Networking Attacks: Link-, IP-, and TCP-layer attacks

CS 161: Computer Security

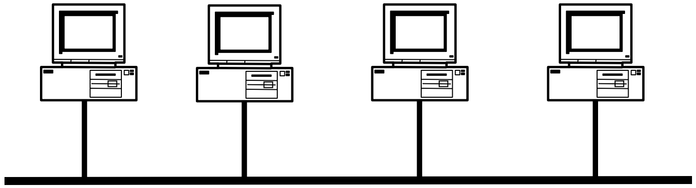
Prof. David Wagner

February 28, 2013

General Communication Security Goals: CIA

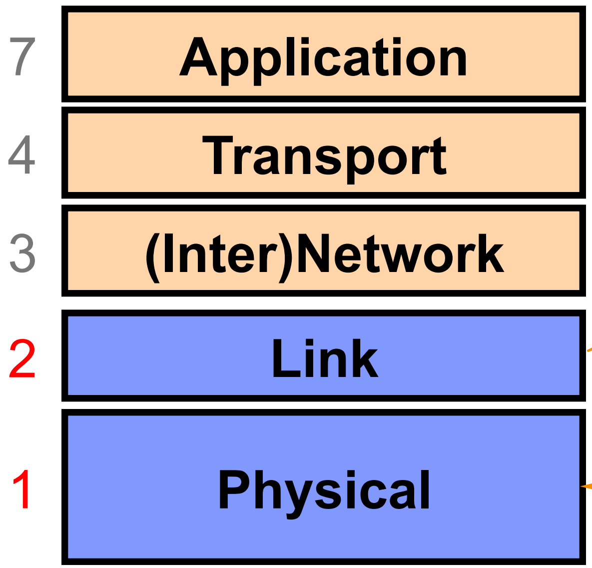
- Confidentiality:
  - No one can *read* our data / communication unless we want them to
- Integrity
  - No one can *manipulate* our data / processing / communication unless we want them to
- Availability
  - We can *access* our data / conduct our processing / use our communication capabilities when we want to
- Also: no *additional* traffic other than ours …

Link-layer threats



- Confidentiality: eavesdropping (aka sniffing)
- Integrity: injection of spoofed packets
- Injection: delete legit packets (e.g., jamming)

Layers 1 & 2: General Threats?



Framing and transmission of a collection of bits into individual messages sent across a single "subnetwork" (one physical technology)

Encoding bits to send them over a single **physical link**

e.g. patterns of

voltage levels /

photon intensities /

RF modulation

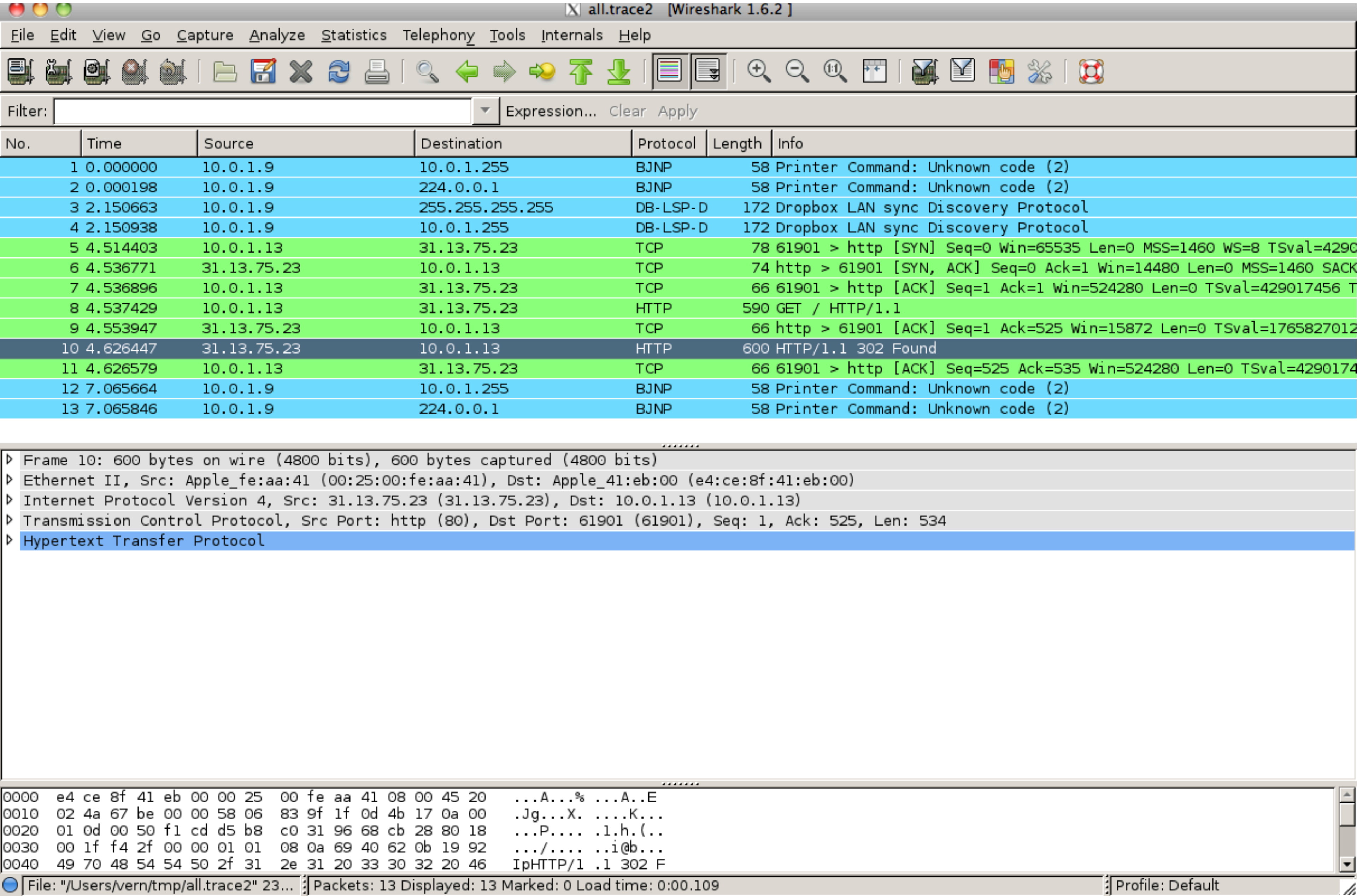
Eavesdropping

- For subnets using broadcast technologies (e.g., WiFi, some types of Ethernet), get it for "free"
  - Each attached system's NIC (= Network Interface Card) can capture any communication on the subnet
  - Some handy tools for doing so
    - tcpdump / windump (low-level ASCII printout)
    - Wireshark (GUI for displaying 800+ protocols)

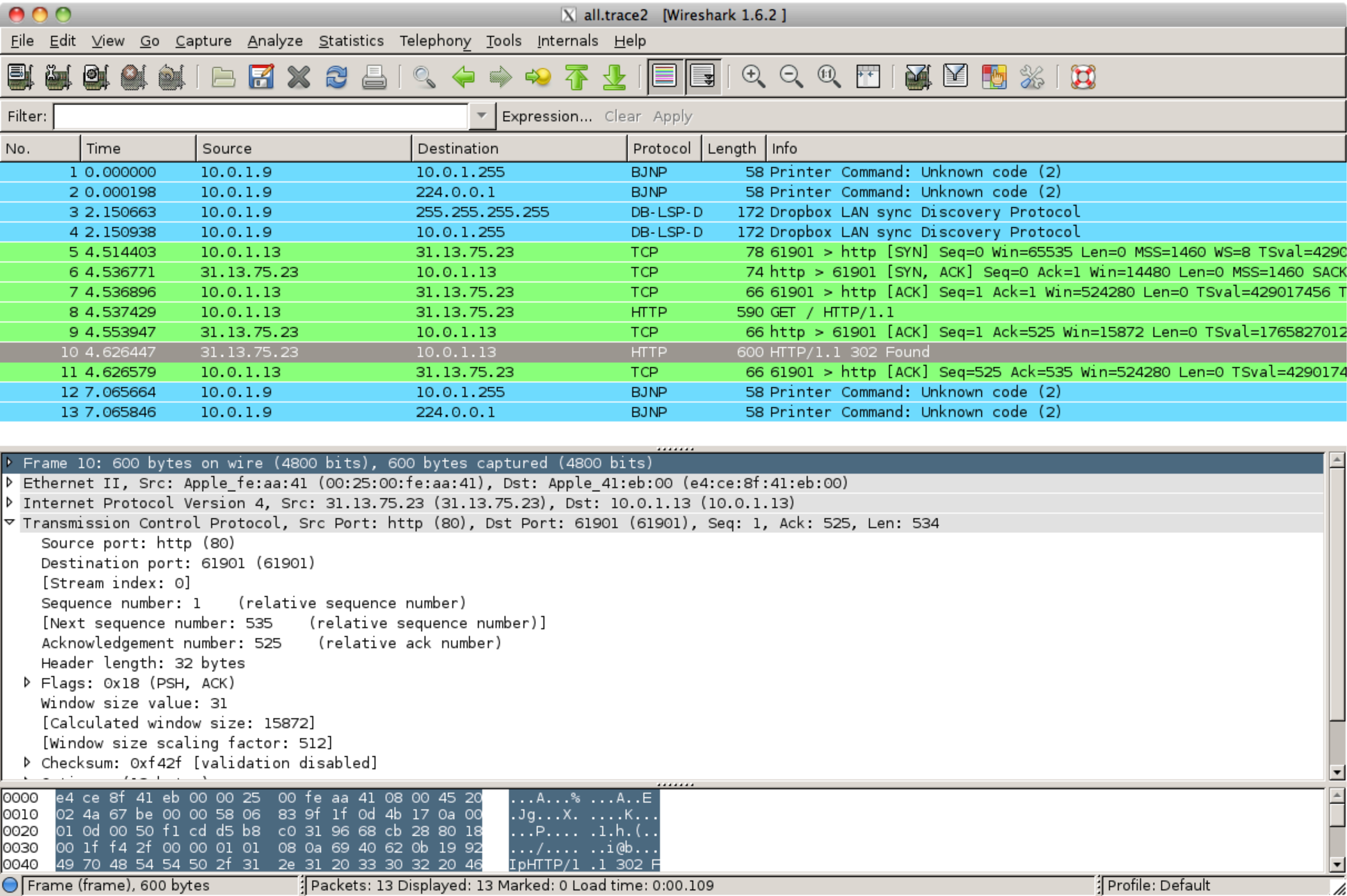
TCPDUMP: Packet Capture & ASCII Dumper

```
demo 2 % tcpdump -r all.trace2
reading from file all.trace2, link-type EN10MB (Ethernet)
21:39:37.772367 IP 10.0.1.9.60627 > 10.0.1.255.canon-bjnp2: UDP, length 16
21:39:37.772565 IP 10.0.1.9.62137 > all-systems.mcast.net.canon-bjnp2: UDP, length 16
21:39:39.923030 IP 10.0.1.9.17500 > broadcasthost.17500: UDP, length 130
21:39:39.923305 IP 10.0.1.9.17500 > 10.0.1.255.17500: UDP, length 130
21:39:42.286770 IP 10.0.1.13.61901 > star-01-02-pao1.facebook.com.http: Flags [S], seq 2
523449627, win 65535, options [mss 1460,nop,wscale 3,nop,nop,TS val 429017455 ecr 0,sack
OK,eol], length 0
21:39:42.309138 IP star-01-02-pao1.facebook.com.http > 10.0.1.13.61901: Flags [S.], seq
3585654832, ack 2523449628, win 14480, options [mss 1460,sackOK,TS val 1765826995 ecr 42
9017455,nop,wscale 9], length 0
21:39:42.309263 IP 10.0.1.13.61901 > star-01-02-pao1.facebook.com.http: Flags [.], ack 1
, win 65535, options [nop,nop,TS val 429017456 ecr 1765826995], length 0
21:39:42.309796 IP 10.0.1.13.61901 > star-01-02-pao1.facebook.com.http: Flags [P.], seq
1:525, ack 1, win 65535, options [nop,nop,TS val 429017456 ecr 1765826995], length 524
21:39:42.326314 IP star-01-02-pao1.facebook.com.http > 10.0.1.13.61901: Flags [.], ack 5
25, win 31, options [nop,nop,TS val 1765827012 ecr 429017456], length 0
21:39:42.398814 IP star-01-02-pao1.facebook.com.http > 10.0.1.13.61901: Flags [P.], seq
1:535, ack 525, win 31, options [nop,nop,TS val 1765827083 ecr 429017456], length 534
21:39:42.398946 IP 10.0.1.13.61901 > star-01-02-pao1.facebook.com.http: Flags [.], ack 5
35, win 65535, options [nop,nop,TS val 429017457 ecr 1765827083], length 0
21:39:44.838031 IP 10.0.1.9.54277 > 10.0.1.255.canon-bjnp2: UDP, length 16
21:39:44.838213 IP 10.0.1.9.62896 > all-systems.mcast.net.canon-bjnp2: UDP, length 16
```

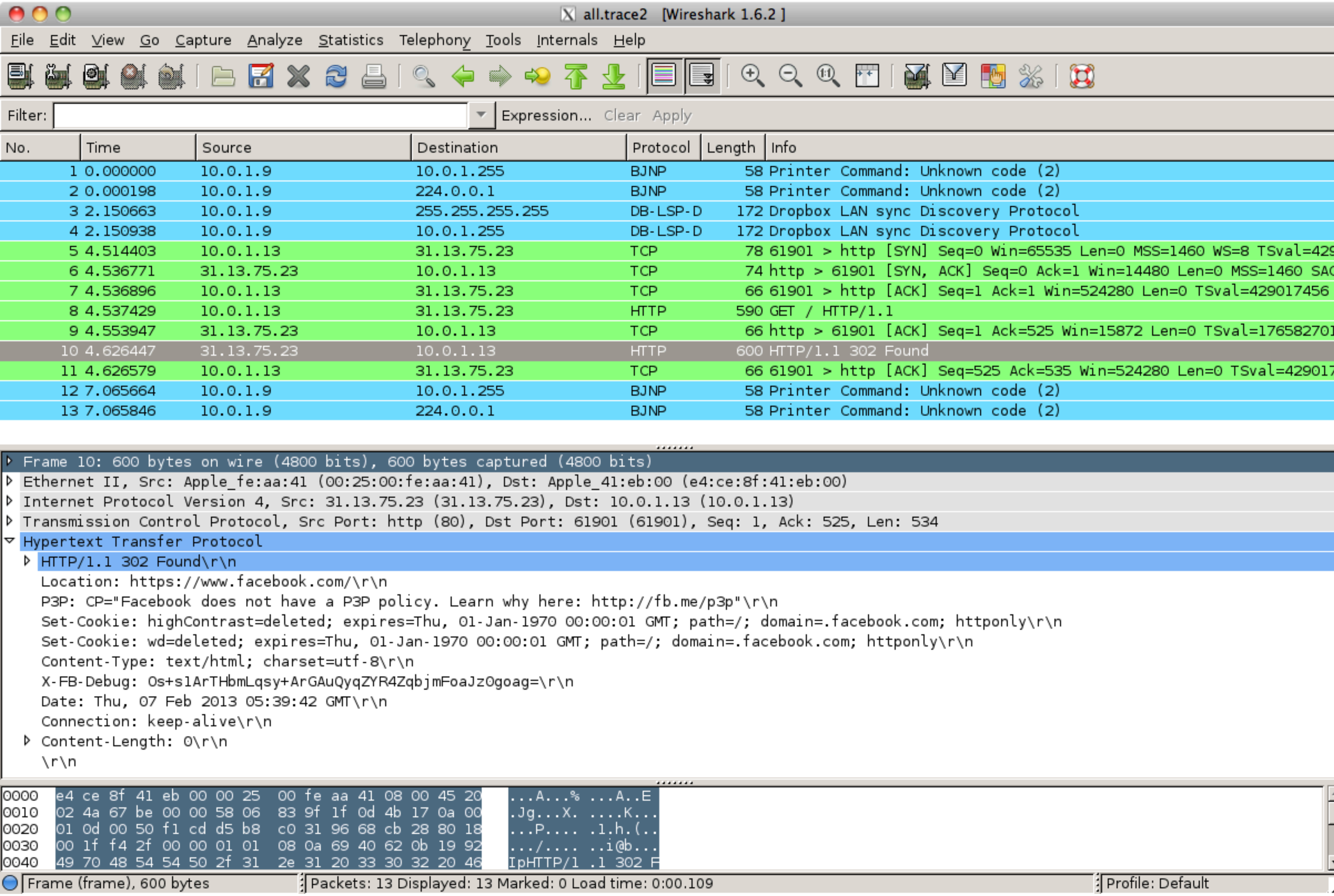
Wireshark: GUI for Packet Capture/Exam.



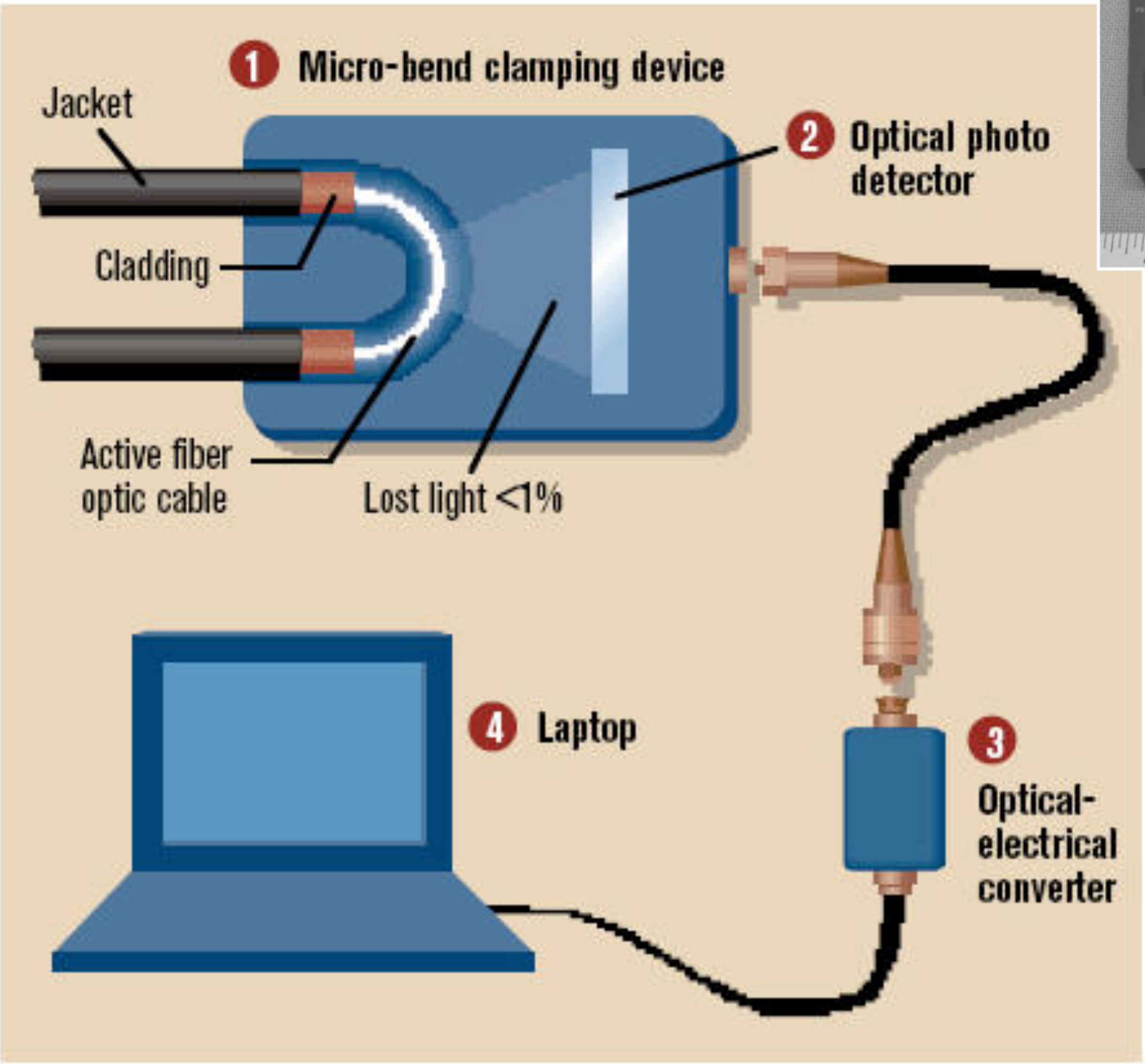
Wireshark: GUI for Packet Capture/Exam.

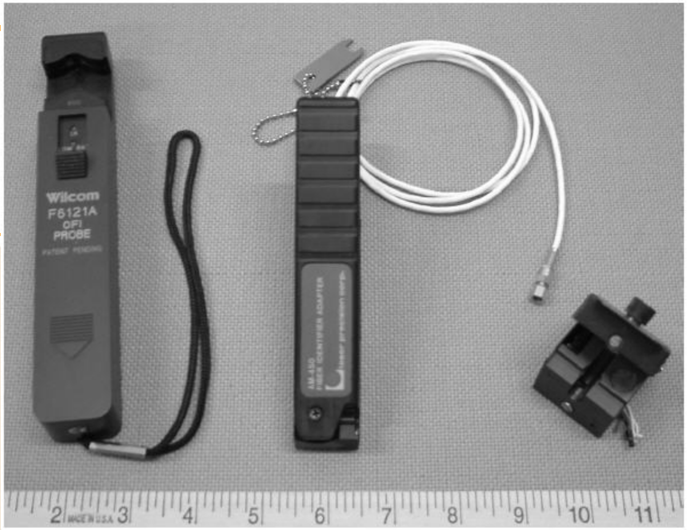


Wireshark: GUI for Packet Capture/Exam.



Stealing Photons





Operation Ivy Bells

By Matthew Carle
Military.com

At the beginning of the 1970's, divers from the specially-equipped submarine, USS Halibut (SSN 587), left their decompression chamber to start a bold and dangerous mission, code named "Ivy Bells".

The Regulus guided missile submarine, USS Halibut (SSN 587) which carried out Operation Ivy Bells.

In an effort to alter the balance of Cold War, these men scoured the ocean floor for a five-inch diameter cable carry secret Soviet communications between military bases.

The divers found the cable and installed a 20-foot long listening device on the cable. designed to attach to the cable without piercing the casing, the device recorded all communications that occurred. If the cable malfunctioned and the Soviets raised it for repair, the bug, by design, would fall to the bottom of the ocean. Each month Navy divers retrieved the recordings and installed a new set of tapes.

Upon their return to the United States, intelligence agents from the NSA analyzed the recordings and tried to decipher any encrypted information. The Soviets apparently were confident in the security of their communications lines, as a surprising amount of sensitive information traveled through the lines without encryption.

prison. The original tap that was discovered by the Soviets is now on exhibit at the KGB museum in Moscow.

Link-Layer Threat: Disruption

- If attacker sees a packet he doesn't like, he can jam it (integrity)
- Attacker can also overwhelm its signaling, e.g., jam WiFi's RF (denial-of-service)

Link-Layer Threat: Disruption

- If attacker sees a packet he doesn't like, he can jam it (integrity)
- Attacker can also overwhelm its signaling, e.g., jam WiFi's RF (denial-of-service)
- There's also the heavy-handed approach …

Sabotage attacks knock out phone service

Nanette Asimov, Ryan Kim,Kevin Fagan, Chronicle Staff Writers
Friday, April 10, 2009

PRINT E-MAIL SHARE COMMENTS (477) FONT | SIZE: – +

(04-10) 04:00 PDT SAN JOSE --

Police are hunting for vandals who chopped fiber-optic cables and killed landlines, cell phones and Internet service for tens of thousands of people in Santa Clara, Santa Cruz and San Benito counties on Thursday.

IMAGES



View More Images

MORE NEWS

- Toyota seeks damage control, in public and private 02.09.10
- Snow shuts down federal government, life goes on 02.09.10
- Iran boosts nuclear enrichment, drawing warnings 02.09.10

The sabotage essentially froze operations in parts of the three counties at hospitals, stores, banks and police and fire departments that rely on 911 calls, computerized medical records, ATMs and credit and debit cards.

The full extent of the havoc might not be known for days, emergency officials said as they finished repairing the damage late Thursday.

Whatever the final toll, one thing is certain: Whoever did this is in a world of trouble if he, she or they get caught.

"I pity the individuals who have done this," said San Jose Police Chief Rob Davis.

Ten fiber-optic cables carrying were cut at four locations in the predawn darkness. Residential and business customers quickly found that telephone service was perhaps more laced into their everyday needs than they thought. Suddenly they couldn't draw out money, send text messages, check e-mail or Web sites, call anyone for help, or even check on friends or relatives down the road.

Several people had to be driven to hospitals because they were unable to summon ambulances. Many businesses lapsed into idleness for hours, without the ability to contact associates or customers.

More than 50,000 landline customers lost service - some were residential, others were business lines that needed the connections for ATMs, Internet and bank card transactions. One line alone could affect hundreds of users.

NEWS | LOCAL BEAT

$250K Reward Out for Vandals Who Cut AT&T Lines

Local emergency declared during outage

By LORI PREUITT

Updated 2:12 PM PST, Fri, Apr 10, 2009

PRINT EMAIL SHARE BUZZ UP! TWITTER FACEBOOK

AT&T is now offering a $250,000 reward for information leading to the arrest of whoever is responsible for severing lines fiber optic cables in San Jose tha left much of the area without phone or cell service Thursday.

John Britton of AT&T said the reward is the largest ever offered by the company.

Link-Layer Threat: Spoofing

- Attacker can inject spoofed packets, and lie about the source address

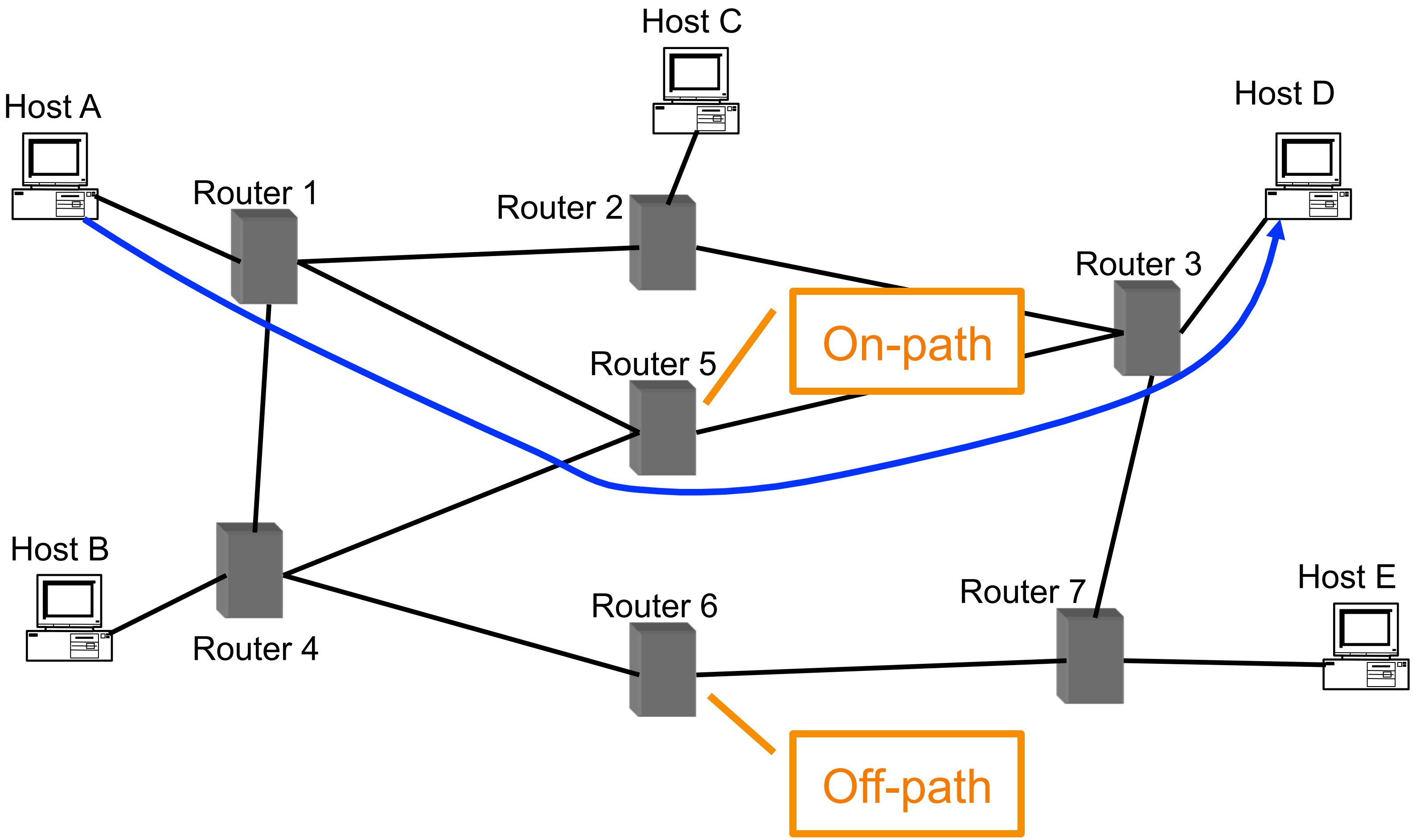
`D`	`C`	`Hello world!`

Physical/Link-Layer Threats: *Spoofing*

- With physical access to a subnetwork, attacker can create any message they like
  - When with a bogus source address: *spoofing*
- When using a typical computer, may require root/administrator to have full freedom
- Particularly powerful when combined with *eavesdropping*
  - Because attacker can understand exact state of victim's communication and craft their spoofed traffic to match it
  - Spoofing w/o eavesdropping = *blind spoofing*

On-path vs Off-path Spoofing

Host A communicates with Host D



Spoofing on the Internet

- On-path attackers can see victim's traffic $\Rightarrow$ spoofing is easy
- Off-path attackers can't see victim's traffic
  - They have to resort to blind spoofing
  - Often must guess/infer header values to succeed
    - We then care about work factor: how hard is this
  - But sometimes they can just **brute force**
    - E.g., 16-bit value: just try all 65,536 possibilities!
- When we say an attacker "can spoof", we usually mean "w/ reasonable chance of success"

Layer 3: General Threats?

- 7 **Application**
- 4 **Transport**
- 3 **(Inter)Network**
- 2 **Link**
- 1 **Physical**

Bridges multiple "subnets" to provide *end-to-end* internet connectivity between nodes

4-bit Version	4-bit Header Length	8-bit Type of Service (TOS)	16-bit Total Length (Bytes)
16-bit Identification		3-bit Flags	13-bit Fragment Offset
8-bit Time to Live (TTL)	8-bit Protocol	16-bit Header Checksum	
32-bit Source IP Address			
32-bit Destination IP Address			
Payload			

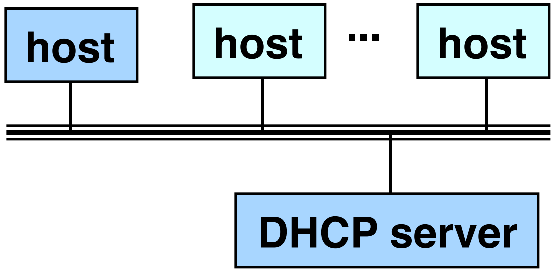
IP = Internet Protocol

IP-Layer Threats

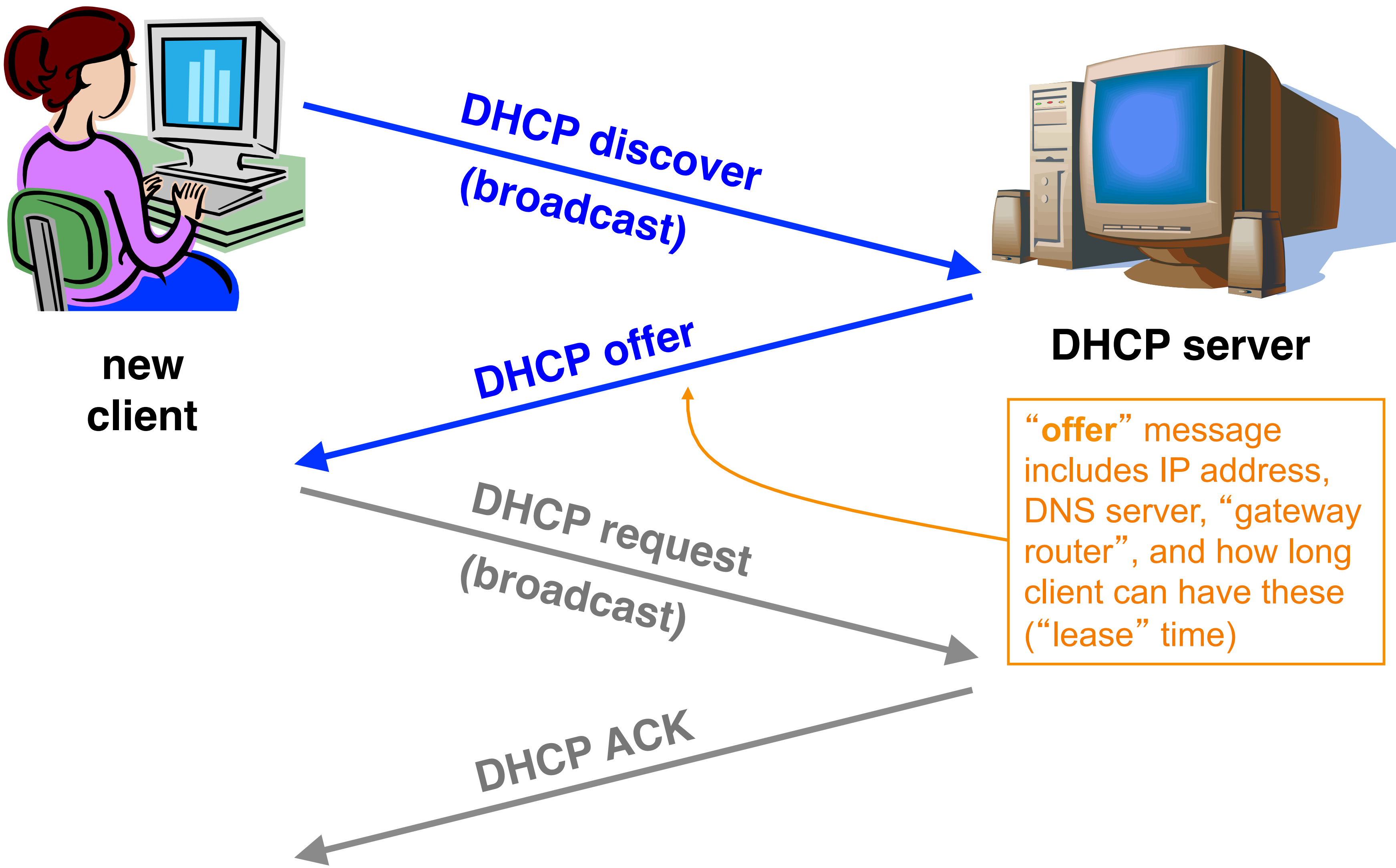
- Can set arbitrary source address
  - "Spoofing" - receiver has no idea who you are
  - Could be blind, or could be coupled w/ sniffing
  - Note: many attacks require two-way communication
    - So successful off-path/blind spoofing might not suffice
- Can set arbitrary destination address
  - Enables "scanning" - brute force searching for hosts
- Can *send like crazy* (flooding)
  - IP has no general mechanism for tracking overuse
  - IP has no general mechanism for tracking consent
  - Very hard to tell where a spoofed flood comes from!
- **If** attacker can manipulate routing, can bring traffic to themselves for *eavesdropping* (not easy)

LAN Bootstrapping: DHCP

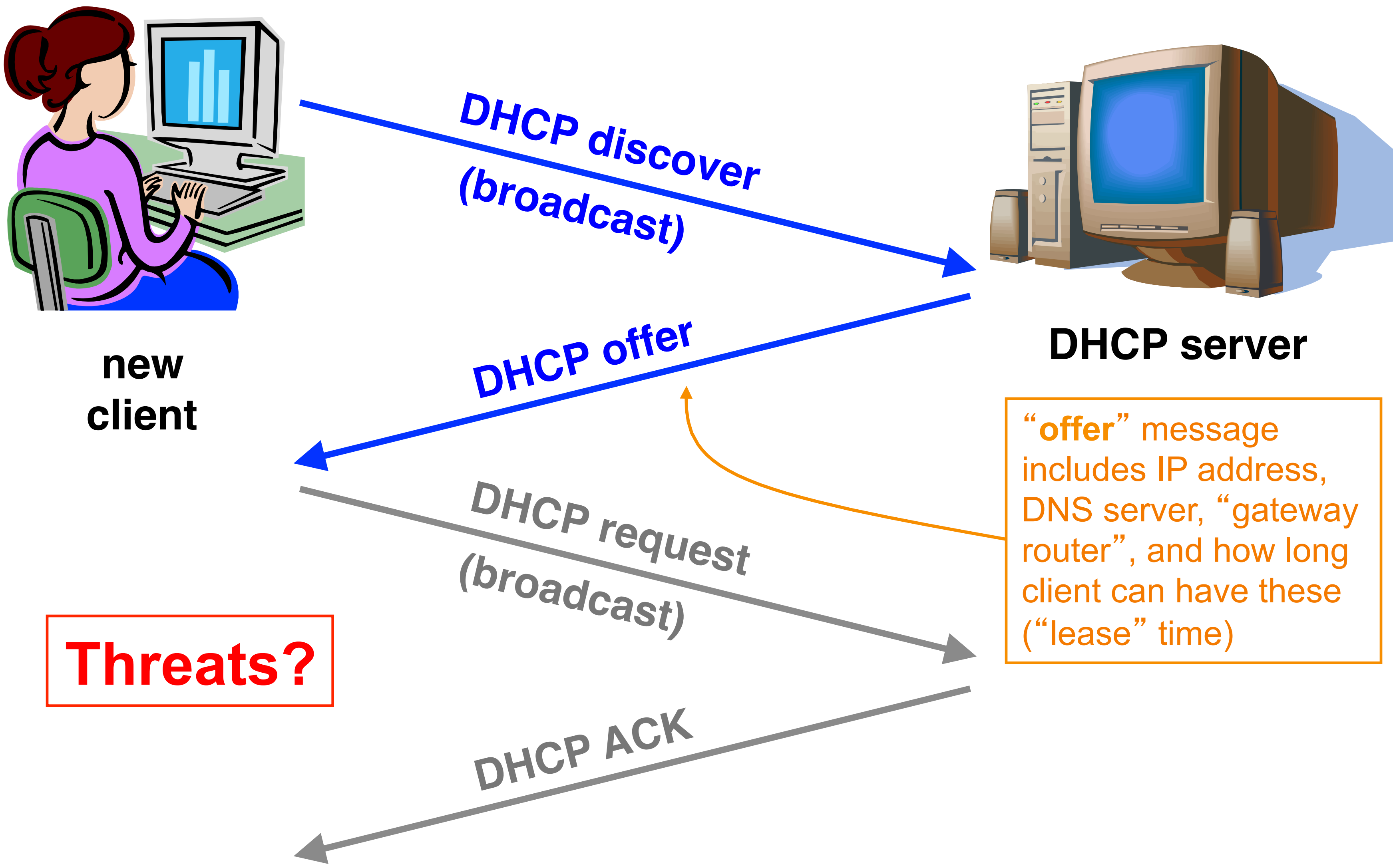
- New host doesn't have an IP address yet
  - So, host doesn't know what source address to use
- Host doesn't know *who to ask* for an IP address
  - So, host doesn't know what destination address to use
- Solution: shout to "**discover**" server that can help
  - Broadcast a server-discovery message (layer 2)
  - Server(s) sends a reply offering an address



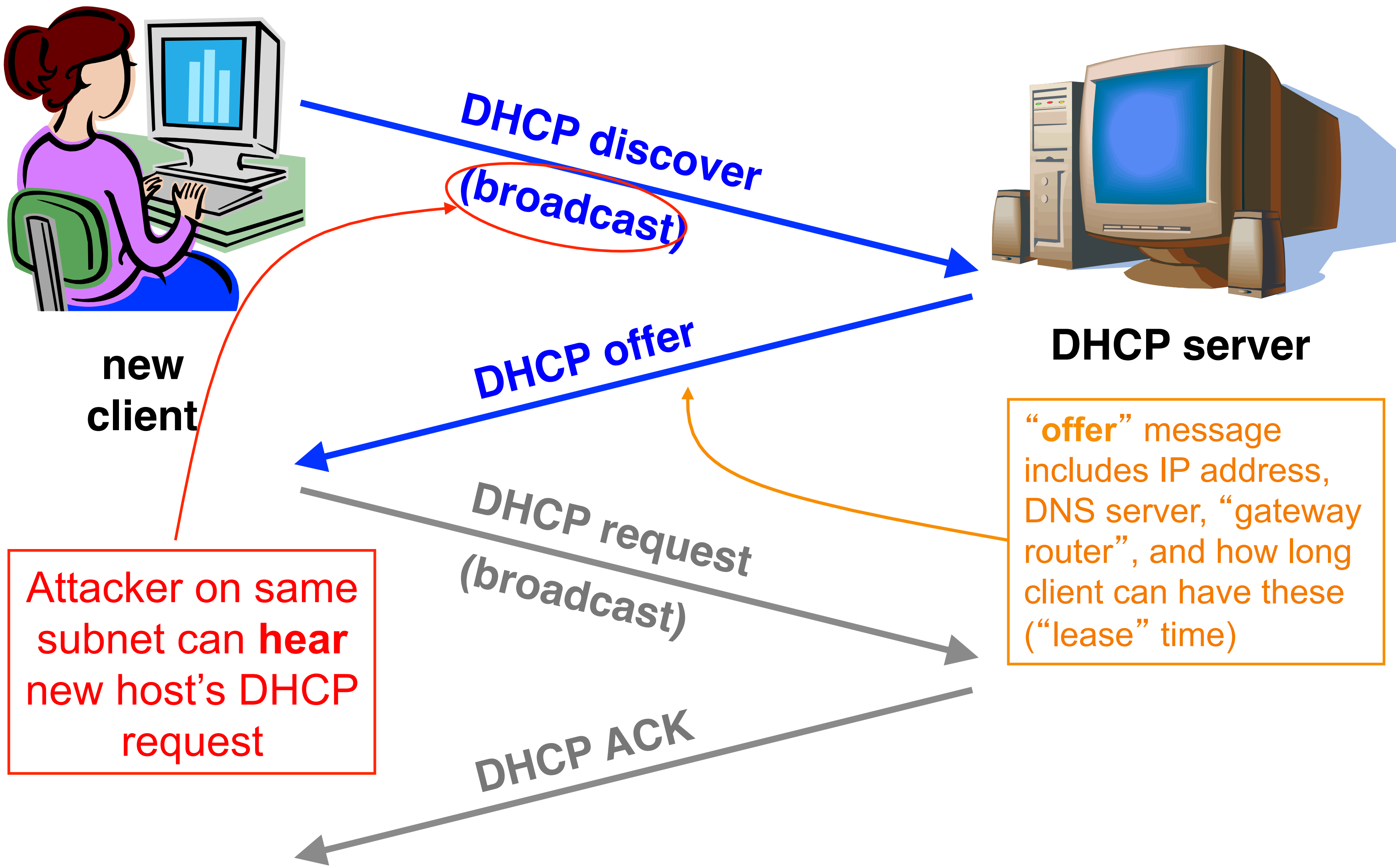
Dynamic Host Configuration Protocol



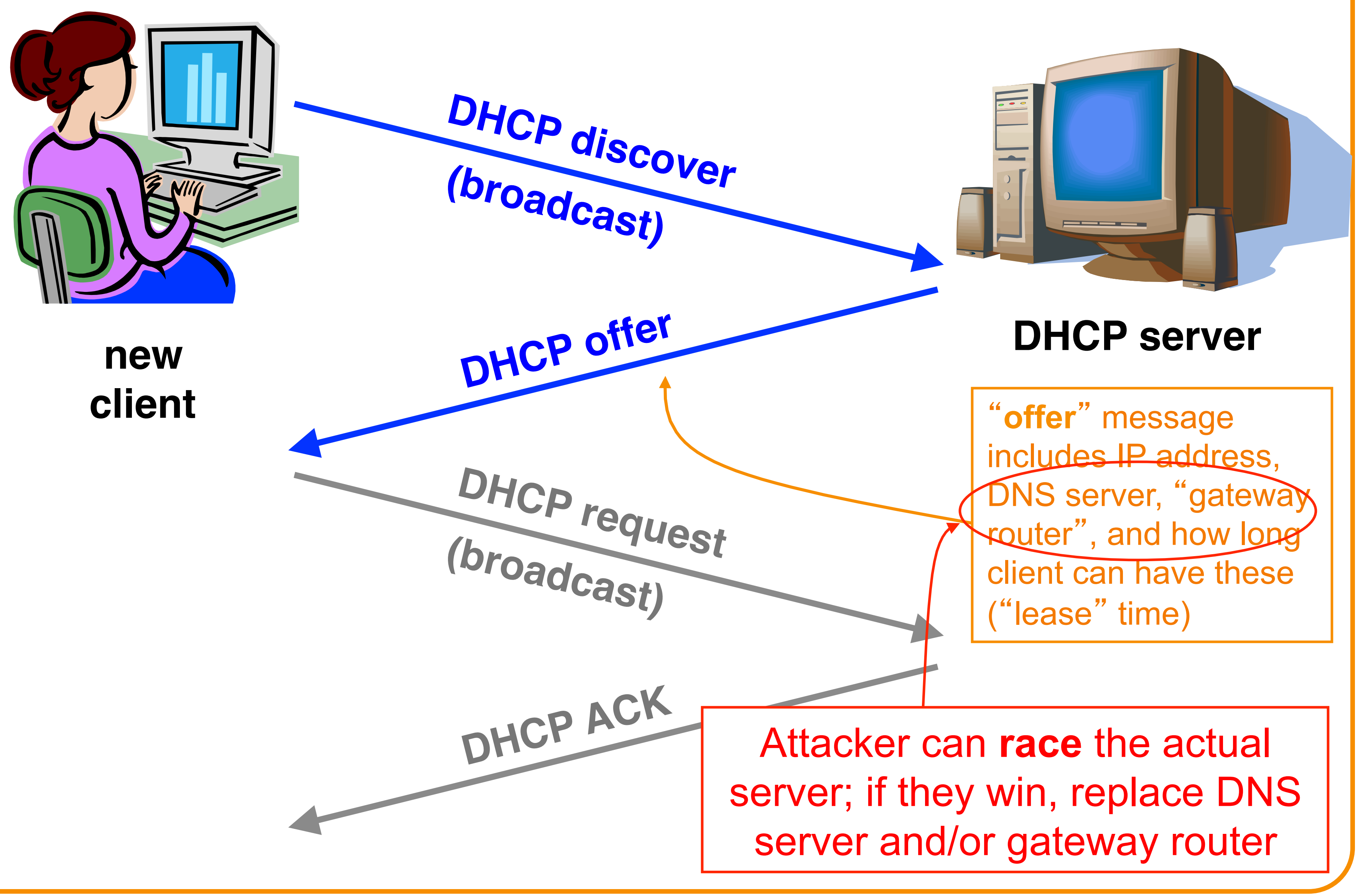
Dynamic Host Configuration Protocol



Dynamic Host Configuration Protocol



Dynamic Host Configuration Protocol

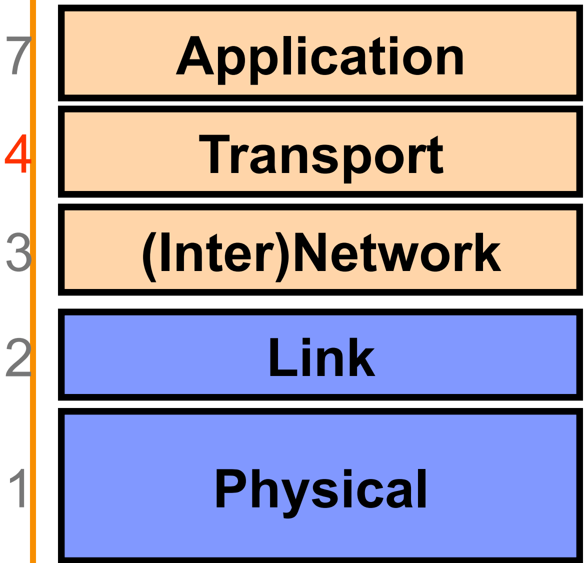


DHCP Threats

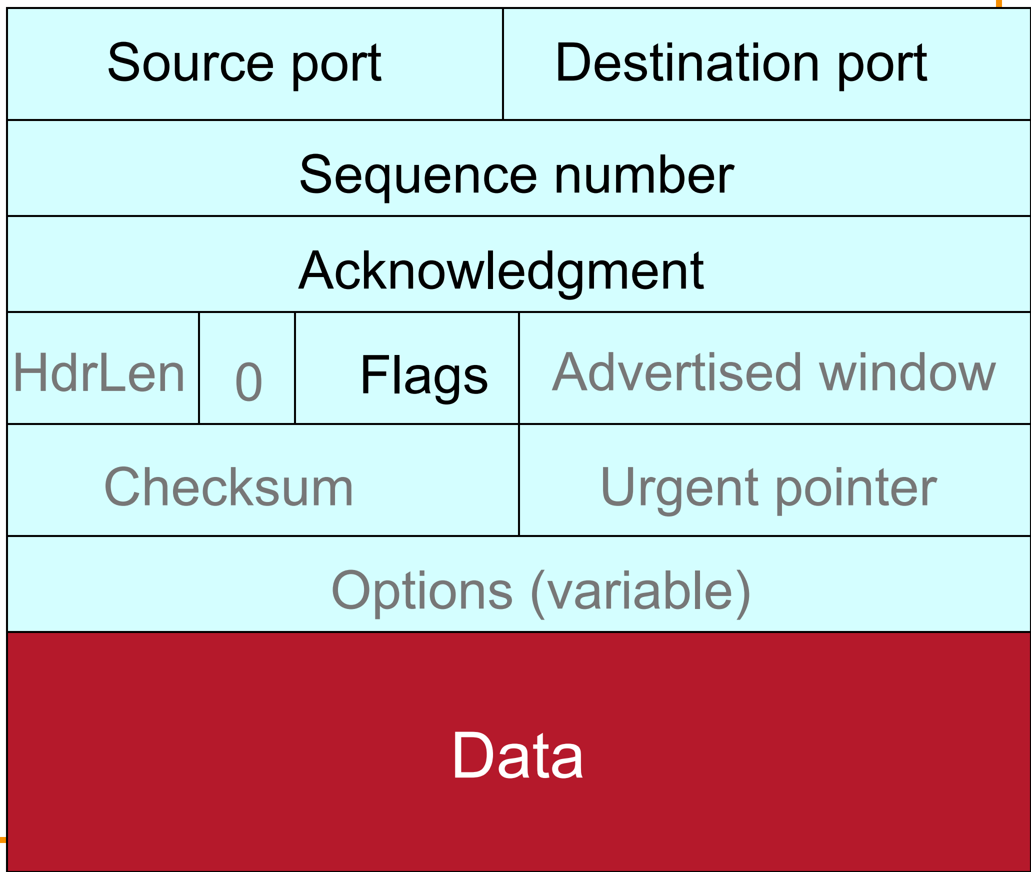
- Substitute a fake DNS server
  - Redirect any of a host's lookups to a machine of attacker's choice
- Substitute a fake gateway router
  - Intercept all of a host's off-subnet traffic
    - (even if not preceded by a DNS lookup)
  - Relay contents back and forth between host and remote server and modify however attacker chooses
- An invisible *Man In The Middle* (MITM)
  - Victim host has no way of knowing it's happening
    - (Can't necessarily alarm on peculiarity of receiving multiple DHCP replies, since that can happen benignly)
- How can we fix this?

Hard

TCP

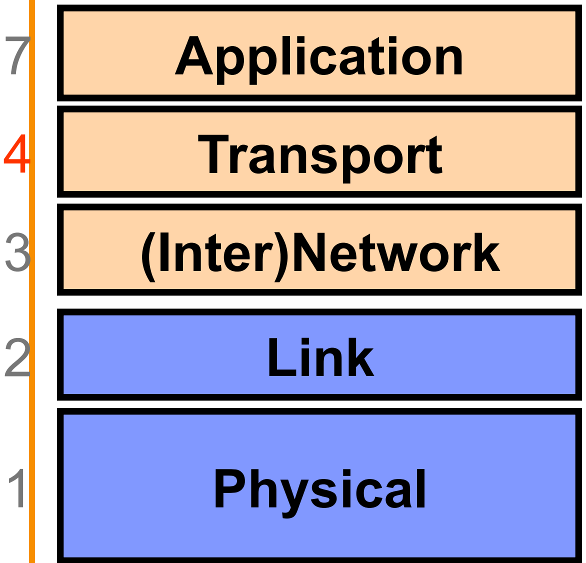


7 Application
4 Transport
3 (Inter)Network
2 Link
1 Physical

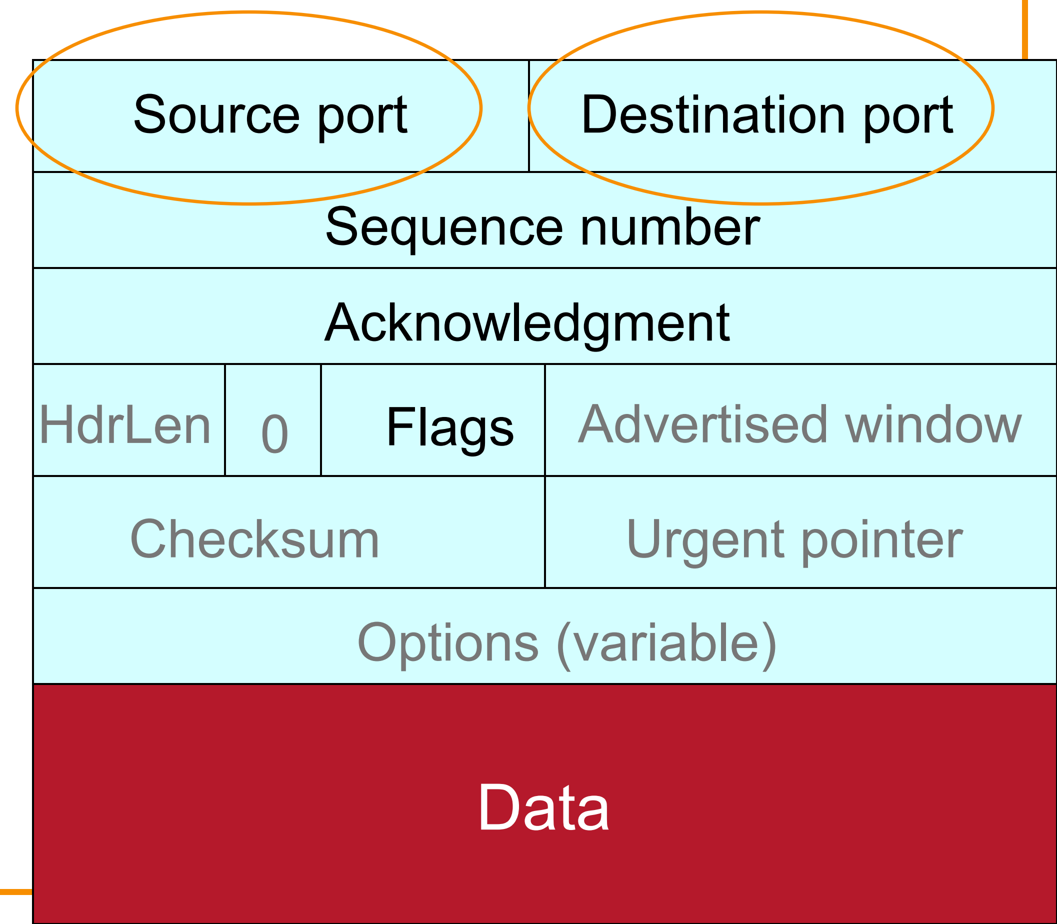


Source port
Destination port
Sequence number
Acknowledgment
HdrLen
0
Flags
Advertised window
Checksum
Urgent pointer
Options (variable)
Data

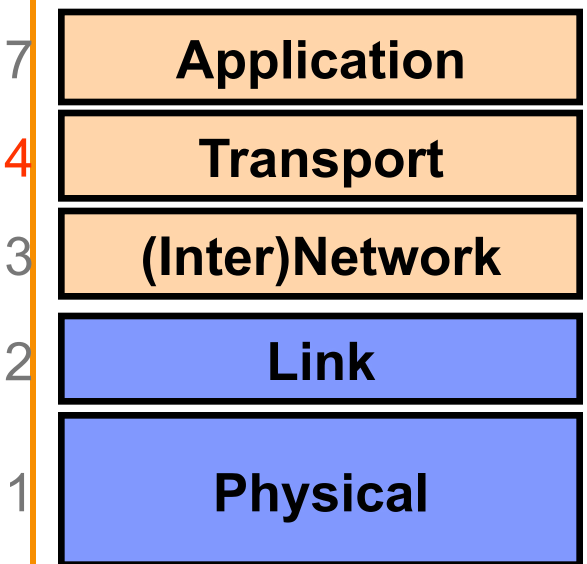
TCP



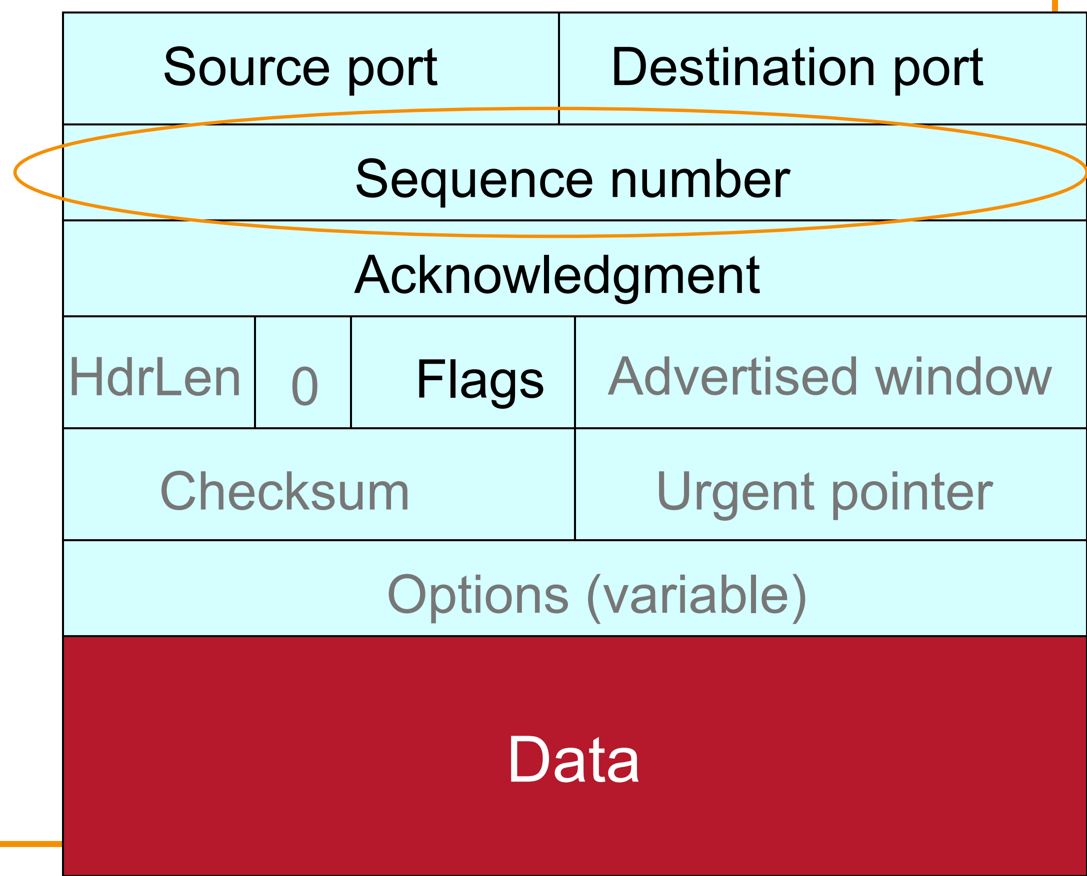
These plus IP addresses define a given connection



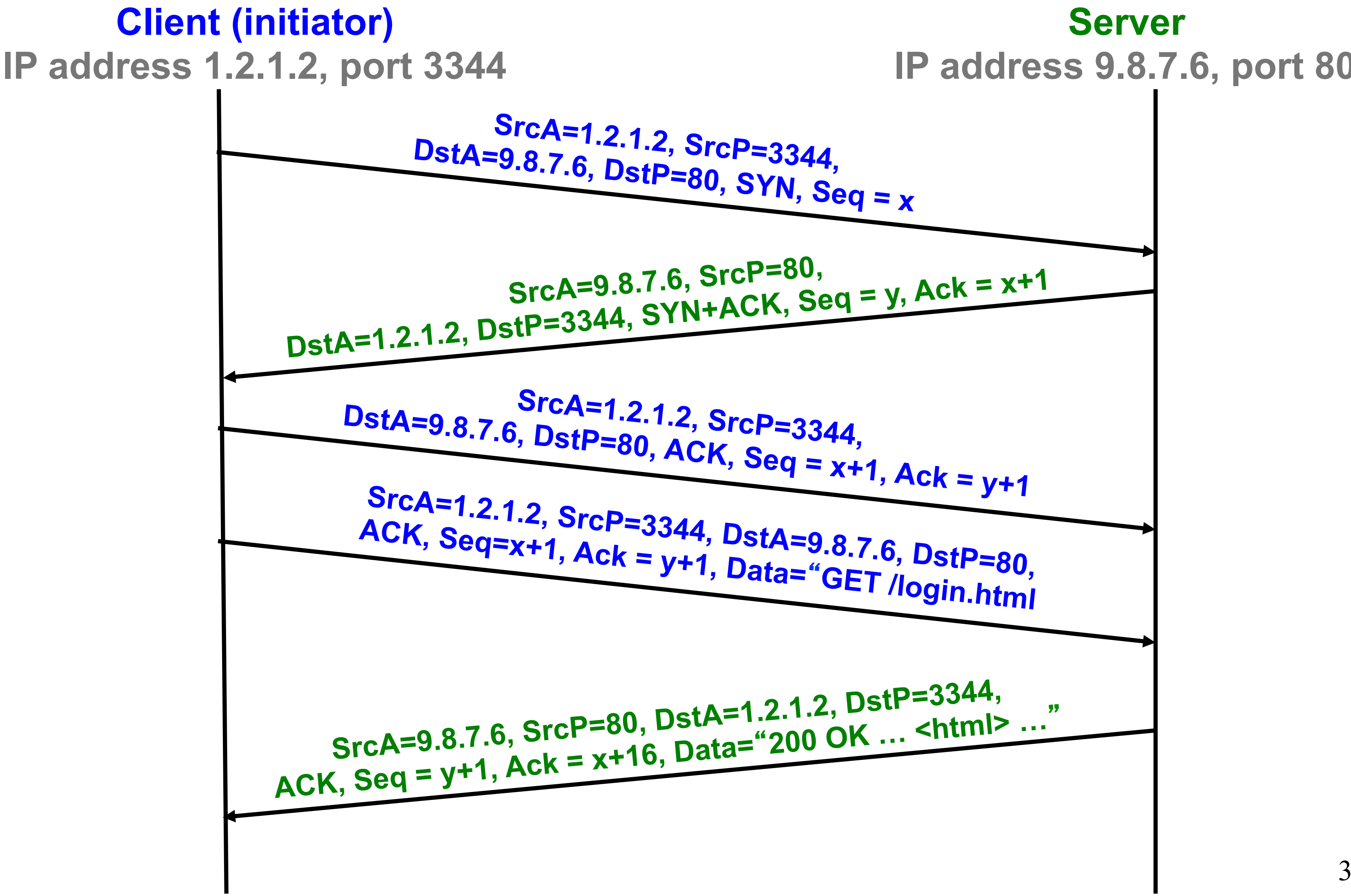
TCP



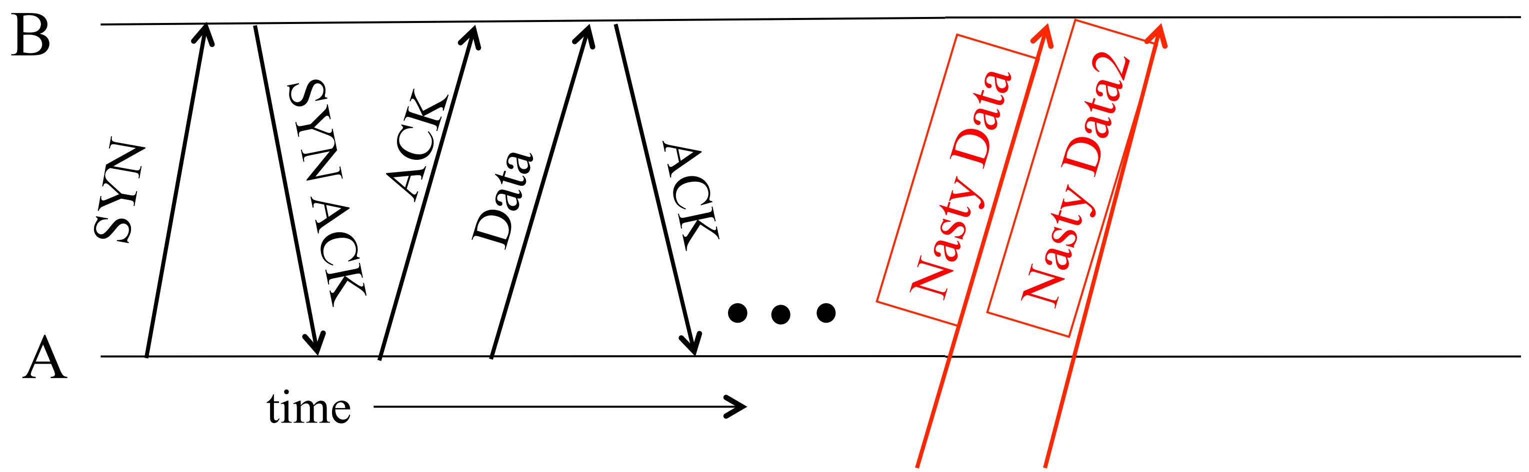
Defines where this packet fits within the sender's bytestream



TCP Conn. Setup & Data Exchange

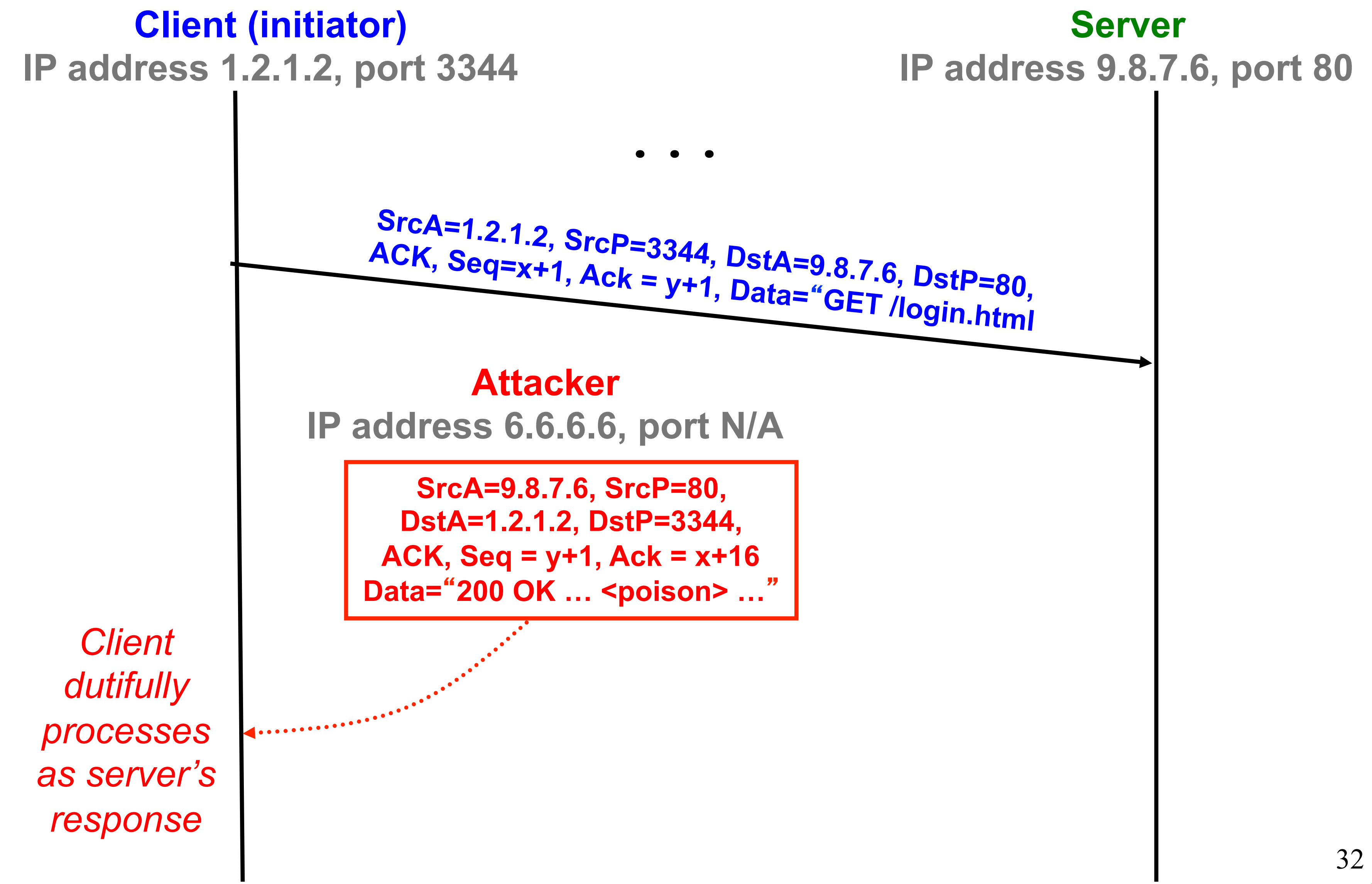


TCP Threat: Data Injection

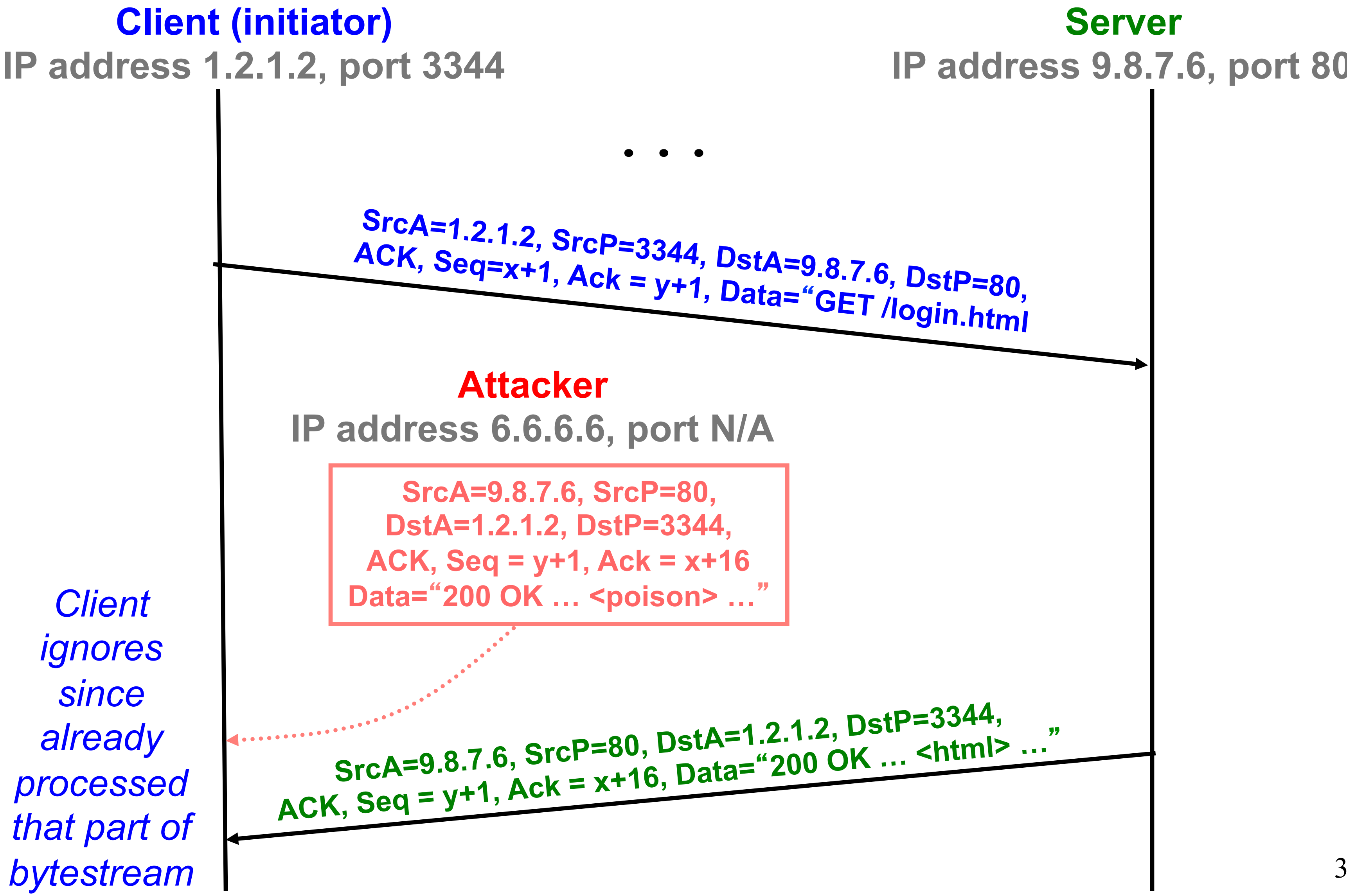


- If attacker knows ports & sequence numbers (e.g., on-path attacker), attacker can inject data into any TCP connection
  - Receiver B is *none the wiser!*
- Termed TCP connection hijacking (or "*session hijacking*")
  - A general means to take over an already-established connection!
- We are toast if an attacker can see our TCP traffic!
  - Because then they immediately know the port & sequence numbers

TCP *Data Injection*



TCP *Data Injection*



TCP Threat: Disruption

- Is it possible for an on-path attacker to shut down a TCP connection if they can see our traffic?
- YES: they can infer the port and sequence numbers – they can insert fake data, too! (Great Firewall of China)

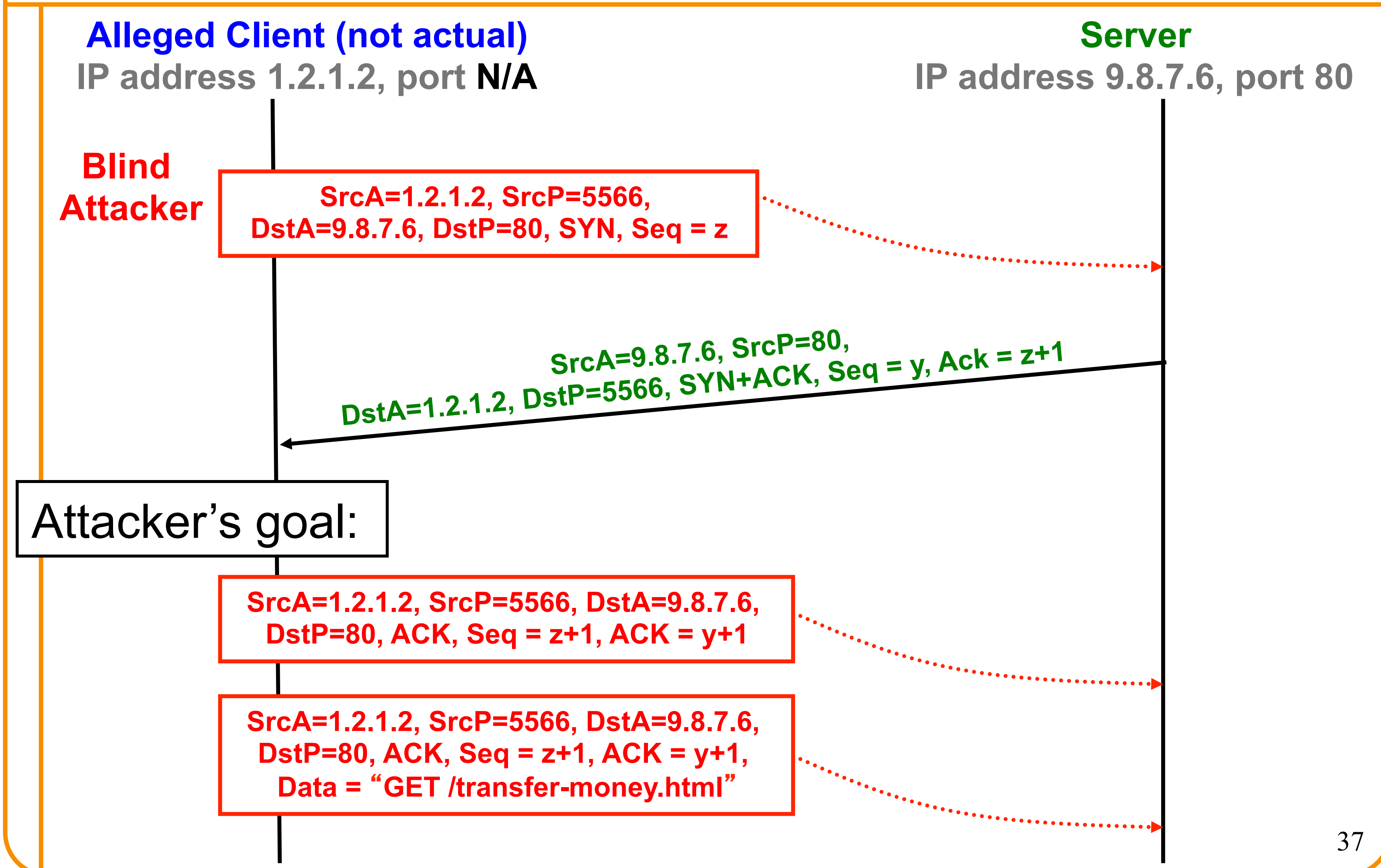
TCP Threat: Blind Hijacking

- Is it possible for an off-path attacker to inject into a TCP connection even if they can't see our traffic?
- YES: if somehow they can infer or guess the port and sequence numbers

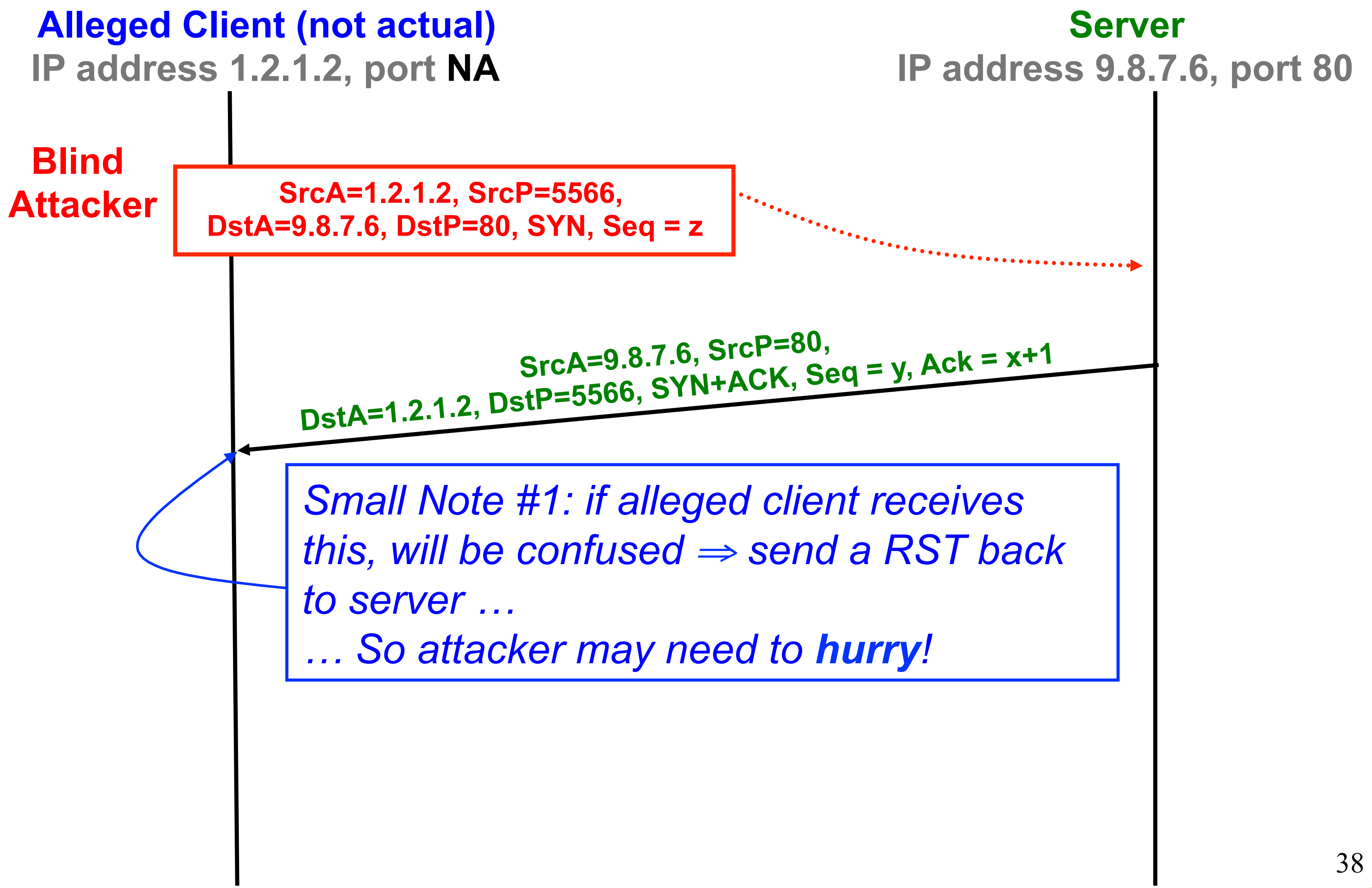
TCP Threat: Blind Spoofing

- Is it possible for an off-path attacker to create a fake TCP connection, even if they can't see responses?
- YES: if somehow they can infer or guess the TCP initial sequence numbers
- Why would an attacker want to do this?
  - Perhaps to leverage a server's trust of a given client as identified by its IP address
  - Perhaps to frame a given client so the attacker's actions during the connections can't be traced back to the attacker

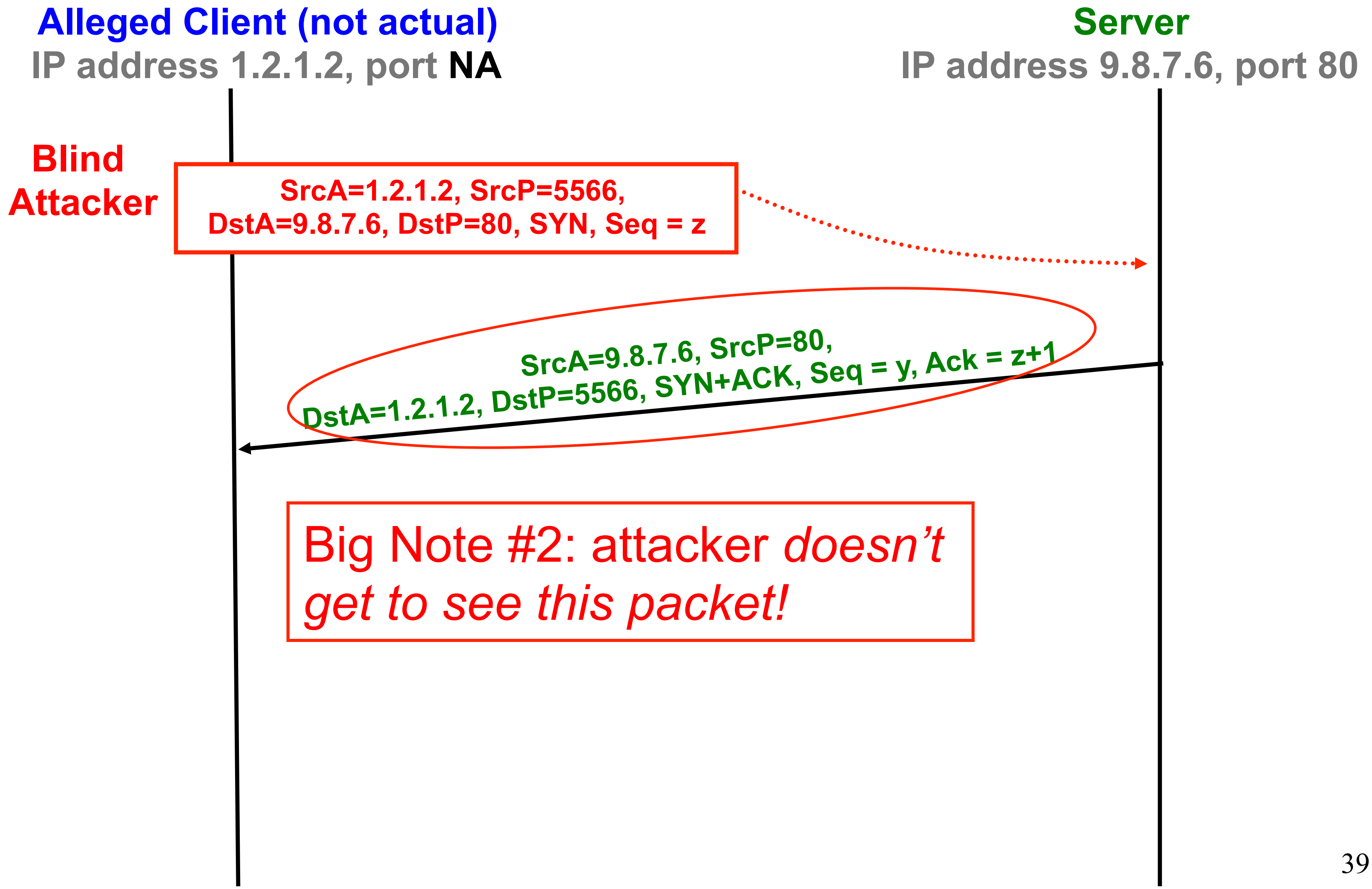
Blind Spoofing on TCP Handshake



Blind Spoofing on TCP Handshake

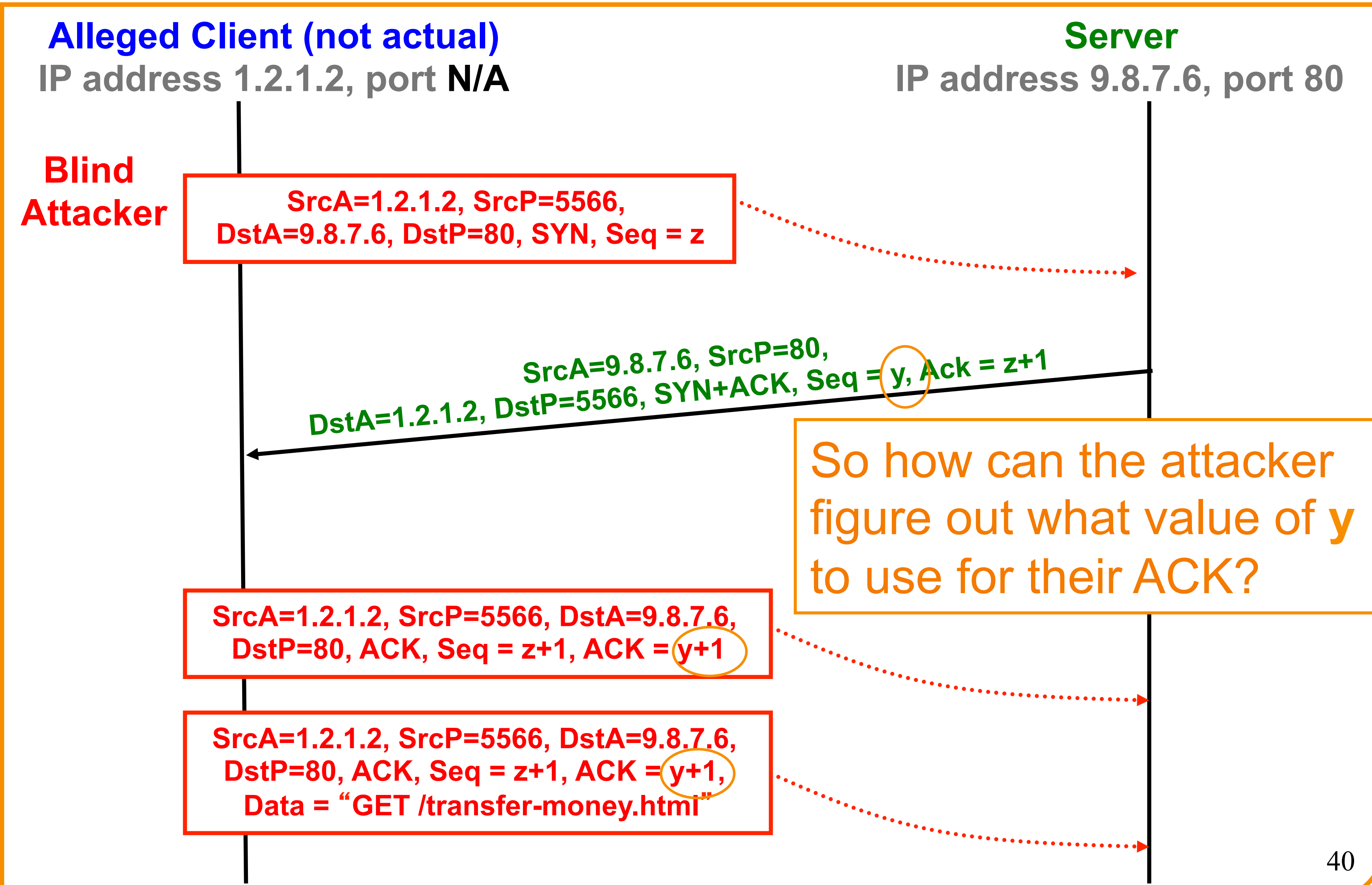


Blind Spoofing on TCP Handshake

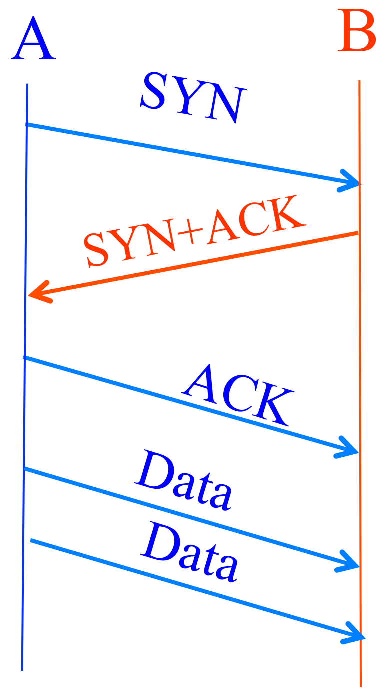


Big Note #2: attacker *doesn't get to see this packet!*

Blind Spoofing on TCP Handshake



Reminder: Establishing a TCP Connection



Each host tells its *Initial Sequence Number* (ISN) to the other host.

(Spec says to pick based on local clock)

Hmm, any way for the attacker to know ***this?***

How Do We Fix This?

Use a (Pseudo)-*Random* ISN

Sure - make a non-spoofed connection *first*, and see what server used for ISN y then!

Summary of TCP Security Issues

- An attacker who can observe your TCP connection can manipulate it:
  - Forcefully **terminate** by forging a RST packet
  - **Inject** (*spoof*) data into either direction by forging data packets
  - Works because they can include in their spoofed traffic the correct sequence numbers (both directions) and TCP ports
  - *Remains a major threat today*

Summary of TCP Security Issues

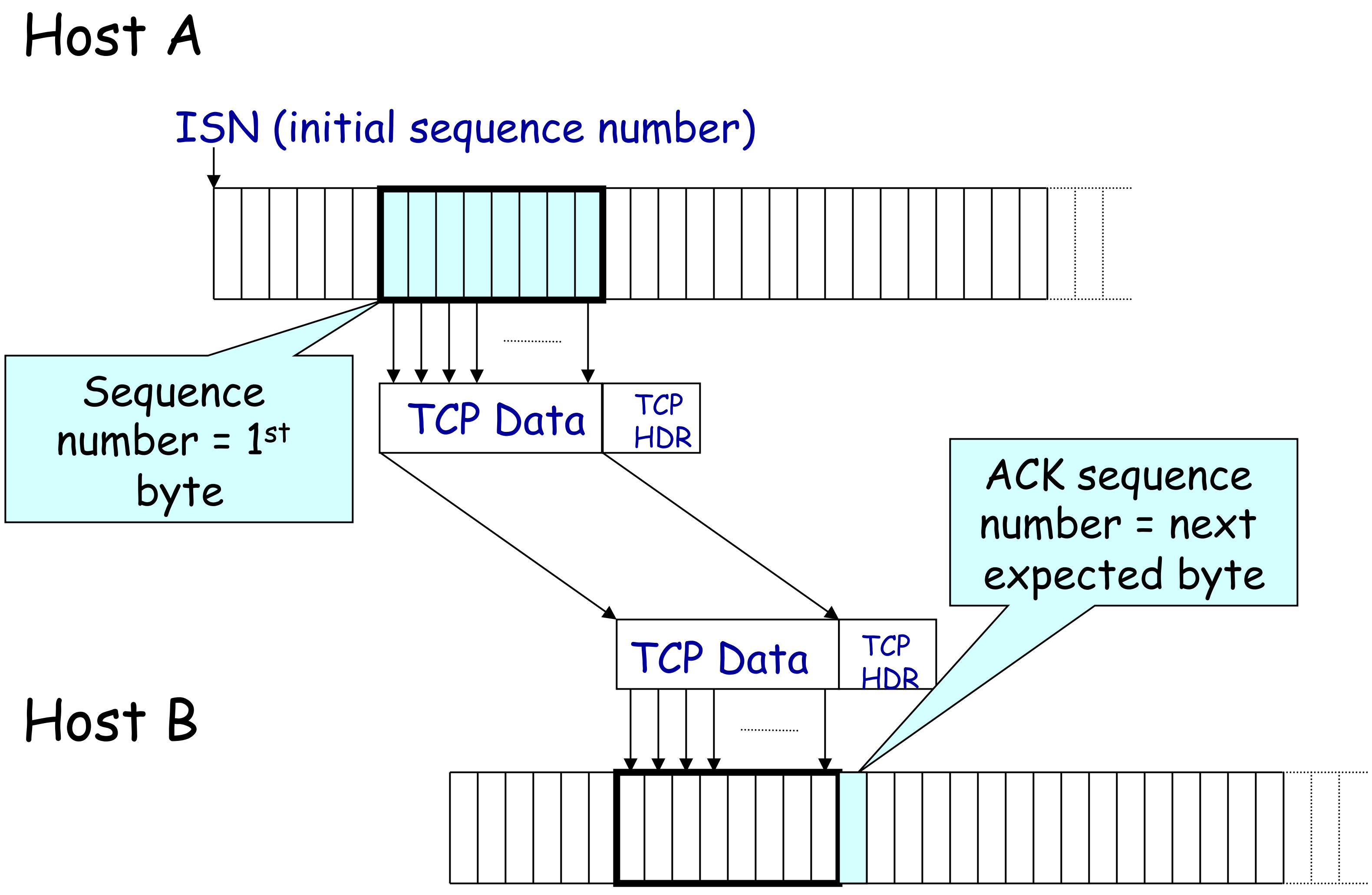
- An attacker who can observe your TCP connection can manipulate it:
  - Forcefully **terminate** by forging a RST packet
  - **Inject** (*spoof*) data into either direction by forging data packets
  - Works because they can include in their spoofed traffic the correct sequence numbers (both directions) and TCP ports
  - *Remains a major threat today*
- If attacker could predict the ISN chosen by a server, could "blind spoof" a connection to the server
  - Makes it appear that host ABC has connected, and has sent data of the attacker's choosing, when in fact it hasn't
  - *Undermines any security based on trusting ABC's IP address*
  - Allows attacker to "frame" ABC or otherwise avoid detection
  - **Fixed** (mostly) today by choosing **random** ISNs

Summary of IP security

- No security against on-path attackers
  - Can sniff, inject packets, mount TCP spoofing, TCP hijacking, man-in-the-middle attacks
  - Typical example: wireless networks, malicious network operator
- Reasonable security against off-path attackers
  - TCP is basically secure, but UDP and IP are not

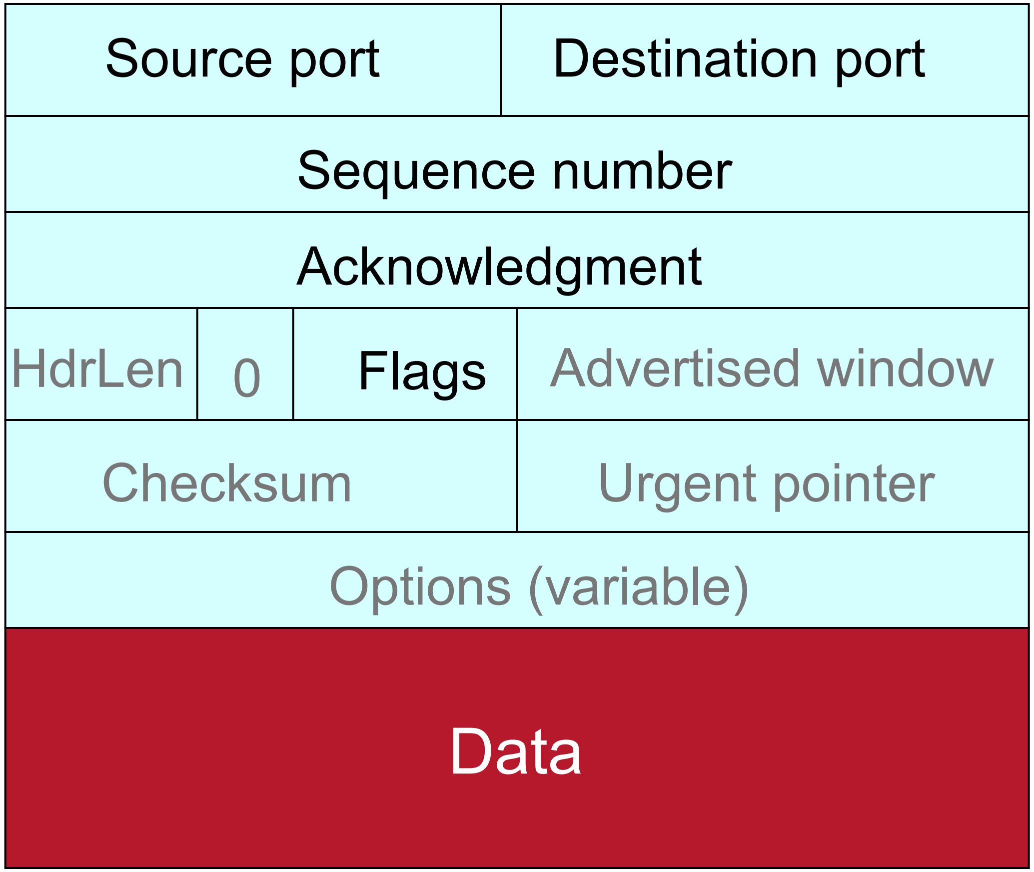
Extra Material

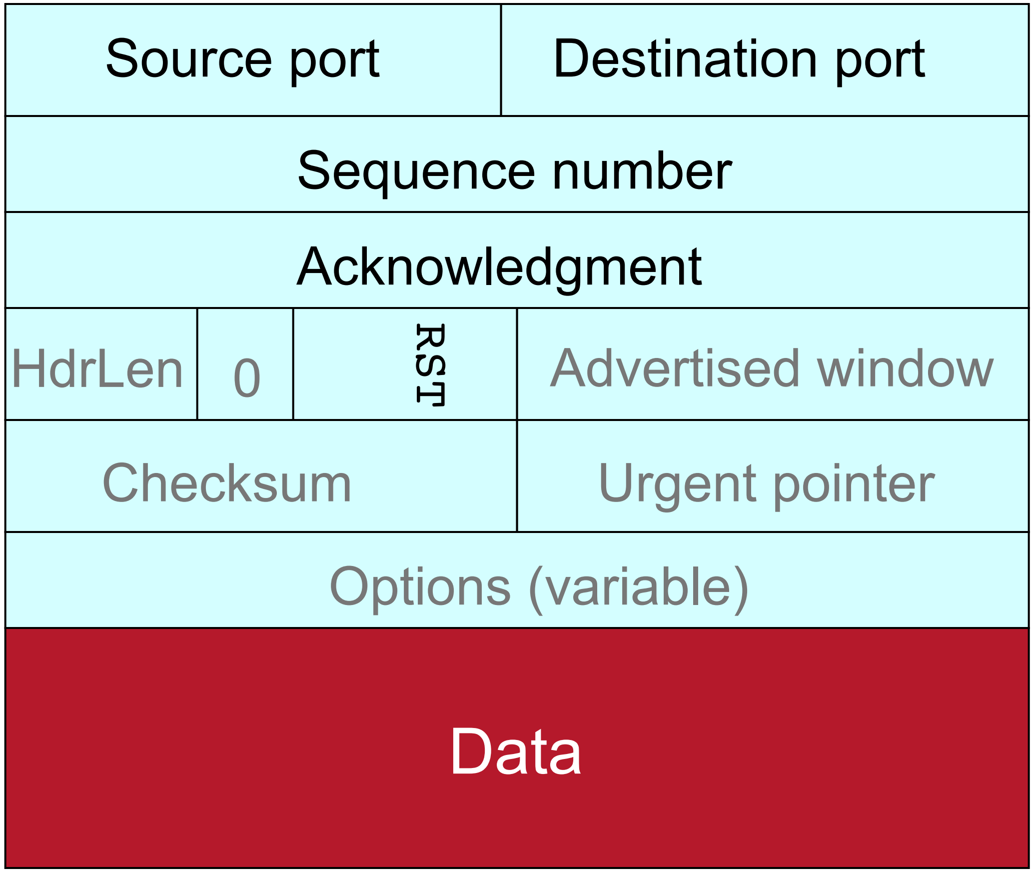
Sequence Numbers



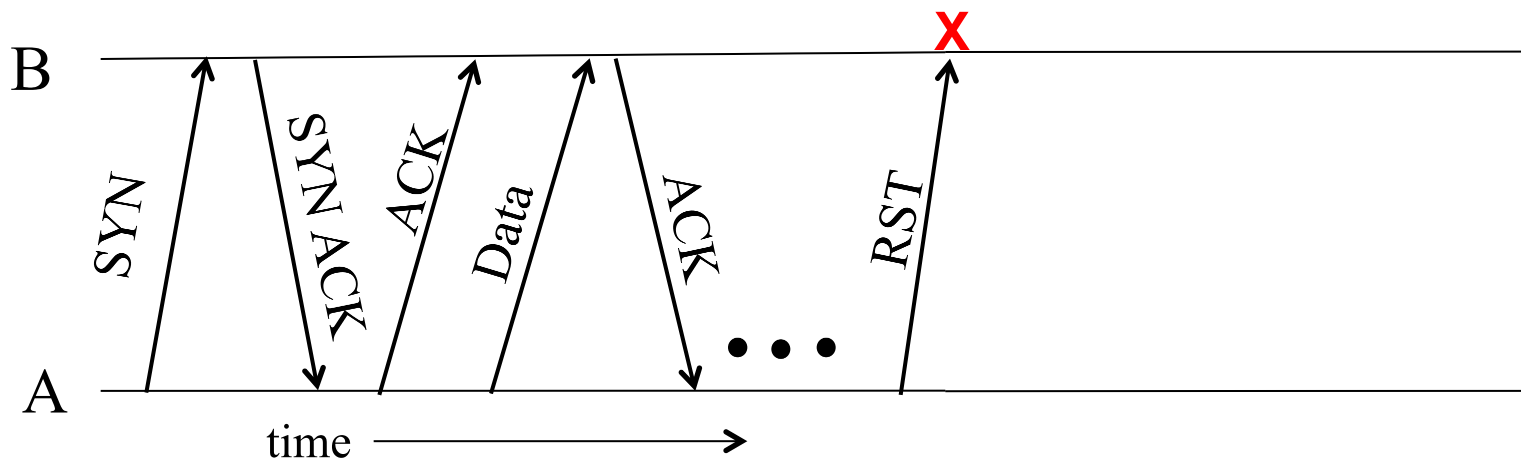
TCP Threat: Disruption

- Normally, TCP finishes ("closes") a connection by each side sending a `FIN` control message
  - Reliably delivered, since other side must <u>ack</u>
- But: if a TCP endpoint finds unable to continue (process dies; info from other "peer" is inconsistent), it abruptly terminates by sending a **RST** control message
  - Unilateral
  - Takes effect immediately (no ack needed)
  - Only accepted by peer if has correct* sequence number


Source port
Destination port
Sequence number
Acknowledgment
HdrLen
0
Flags
Advertised window
Checksum
Urgent pointer
Options (variable)
Data


Source port
Destination port
Sequence number
Acknowledgment
HdrLen
0
RST
Advertised window
Checksum
Urgent pointer
Options (variable)
Data

Abrupt Termination

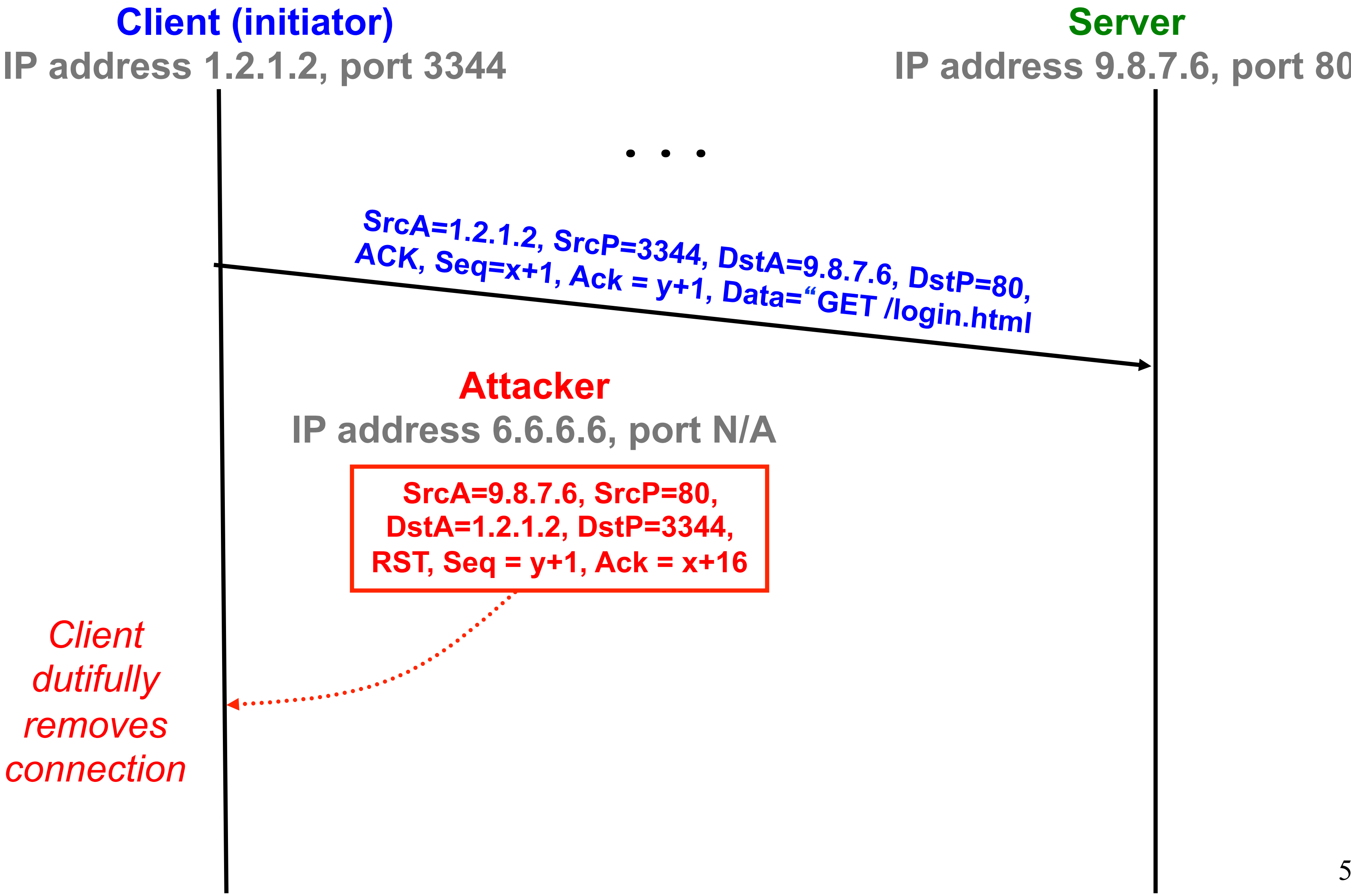


- A sends a TCP packet with RESET (**RST**) flag to B
  - E.g., because app. process on A crashed
  - (Could instead be that B sends a RST to A)
- Assuming that the sequence numbers in the **RST** fit with what B expects, That's It:
  - B's user-level process receives: `ECONNRESET`
  - No further communication on connection is possible

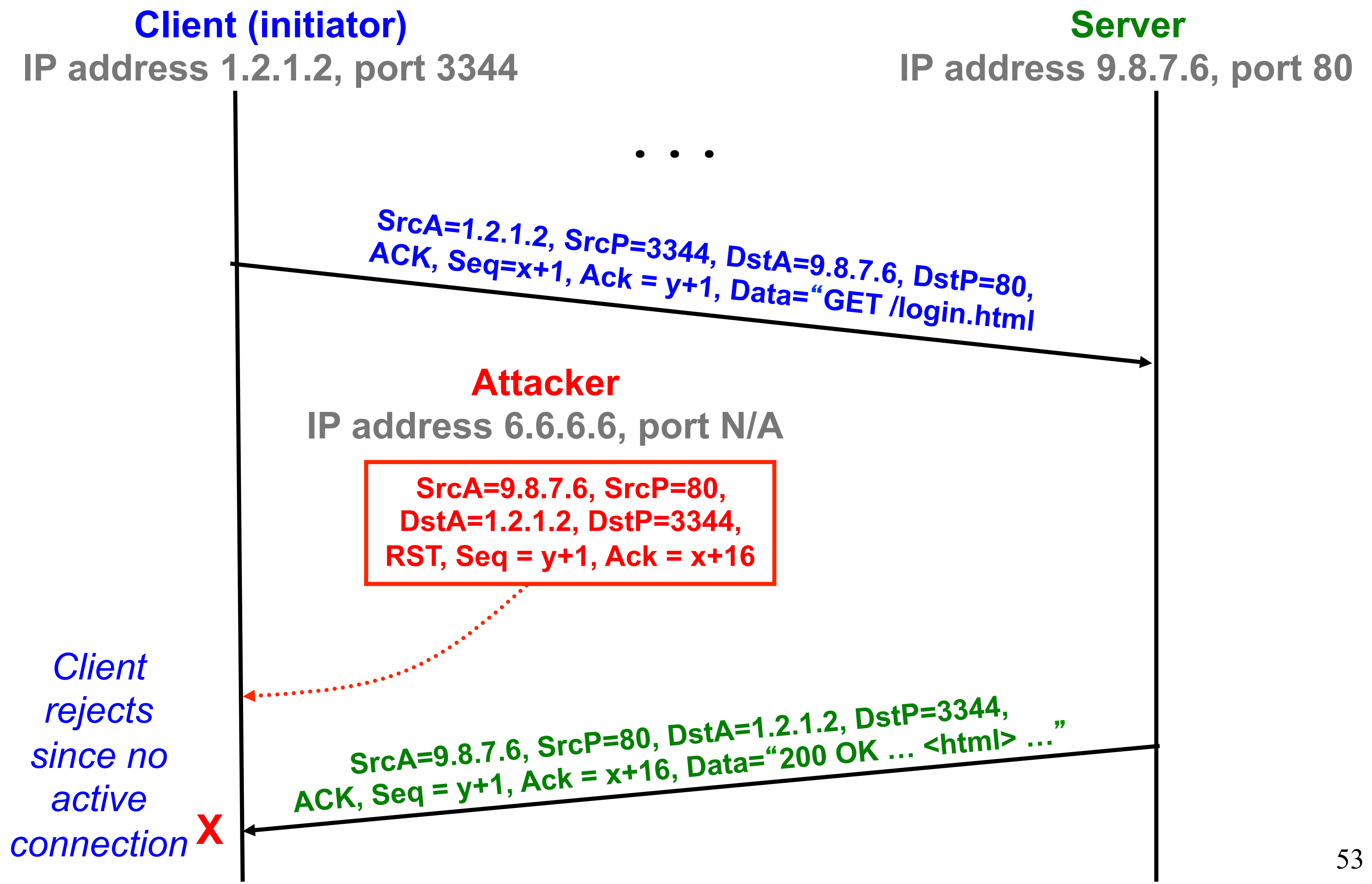
TCP Threat: Disruption

- Normally, TCP finishes ("closes") a connection by each side sending a `FIN` control message
  - Reliably delivered, since other side must ack
- But: if a TCP endpoint finds unable to continue (process dies; info from other "peer" is inconsistent), it abruptly terminates by sending a `RST` control message
  - Unilateral
  - Takes effect immediately (no ack needed)
  - Only accepted by peer if has correct* sequence number
- So: if attacker knows ports & sequence numbers, can disrupt any TCP connection

TCP *RST Injection*



TCP *RST Injection*



Threats to Comm. Security Goals

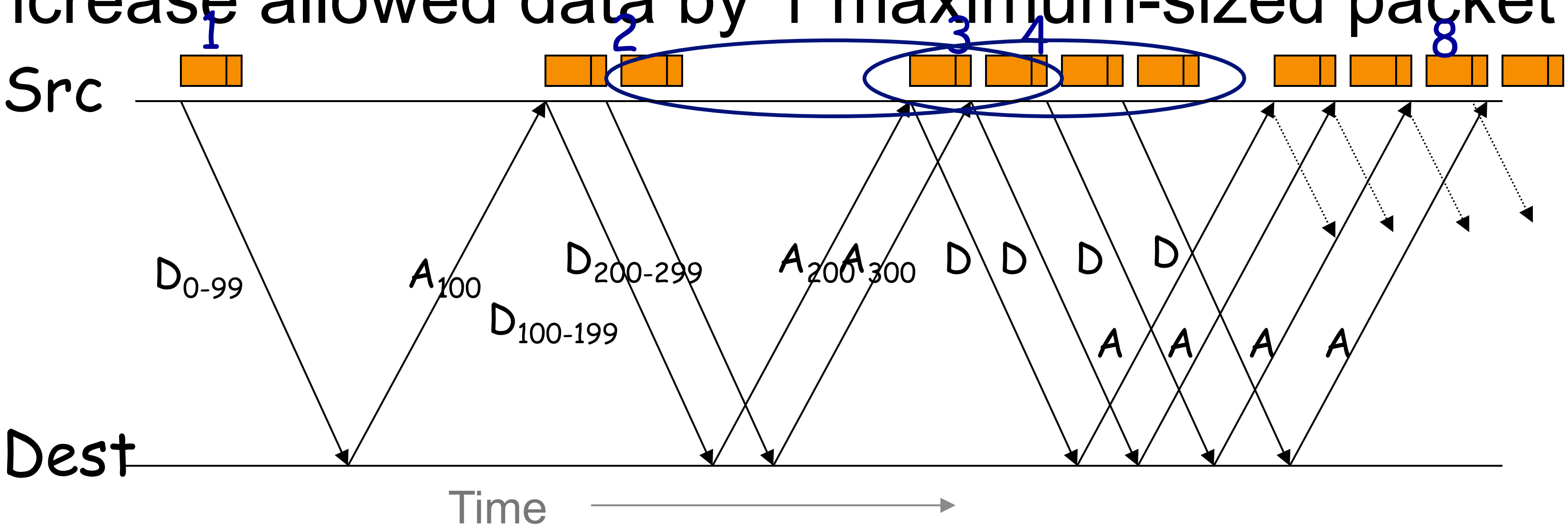
- Attacks can subvert each type of goal
  - Confidentiality: eavesdropping / theft of information
  - Integrity: altering data, manipulating execution (e.g., code injection)
  - Availability: *denial-of-service*

- Attackers can also combine different types of attacks towards an overarching goal
  - E.g. use eavesdropping (confidentiality) to construct a spoofing attack (integrity) that tells a server to drop an important connection (denial-of-service)

TCP's Rate Management

Unless there's loss, TCP doubles data in flight every "round-trip". All TCPs expected to obey ("fairness").

Mechanism: for each arriving ack for new data, increase allowed data by 1 maximum-sized packet

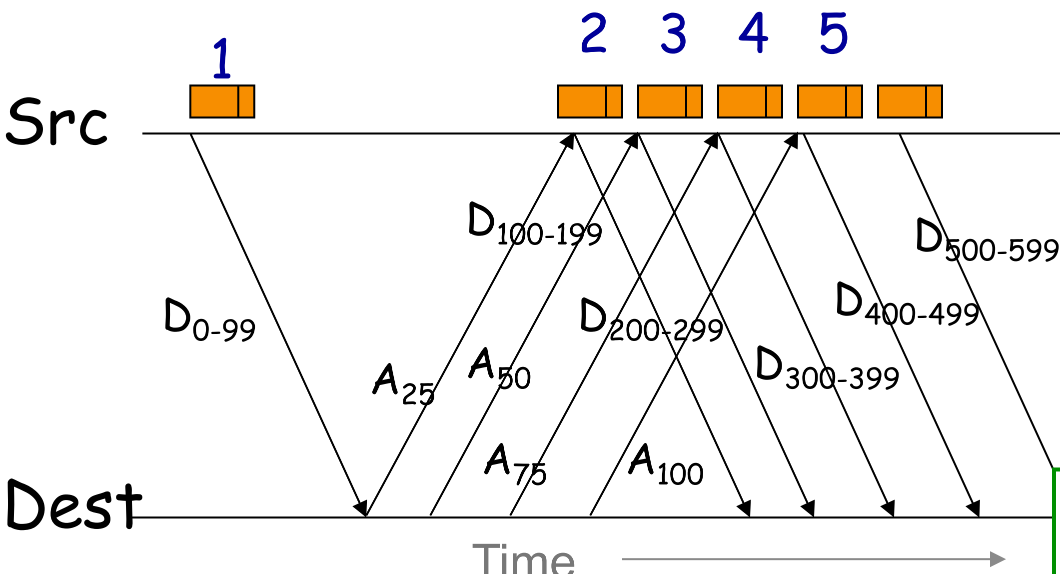


E.g., suppose maximum-sized packet = 100 bytes

Protocol Cheating

How can the destination (receiver) get data to come to them faster than normally allowed?

ACK-Splitting: each ack, even though **partial**, increases allowed data by one maximum-sized packet



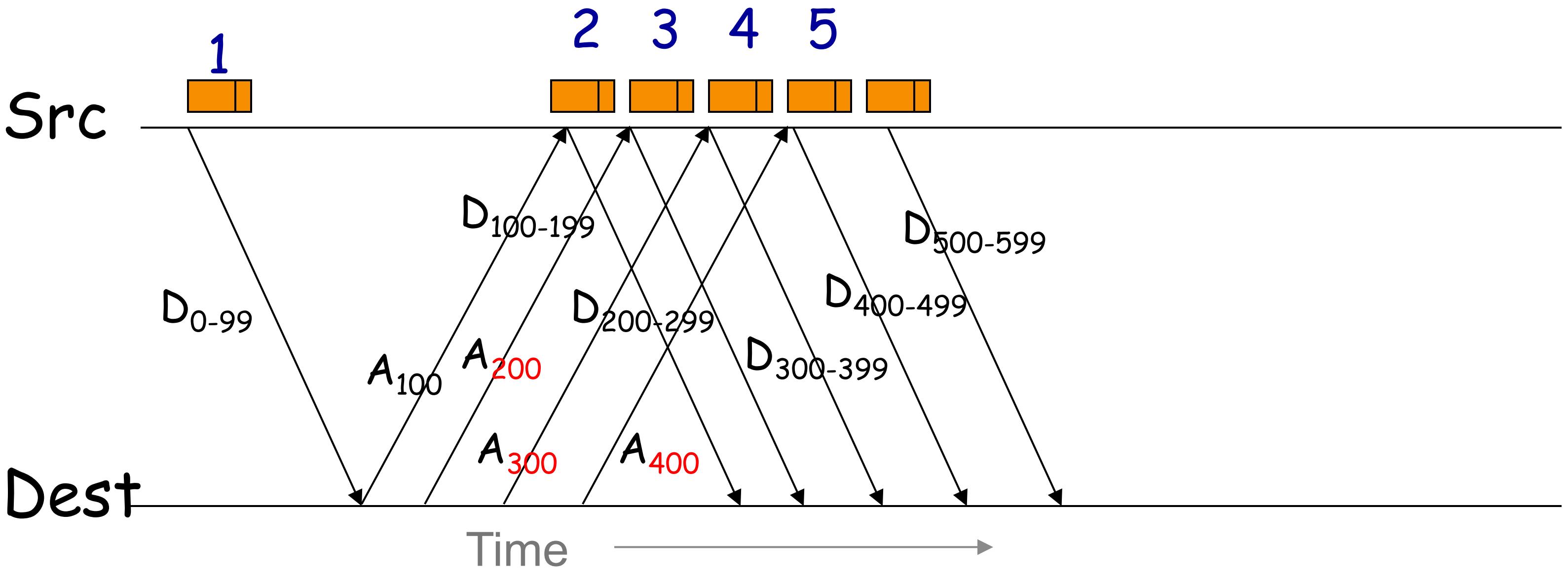
How do we defend against this?

Change rule to require "full" ack for *all* data sent in a packet

Protocol Cheating

How can the destination (receiver) *still* get data to come to them faster than normally allowed?

Opportunistic ack'ing: acknowledge data not yet seen!



How do we defend against *this*?

Keeping Receivers Honest

- Approach #1: if you receive an ack for data you haven't sent, kill the connection
  - Works only if receiver acks too far ahead
- Approach #2: follow the "round trip time" (RTT) and if ack arrives too quickly, kill the connection
  - Flaky: RTT can vary a lot, so you might kill innocent connections
- Approach #3: make the receiver prove they received the data

Note: a *protocol* change

  - Add a **nonce** ("random" marker) & require receiver to include it in ack. Kill connections w/ incorrect nonces
    - (nonce could be function computed over payload, so sender doesn't explicitly transmit, only implicitly)