# CMSC414 Computer and Network Security

## BGP and Transport Layer (TCP and UDP)

Yizheng Chen | University of Maryland
surrealyz.github.io

Apr 25, 2024

Credits: original slides from instructors and staff from CS161 at UC Berkeley. Blue slides will not be tested.

# Types of Network Attackers

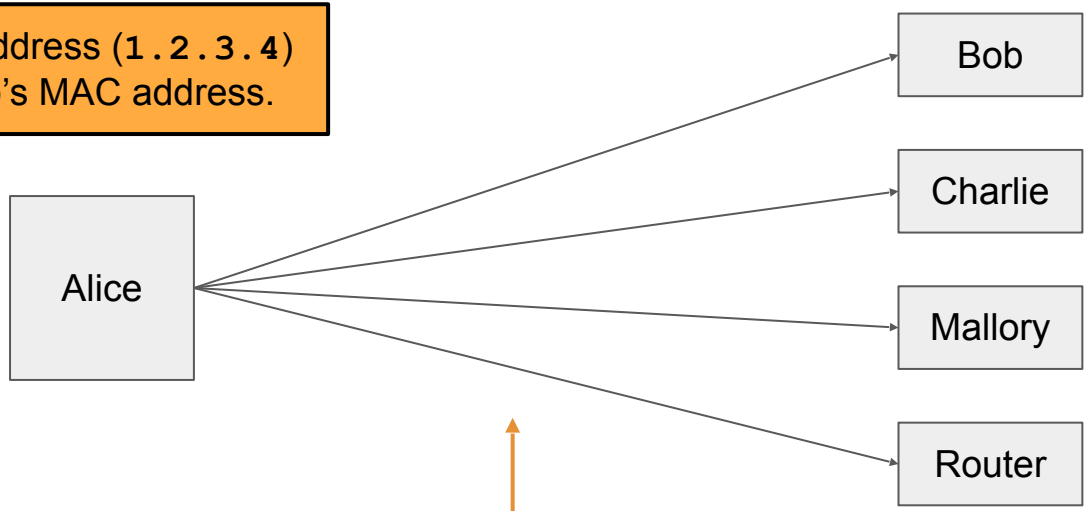
- Threat model: There are 3 types of attackers we'll consider

| | Can modify or delete packets | Can read packets |
|---|---|---|
| **Man-in-the-middle attacker** | ✓ | ✓ |
| **On-path attacker** | | ✓ |
| **Off-path attacker** | | |

# Attacks on ARP

Alice knows Bob's IP address (`1.2.3.4`) but wants to learn Bob's MAC address.

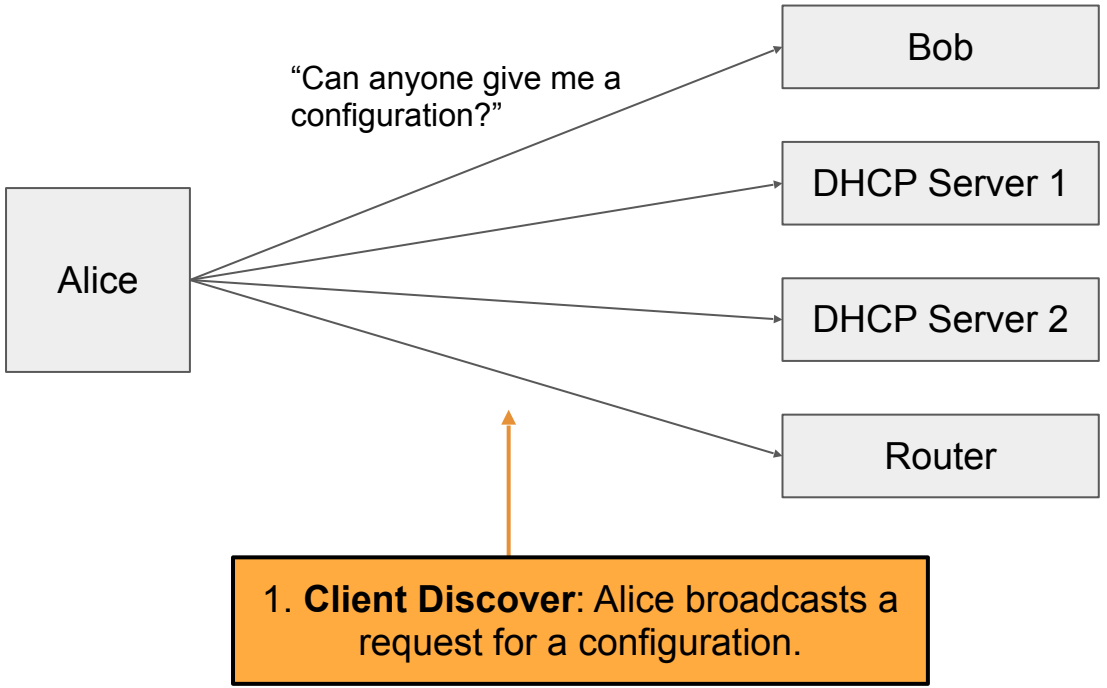
| Alice's cache | |
|---|---|
| IP | MAC |
| | |




2. Alice asks everyone else on the local network: "What is the MAC address of `1.2.3.4`?"

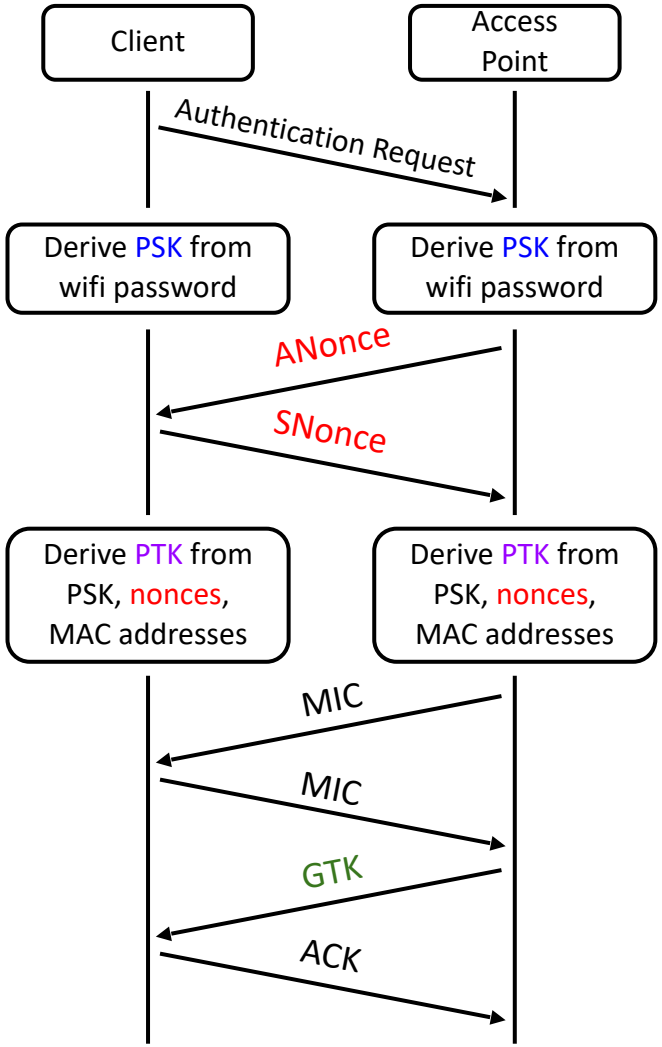
# Dynamic Host Configuration Protocol (DHCP)

| Alice's configuration | |
|---|---|
| My IP | ??? |
| DNS Server | ??? |
| Gateway | ??? |



# WPA Handshake

1. The client sends an authentication request to the access point
2. Both use the password to derive the *PSK* (pre-shared key)
3. Both exchange random nonces
4. Both use the *PSK*, nonces, and MAC addresses to derive the *PTK* (pairwise transport keys)
5. Both exchange MICs (Message Integrity Check, these are MACs from the crypto unit) to ensure no one has tampered with the nonces, and that the PTK was correctly derived
6. The access point encrypts and sends the *GTK* (group temporal key) to the client, used for broadcasts that anyone can decrypt
7. The client acknowledges receiving the GTK



# Today: Transport Layer Protocols

- IP Routing: Border Gateway Protocol (BGP)
- Transmission Control Protocol (TCP): Reliably sending packets
- User Datagram Protocol (UDP): Non-reliably sending packets

# Review: Internet Protocol (IP)

- **Internet Protocol** (**IP**): The universal layer-3 protocol that all devices use to transmit data over the Internet

- **IP address**: An address that identifies a device on the Internet
  - IPv4 is 32 bits (e.g. `35.163.72.93`)
  - IPv6 is 128 bits (e.g. `2607:f140:8801:0000:0000:0000:0001:0023`)
    - Shorthand: omit sets of zeros: `2607:f140:8801::1:23`
  - (Almost) globally unique from any single perspective
    - In reality not
  - IP addresses help nodes make decisions on where to forward the packet

# Subnets

- Recall: Layer 3 routes packets across multiple nodes on different LANs

- IP routes by **subnets**, which are groups of addresses with a common prefix
  - A subnet is written as a prefix followed by the length of the prefix
    - Example: **128.32.0.0/16** is an IPv4 subnet with all addresses that begin with the prefix of **128.32.***
    - Since an IPv4 is a 32-bit address and there are 16 bits in the prefix, this subnet represents $2^{32 - 16} = 2^{16}$ addresses

# Routing Packets

- To send a packet to a computer within the local network:
  - Verify that the destination IP is in the same subnet
  - Use ARP (or contact a switch) to get the destination MAC address
  - Send the packet directly to the destination using the destination MAC address

- To send a packet to a computer that is not within the local network:
  - Send the packet to the gateway
  - Past the gateway, the packet goes to the Internet
  - It’s the gateway’s job to deliver the packet closer to the destination

# Autonomous Systems

- Once your system sends the packet to the gateway, the packet has to be routed through the Internet

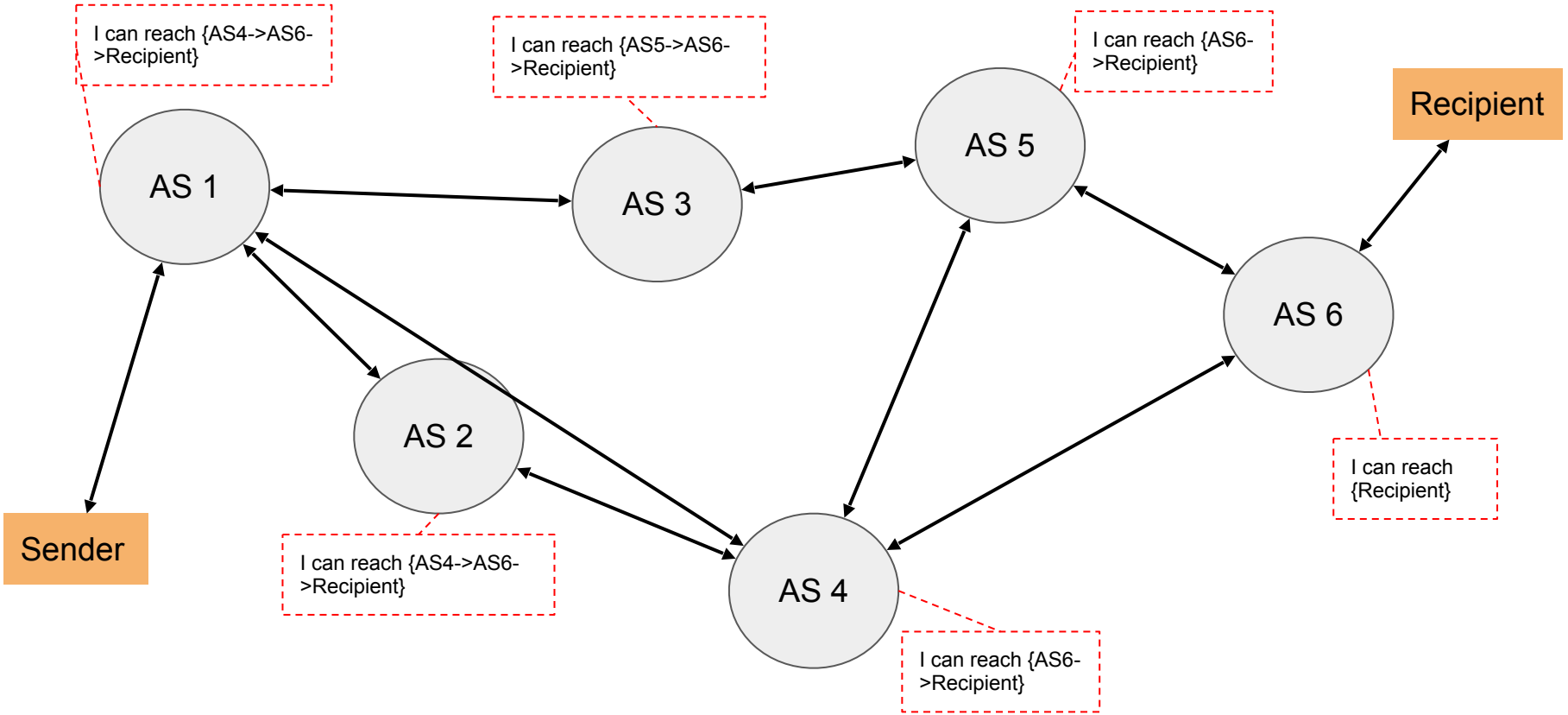
- The Internet is a network of networks, comprised of many **autonomous systems** (**AS**)
  - Each AS handles its own internal routing
  - Each AS is uniquely identified by its **autonomous system number** (**ASN**)
  - Each AS is comprised of one or more LANs
  - The AS can forward packet to other connected ASes

- The protocol for communicating between different Autonomous Systems is **Border Gateway Protocol** (**BGP**)

# BGP

- The protocol for communicating between different Autonomous Systems is **Border Gateway Protocol** (**BGP**)
  - Each router announces what networks it can provide and the path onward from the router
  - The most precise route with the shortest path and no loops is the preferred route

# BGP


I can reach {AS4->AS6->Recipient}
I can reach {AS5->AS6->Recipient}
I can reach {AS6->Recipient}
Recipient
AS 1
AS 3
AS 5
AS 6
AS 2
AS 4
Sender
I can reach {Recipient}
I can reach {AS4->AS6->Recipient}
I can reach {AS6->Recipient}

# IP and BGP Attacks

- Each AS implicitly trusts the surrounding ASes and accepts advertised routes

- **BGP hijacking**: A malicious autonomous system can lie and claims itself to be responsible for a network which it isn't
  - Example: AS3 broadcasts that it is responsible for 128.32.0.0/16
    - Now, the malicious AS can act as a MITM for traffic to 128.32.0.0!
  - Rely on defenses from higher levels

- **IP spoofing**: Malicious clients can send IP packets with source IP values set to a spoofed value
  - Edge ASes should block packets with source IPs set to the wrong value, but some don't
  - Rely on defenses from higher layers ("defense in depth")

# Review: IP Reliability

- 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

- **Reliability** ensures that packets are received correctly
  - IP packets include a checksum (unkeyed function, protects against random errors)
  - No guarantees against an attacker

# How TCP Addresses the 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
- Now the user doesn’t need to manually split up messages!

Provides a byte stream abstraction

# How TCP Addresses the 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

Provides ordering

# How TCP Addresses the 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

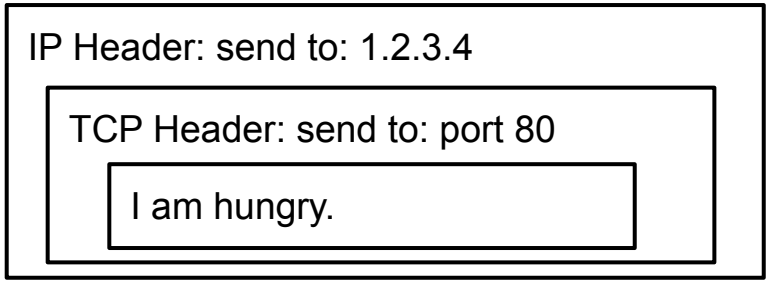
Provides reliability

# 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

- **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, as we'll see soon)

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

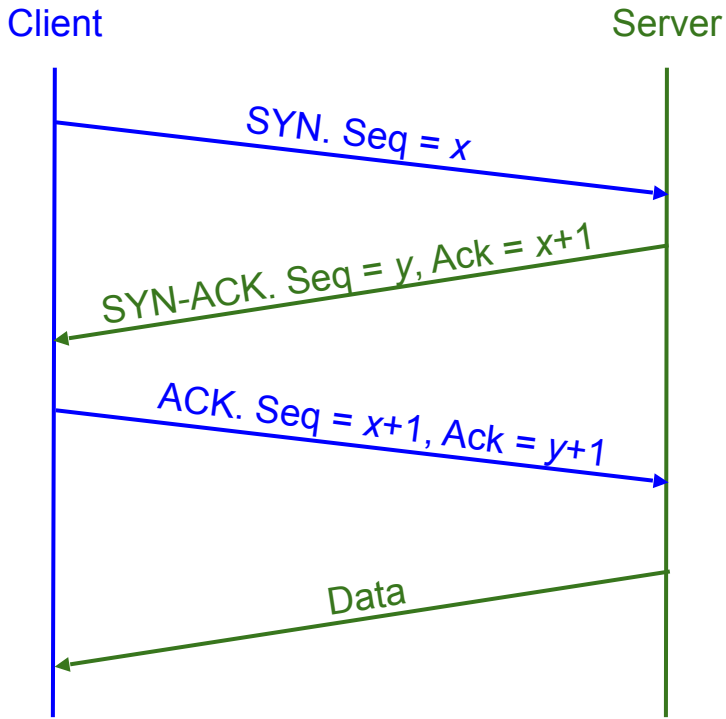
Messages from the client are numbered starting at 50.

| 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

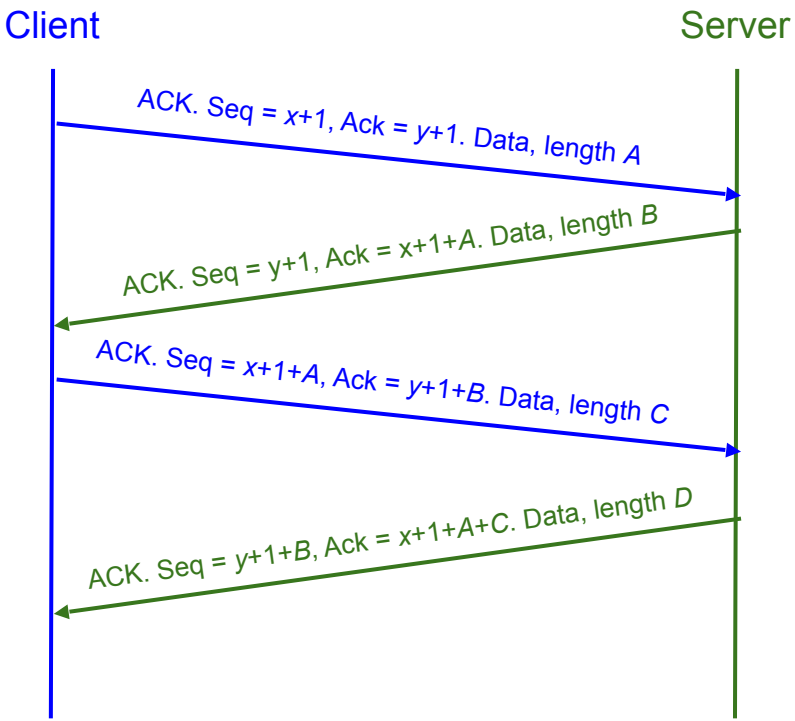
1. Client chooses an initial sequence number *x* its bytes and sends a SYN (synchronize) packet to the server
2. Server chooses an initial sequence number *y* for its bytes and responds with a SYN-ACK packet
3. Client then returns with an ACK packet
4. Once both hosts have synchronized sequence numbers,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



# TCP: Sending and Receiving Data

- 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: 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

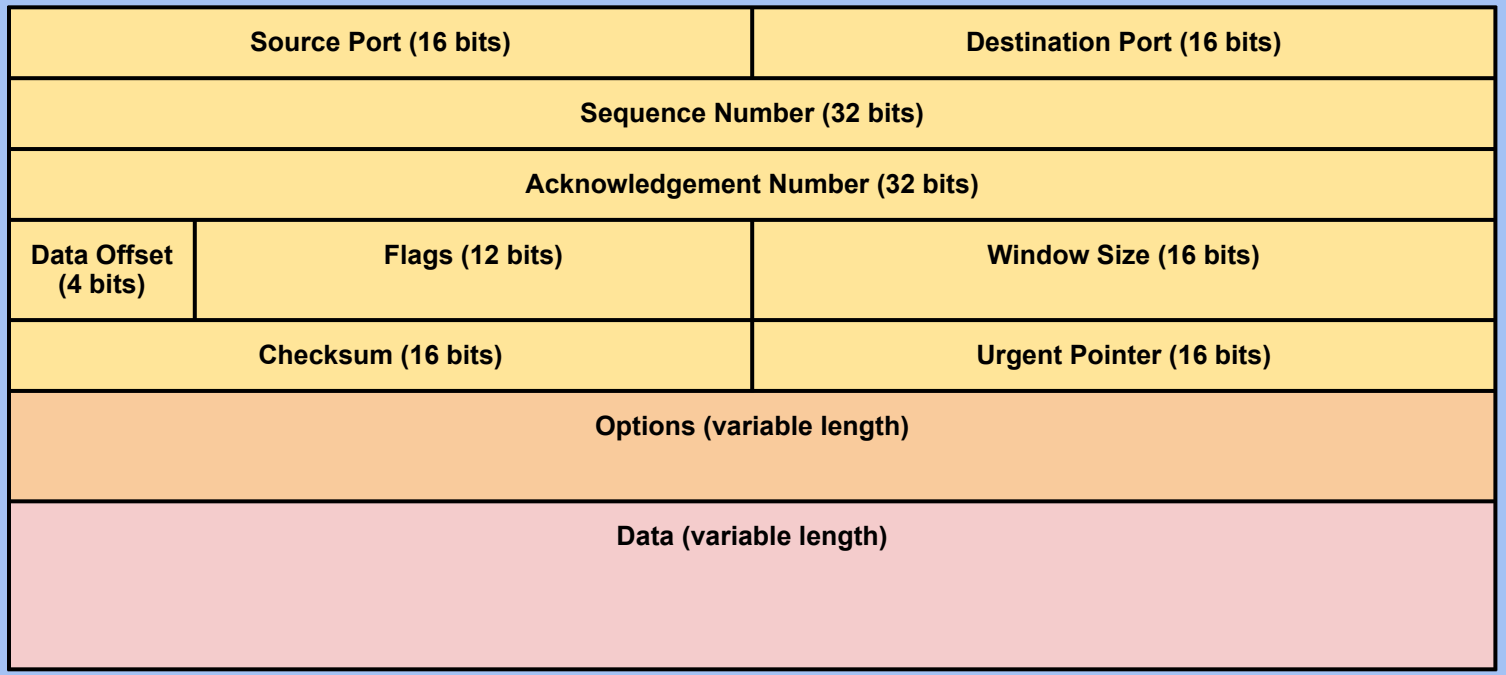
- To **end** a connection, one side sends a packet with the FIN (finish) flag set, which should then be acknowledged
  - This means “I will no longer be sending any more packets, but I will continue to receive packets”

- 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”

# 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 Packet Structure




TCP segment header

# TCP Attacks

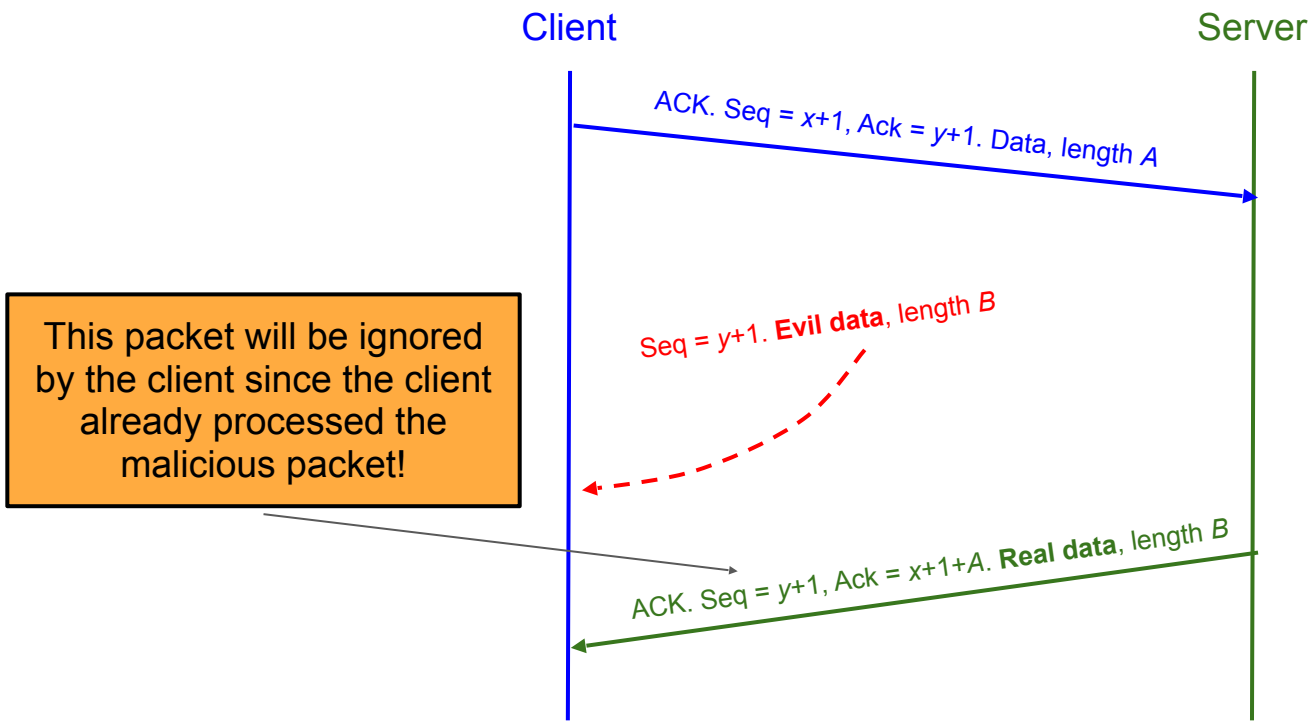
- **TCP hijacking**: Tampering with an existing session to modify or inject data into a connection

- **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

# TCP Attacks

- **RST injection**: Spoofing a RST packet to forcibly terminate a connection
  - Need to know: The sender's sequence number
  - Easy for on-path and MITM attackers, but hard for off-path attackers
  - Often used in censorship scenarios to block access to sites

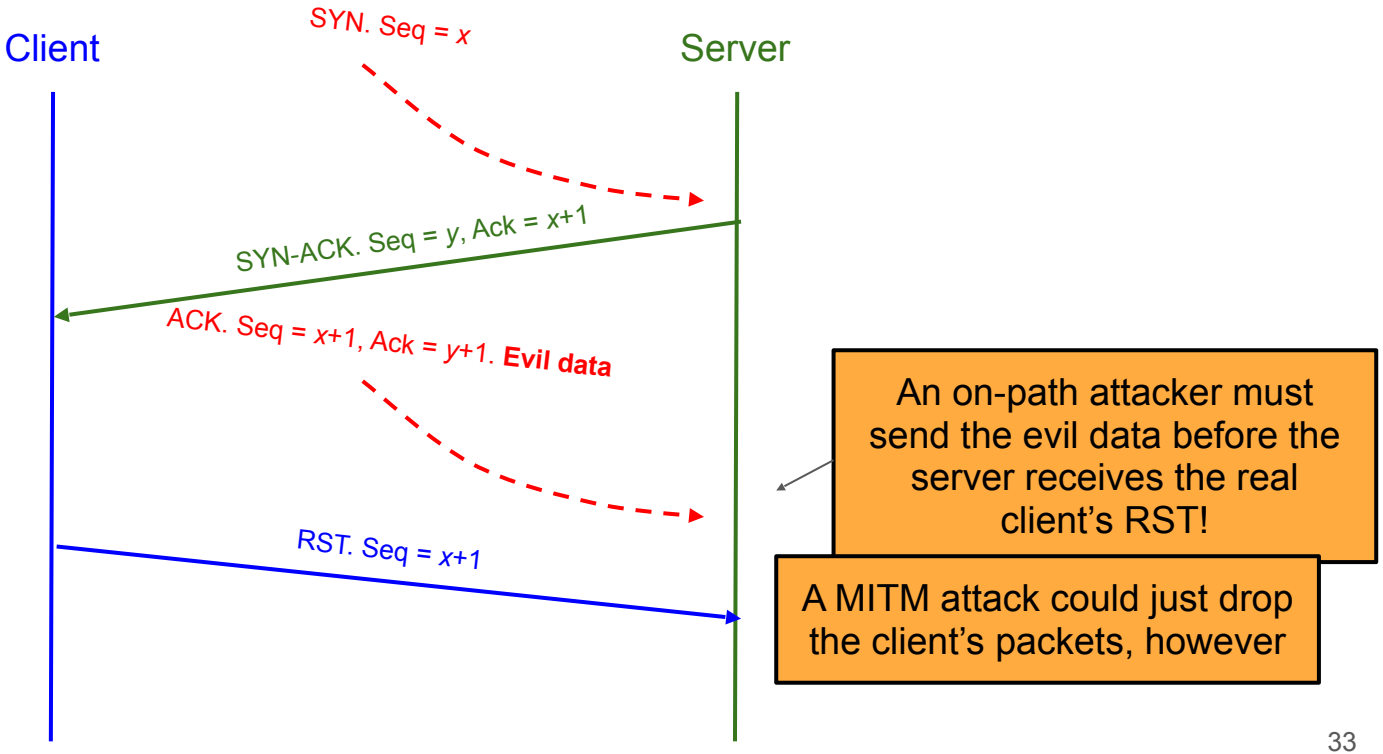
# TCP Data Injection



# TCP Attacks

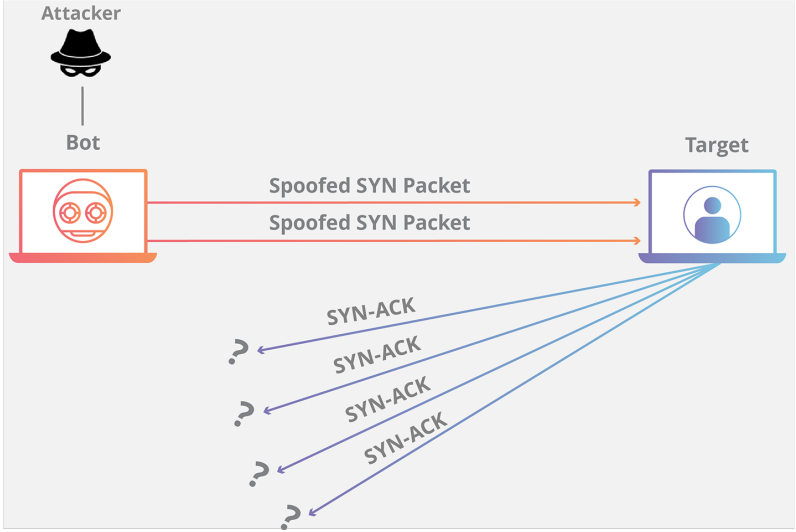
- **TCP spoofing**: Spoofing a TCP connection to appear to come from another source IP address
  - Recall: IP packets can often be spoofed if the AS doesn't block source addresses
  - Need to know: Sequence number in the server's response SYN-ACK packet
  - Easy for MITM and on-path attackers, but off-path 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 Spoofing



# TCP SYN Food Attacks

1. The attacker sends a high volume of SYN packets to the targeted server, often with spoofed IP addresses.
2. The server then responds to each one of the connection requests and leaves an open port ready to receive the response.
3. While the server waits for the final ACK packet, which never arrives, the attacker continues to send more SYN packets. The arrival of each new SYN packet causes the server to temporarily maintain a new open port connection for a certain length of time, and once all the available ports have been utilized the server is unable to function normally.



# TCP Attacks

- TCP provides no confidentiality or integrity
  - Instead, we rely on higher layers (like TLS) 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!

# User Datagram Protocol (UDP)

- Provides a **datagram** abstraction
  - A message, sent in a single layer 3 packet (though layer 3 could fragment the packet)
  - Max size limited by max size of packet
  - Applications break their data into datagrams, which are sent and received as a single unit
    - Contrast with TCP, where the application can use a bytestream abstraction

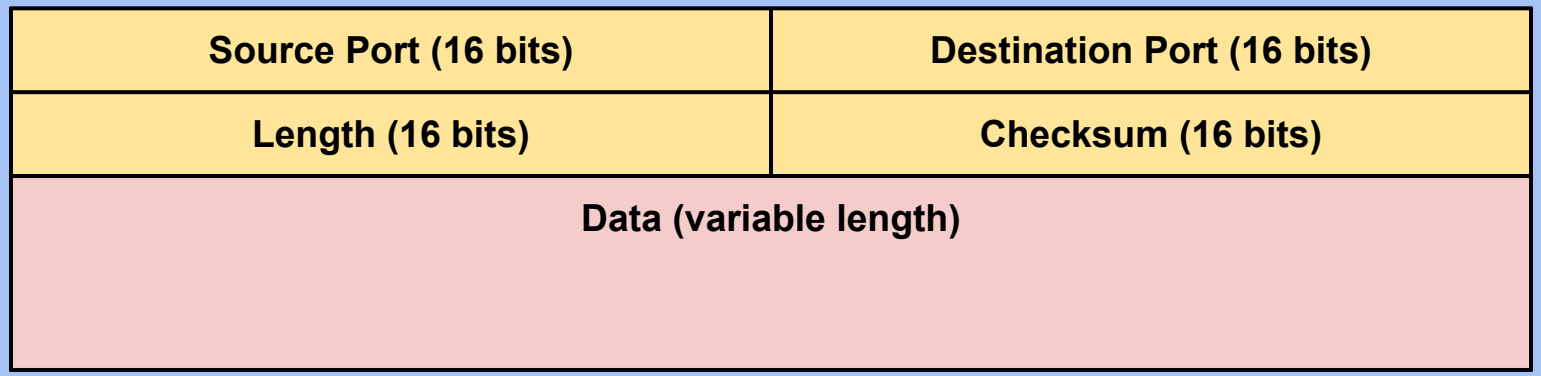
- No reliability or ordering guarantees, but adds ports
  - It still has *best effort* delivery

- Much faster than TCP, since there is no 3-way handshake
  - Usually used by low-latency, high-speed applications where errors are okay (e.g. video streaming, games)

# UDP Attacks

- No sequence numbers, so relatively easy to inject data into a connection or spoof connections
  - Higher layers must provide their own defenses against these attacks!

# UDP Packet Structure




UDP datagram header