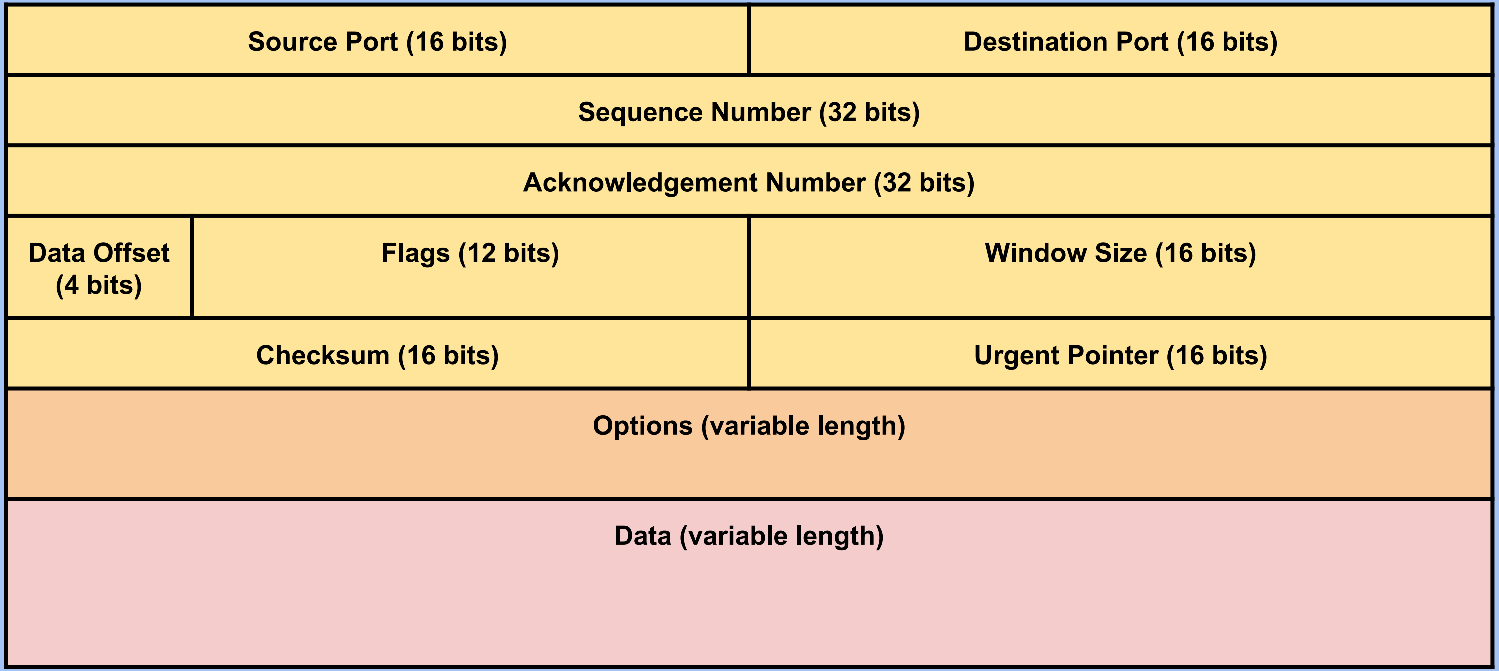
# TCP Flags



- ACK
  - Indicator that the user is acknowledging the receipt of something (in the ack number)
  - Pretty much always set except the very first packet
- SYN
  - Indicator of the beginning of the connection
- FIN
  - One way to end the connection
  - Requires an acknowledgement
  - No longer sending packets, but will continue to receive
- RST
  - One way to end a connection
  - Does not require an acknowledgement
  - No longer sending or receiving packets

# TCP Segment Structure

ITIS 6200 / 8200




TCP segment header

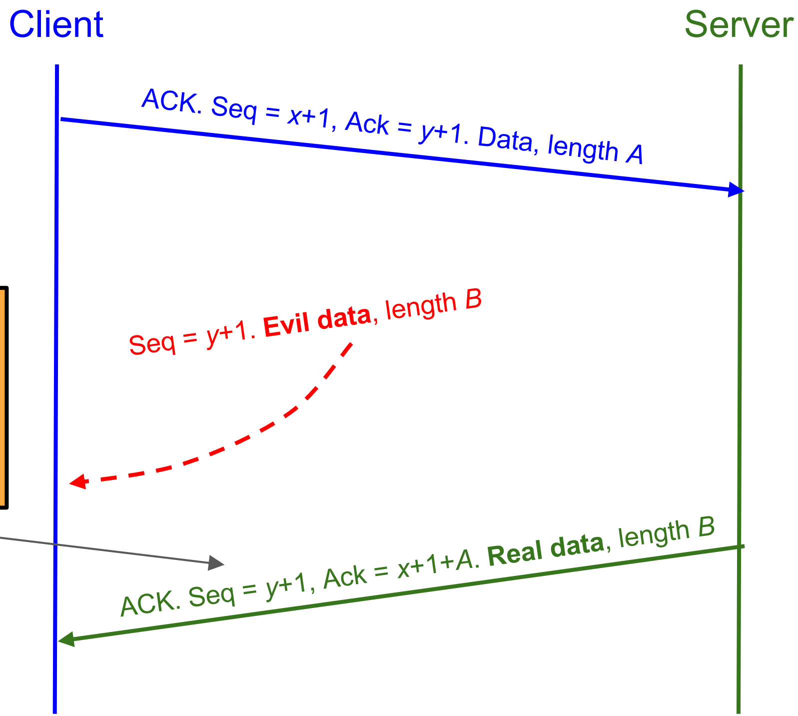
# TCP Attacks



- **TCP hijacking**: Tampering with an existing session to modify or inject data into a connection
    - **Data injection**: Spoofing packets to inject malicious data into a connection
        - Need to know: The sender's sequence number
        - Easy for MITM and on-path attackers, but off-path attackers must guess 32-bit sequence number (called **blind injection/hijacking**, considered difficult)
        - For on-path attackers, this becomes a race condition since they must beat the server's legitimate response
    - **RST injection**: Spoofing a RST packet to forcibly terminate a connection
        - Same requirements as packet injection, so easy for on-path and MITM attackers, but hard for off-path attackers
        - Often used in censorship scenarios to block access to sites

# TCP Data Injection

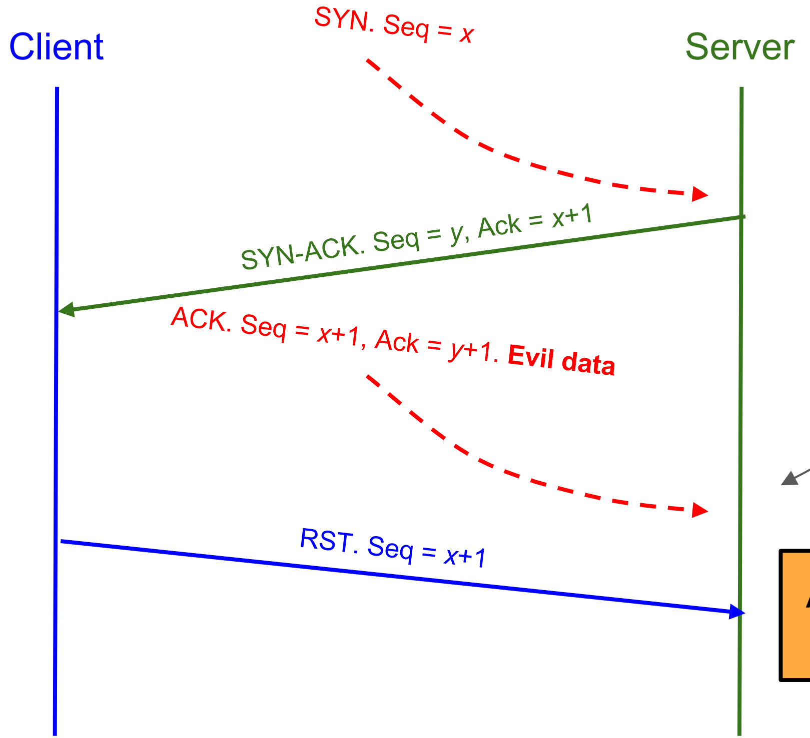





This packet will be ignored by the client since the client already processed the malicious packet!

# TCP Spoofing






An on-path attacker must send the evil data before the server receives the real client's RST!

A MITM attack could just drop the client's packets, however

# TCP Attacks



- **TCP spoofing**: Spoofing a TCP connection to appear to come from another source IP address
  - Need to know: Sequence number in the server's response SYN-ACK packet
  - Easy for MITM and on-path attackers, but off-path attackers must guess 32-bit sequence number (called **blind spoofing**, also considered difficult)
  - For on-path attackers, this is a race condition, since the real client will send a RST upon receiving the server's SYN-ACK!

# TCP Attacks



- TCP provides no confidentiality or integrity
  - Instead, we rely on higher layers to prevent those kind of attacks

- Defense against off-path attackers rely on choosing random sequence numbers
  - Bad randomness can lead to trivial off-path attacks: TCP sequence numbers used to be based on the system clock!

# Summary

ITIS 6200 / 8200

- Transmission Control Protocol (TCP): Reliably sending packets
    - 3-way handshake: Client sends SYN, server sends SYN-ACK, client sends ACK
    - Provides reliability, ordering, and ports
    - Attack: TCP hijacking through data injection or RST injection
        - Blind attacks must guess the client’s or server’s sequence numbers
    - Attack: TCP spoofing by sending a spoofed SYN packet
        - Blind attacks must guess the server’s sequence number