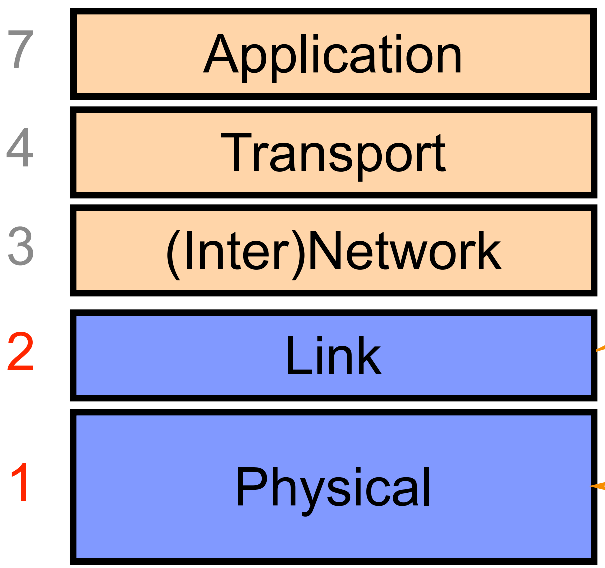
Network Attacks

CS 334 - Computer Security

Once again thanks to Vern Paxson and David Wagner

Layers 1 & 2: General Threats?



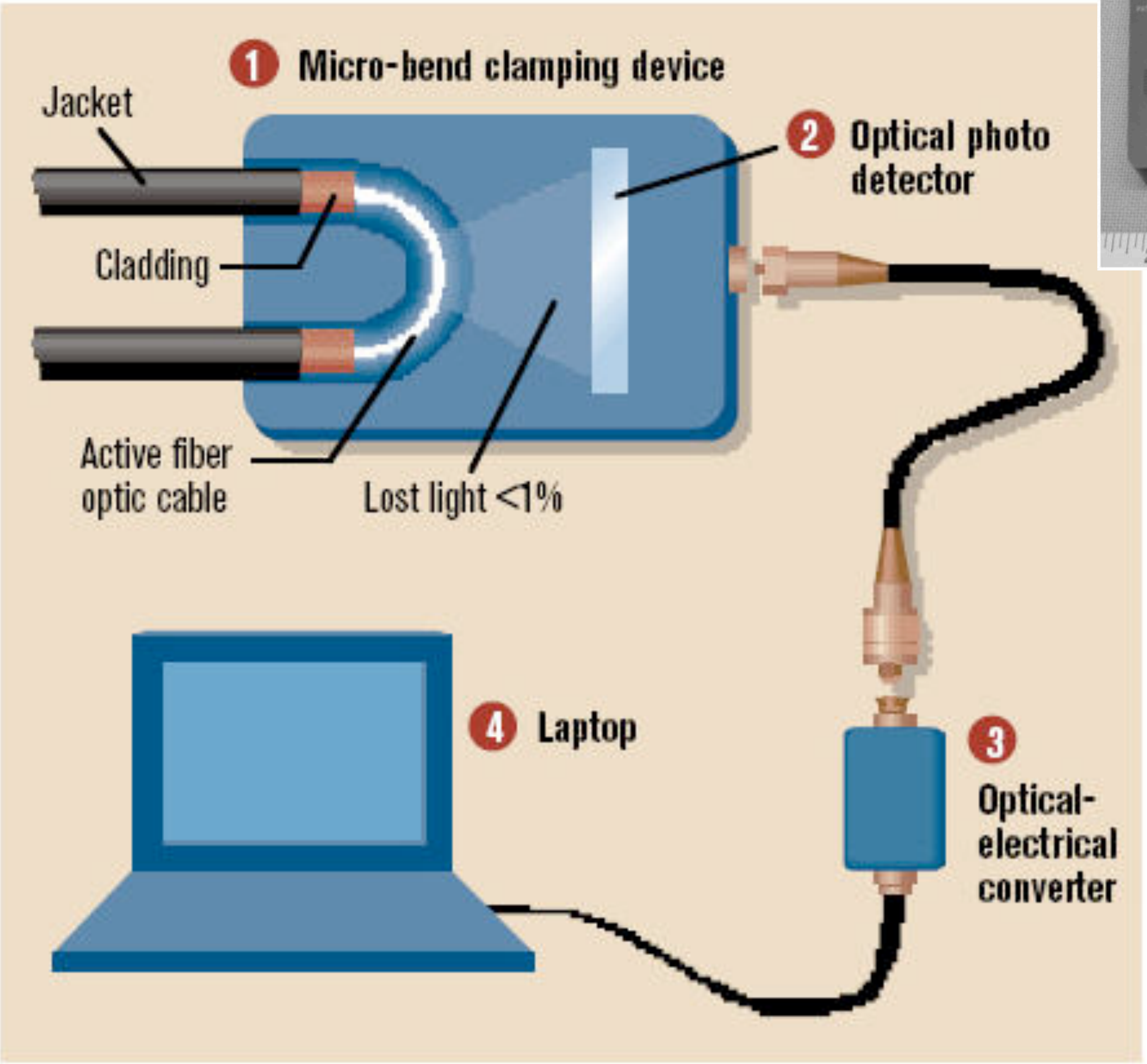
Framing and transmission of a collection of bits into individual messages sent across a single “subnetwork” (one physical technology)

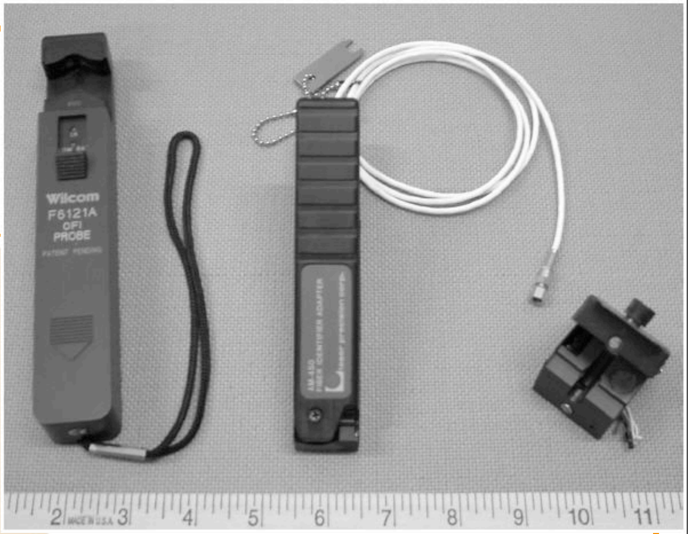
Encoding bits to send them over a single physical link
e.g. patterns of
voltage levels / photon intensities / RF modulation

Physical/Link-Layer Threats: *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
    - Wireshark
    - tcpdump / windump
    - bro (demo)
- For any technology, routers (and internal "switches") can look at / export traffic they forward
- You can also "tap" a link
  - Insert a device to mirror physical signal
  - Or: just steal it!

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.

Physical/Link-Layer Threats: *Disruption*

- With physical access to a subnetwork, attacker can
  - Overwhelm its signaling
    - E.g., jam WiFi's RF
  - Send messages that violate the Layer-2 protocol's rules
    - E.g., send messages > maximum allowed size, sever timing synchronization, ignore fairness rules
- Routers & switches can simply "drop" traffic
- 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

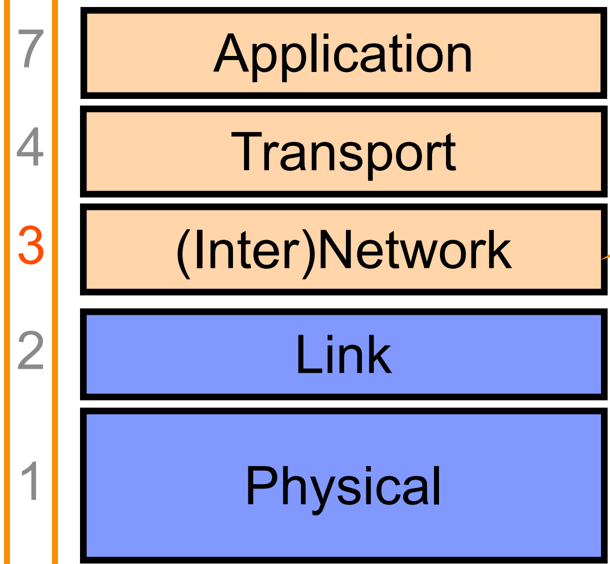
AT&T is now offering a $250,000 reward for information leading to the arrest of whoever is responsible for severing lines fiber optic cables in San Jose tha left much of the area without phone or cell service Thursday.

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

Physical/Link-Layer Threats: *Injection*

- With physical access to a subnetwork, attacker can create any message they like
- May require root/administrator access to have full freedom
- Particularly powerful when combined with *eavesdropping*
  – Can manipulate existing communications

Layer 3: General Threats?



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

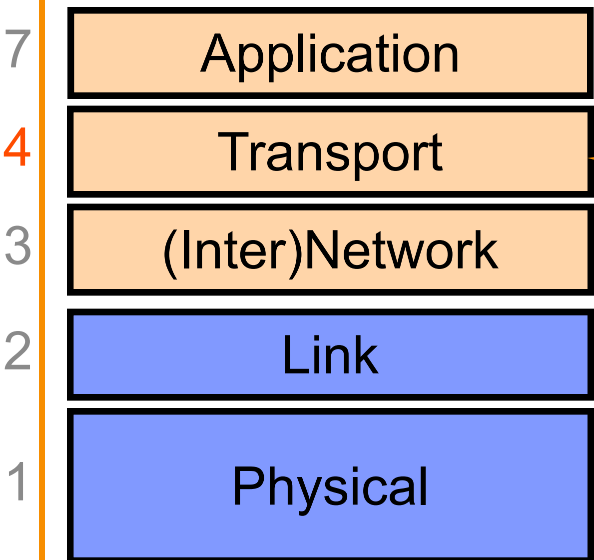
4-bit Version	4-bit Header Length	8-bit Type of Service (TOS)	16-bit Total Length (Bytes)
16-bit Identification		3-bit Flags	13-bit Fragment Offset
8-bit Time to Live (TTL)	8-bit Protocol	16-bit Header Checksum	
32-bit Source IP Address			
32-bit Destination IP Address			
Payload			

IP = Internet Protocol

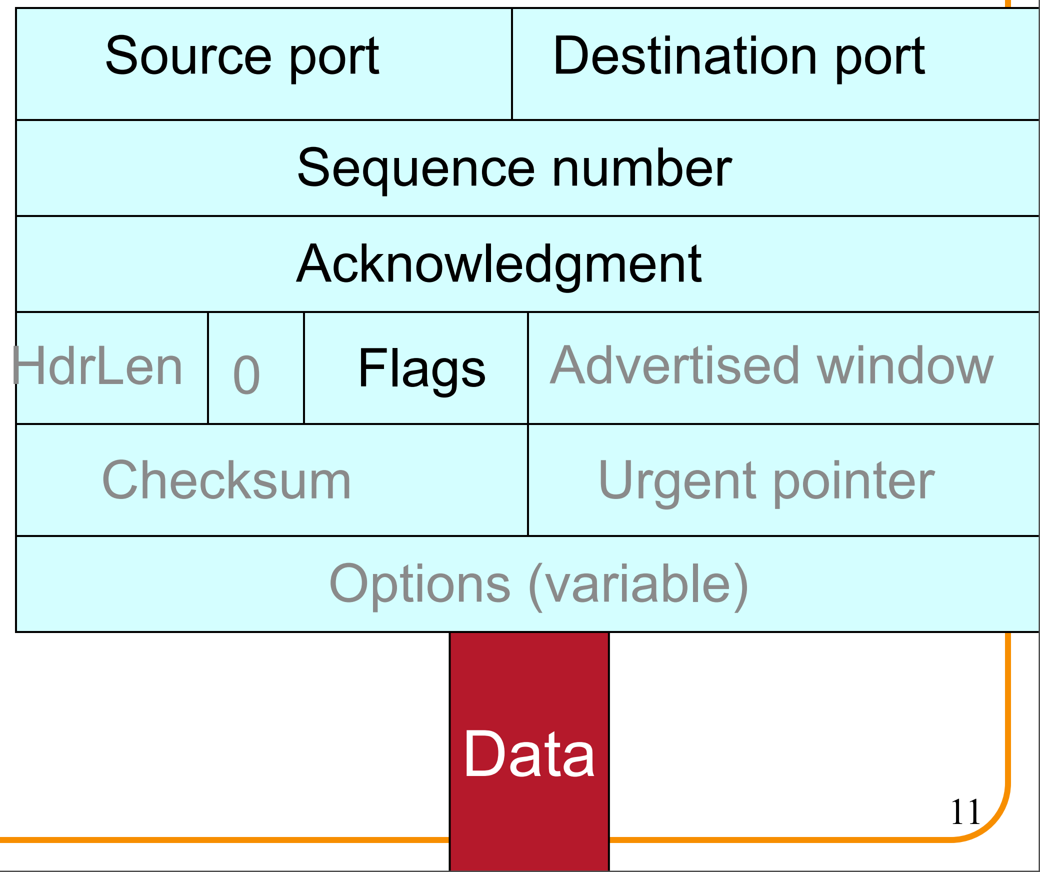
Network-Layer Threats

- Major:
  - Can set arbitrary source address
    - "*Spoofing*" - receiver has no idea who you are
  - Can set arbitrary destination address
    - Enables "*scanning*" - brute force searching for hosts
- Lesser: (FYI; don't worry about unless later explicitly covered)
  - Fragmentation mechanism can evade network monitoring
  - Identification field leaks information
  - Time To Live allows discovery of topology
  - TOS can let you steal high priority service
  - IP "options" can reroute traffic

Layer 4: General Threats?



End-to-end communication between processes (TCP, UDP)



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 numbers
- So: if attacker knows sequence numbers …

TCP Threat: Injection

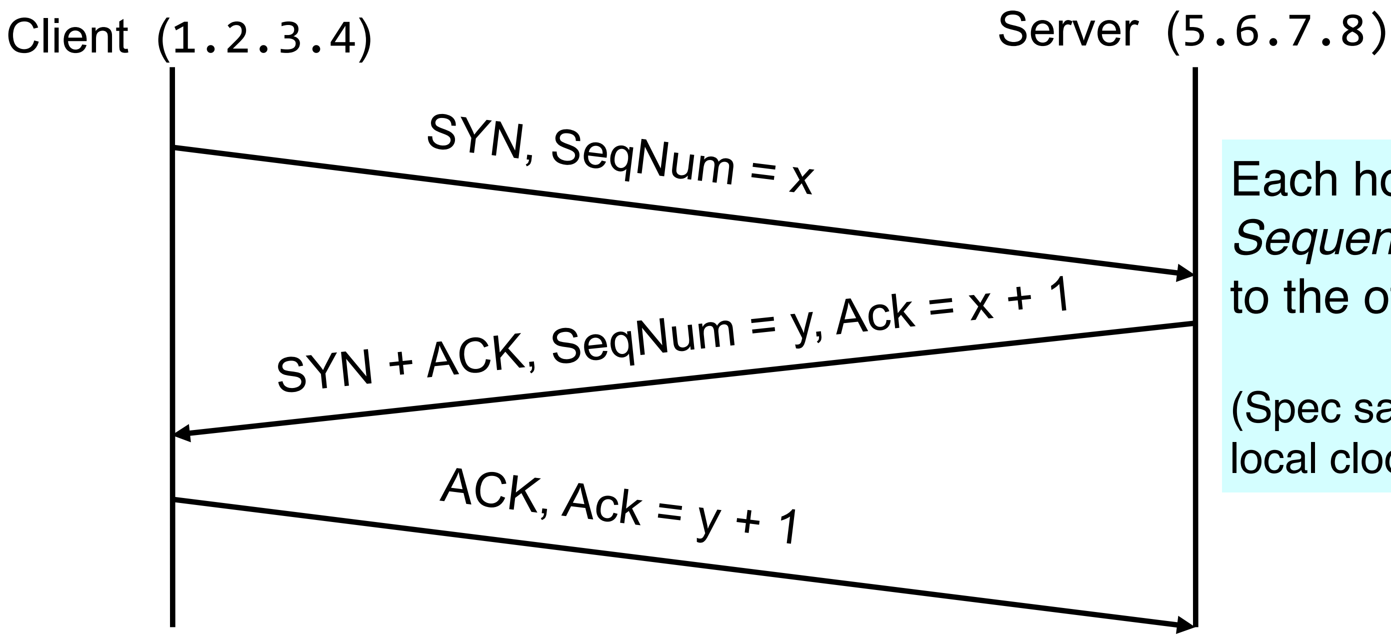
- If attacker knows sequence numbers, can inject whatever they like into TCP connection
- Instead of a RST, how about data?
- Note: *desynchronizes* client & server
  - They have inconsistent views of the byte stream and what acknowledgments refer to
  - However, if you've already killed one end with a spoofed RST, doesn't matter

⇒ TCP *session hijacking*

- General means to take over an already-established connection!
- We are toast if an attacker can see our TCP traffic

TCP Threat: Blind Spoofing

- TCP connection establishment:

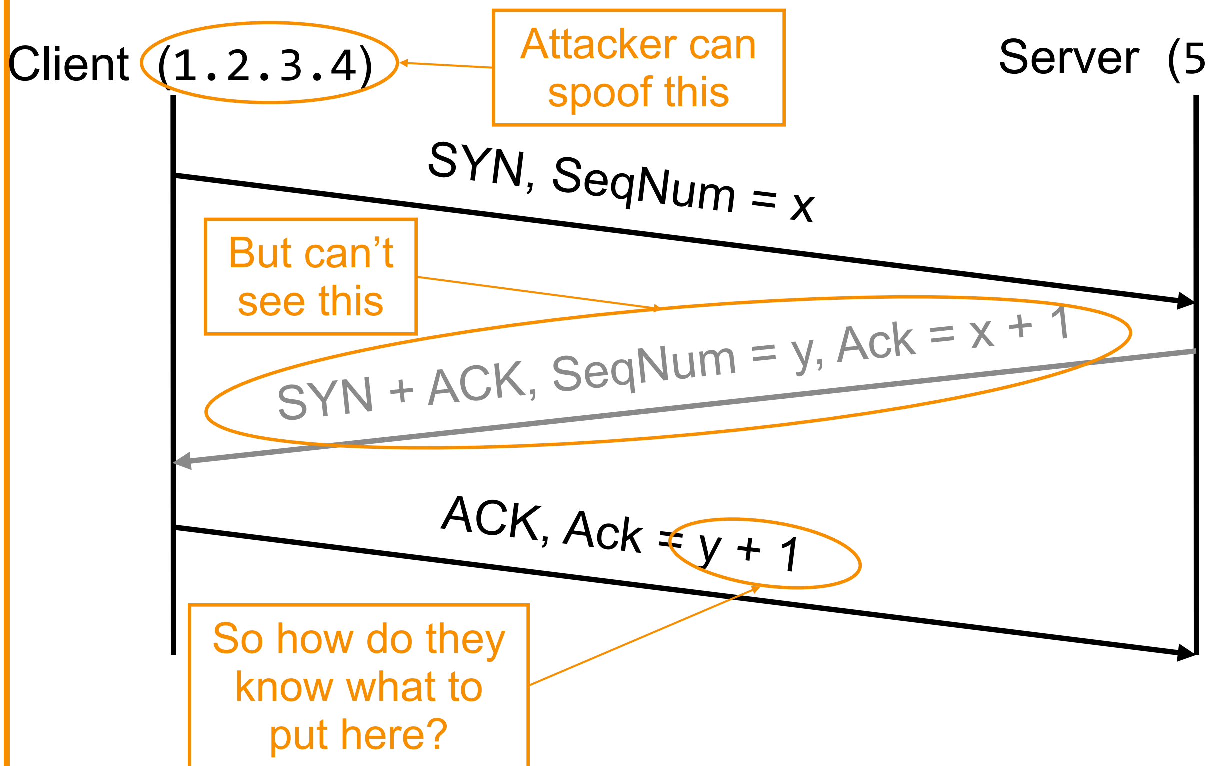


Each host tells its *Initial Sequence Number* (ISN) to the other host.

(Spec says to pick based on local clock)

- How can an attacker create an *apparent* connection from `1.2.3.4` to `5.6.7.8` even if they can't see the *real* `1.2.3.4`'s traffic?

Blind Spoofing: Attacker's Viewpoint



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?

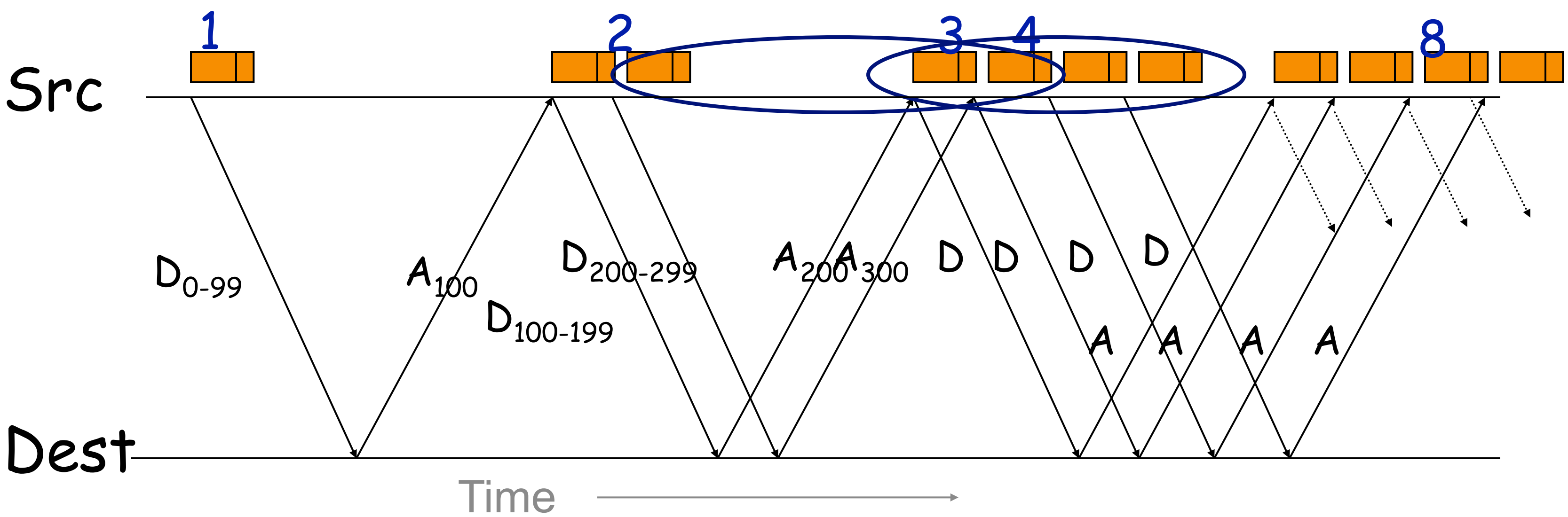
Use a *random* ISN

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

TCP's Exponential Rate Increase

Unless there's loss, TCP doubles data in flight every "round-trip"

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

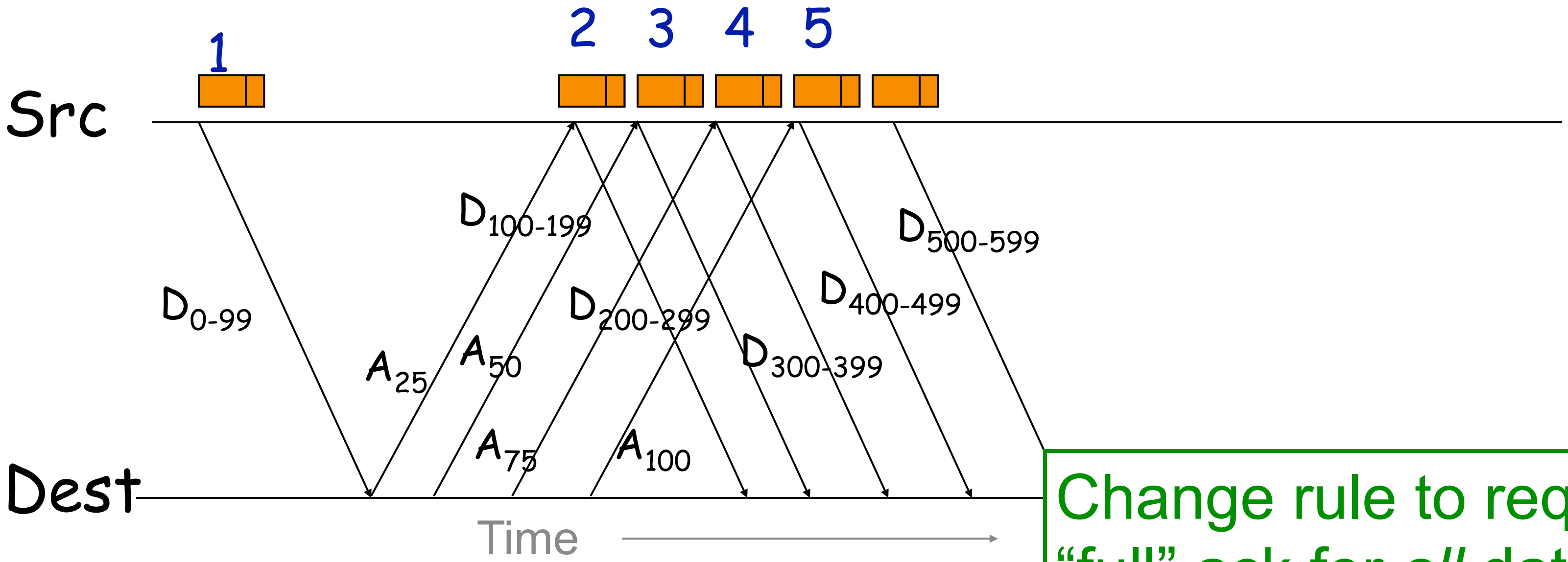


E.g., suppose maximum-sized packet = 100 bytes

TCP Threat: Cheating on Allowed Rate

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



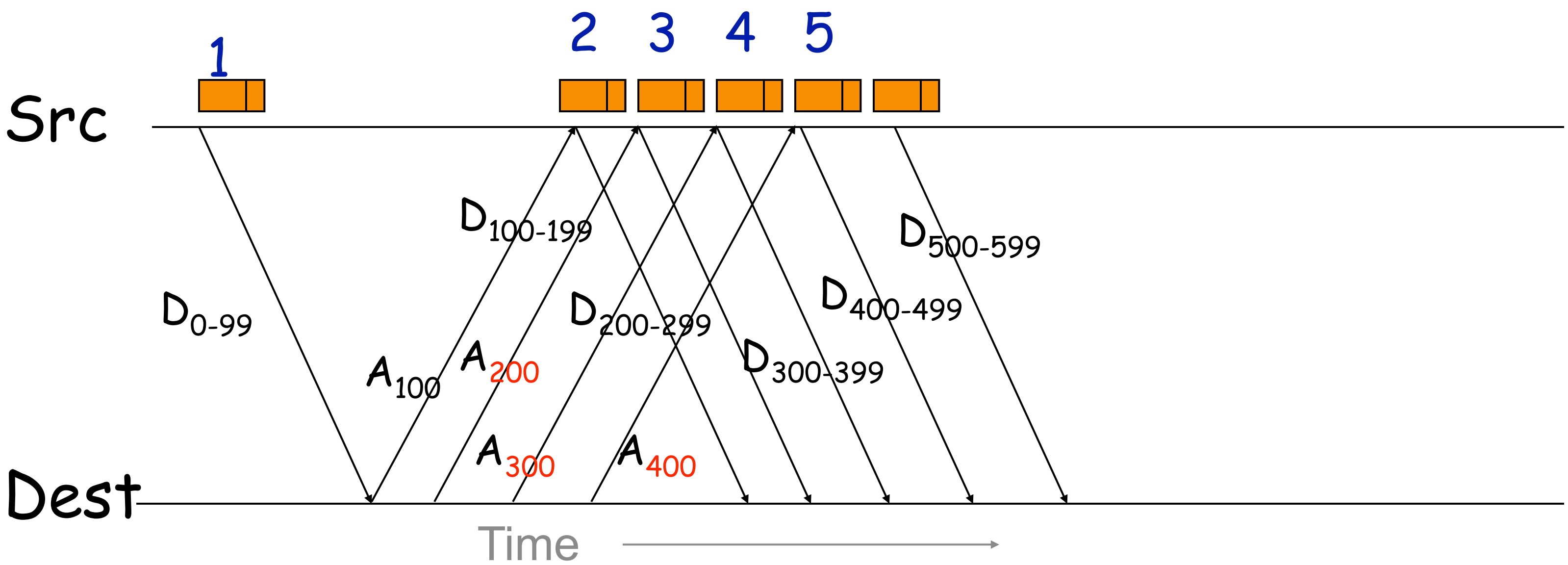
How do we defend against this?

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

TCP Threat: Cheating on Allowed Rate

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

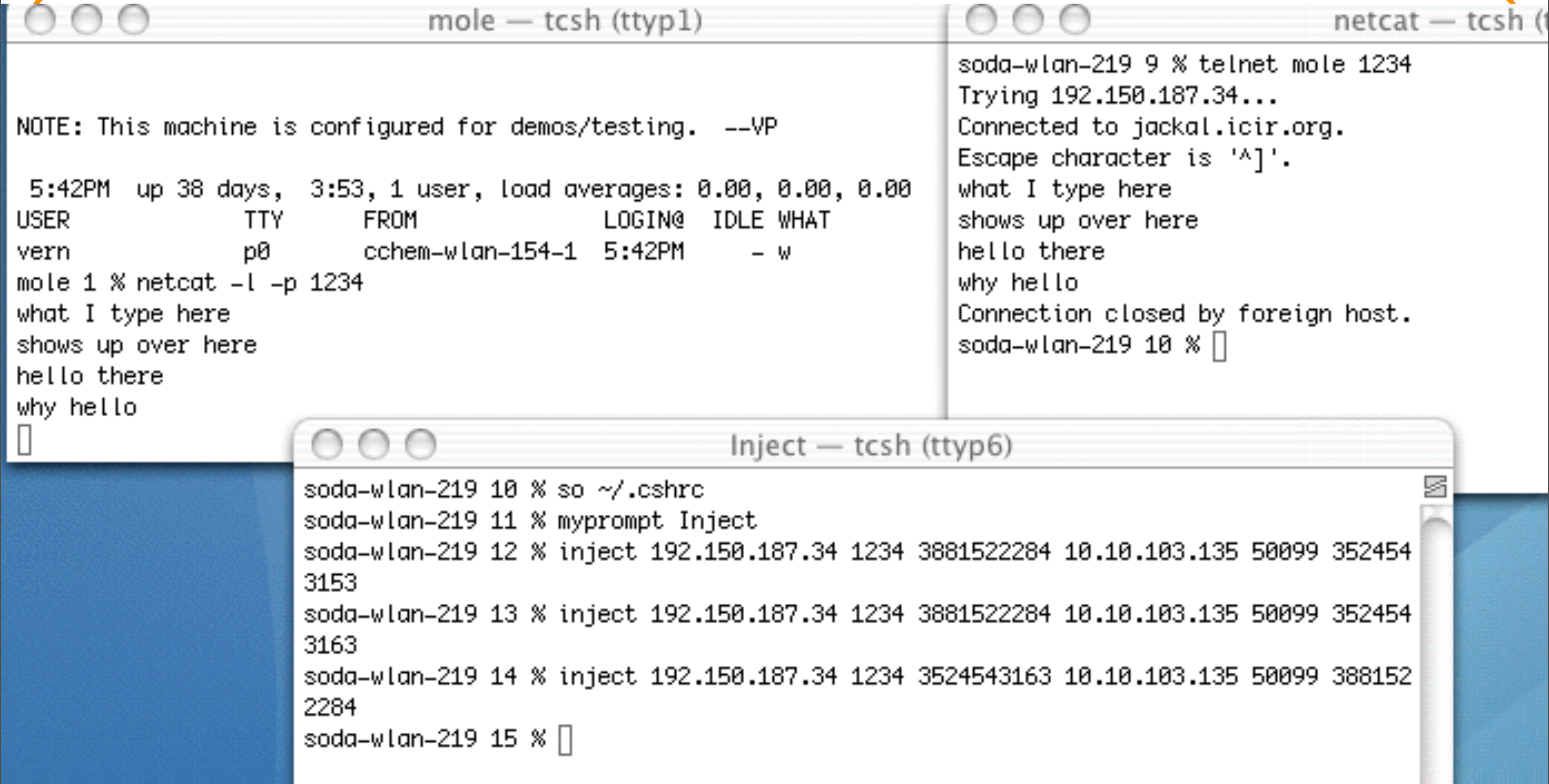
- Approach #1: if you receive an ack for data you haven't sent, kill the connection
  - Works only if receiver acks too far ahead
- Approach #2: follow the "round trip time" (RTT) and if an ack arrives too fast, kill the connection
  - Flaky: RTT can vary a lot, so you might kill innocent connections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

Summary of TCP Security Issues

- An attacker who can observe your TCP connection can manipulate it:
  - Forcefully terminate by forging a RST packet
  - Inject 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*



What we see here is that inject is taking over the connection. The netcat window has initiated a connection with mole on port 1234, and has sent some data ("what I type here", etc). Then we see that netcat indicates the connection has been closed. But mole has not closed the connection. Rather the inject window has closed the connection with netcat window, and remains connected to mole, who thinks it is talking to netcat.

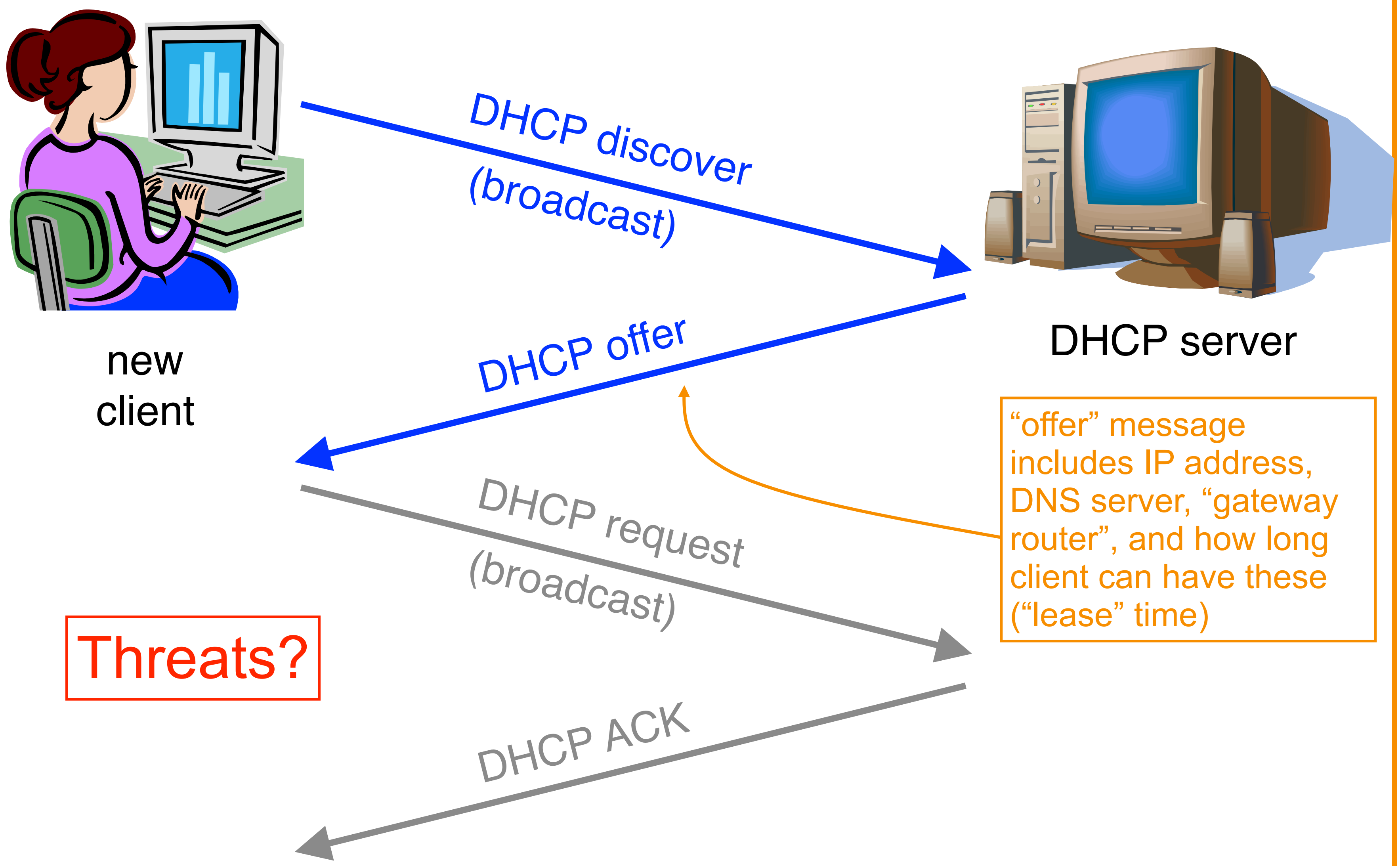
Summary of TCP Security Issues

- An attacker who can observe your TCP connection can manipulate it:
  - Forcefully terminate by forging a RST packet
  - Inject 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*
- An attacker who can predict the ISN chosen by a server can "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 today by choosing random ISNs
- Both highlight flawed "security-by-obscurity" assumption

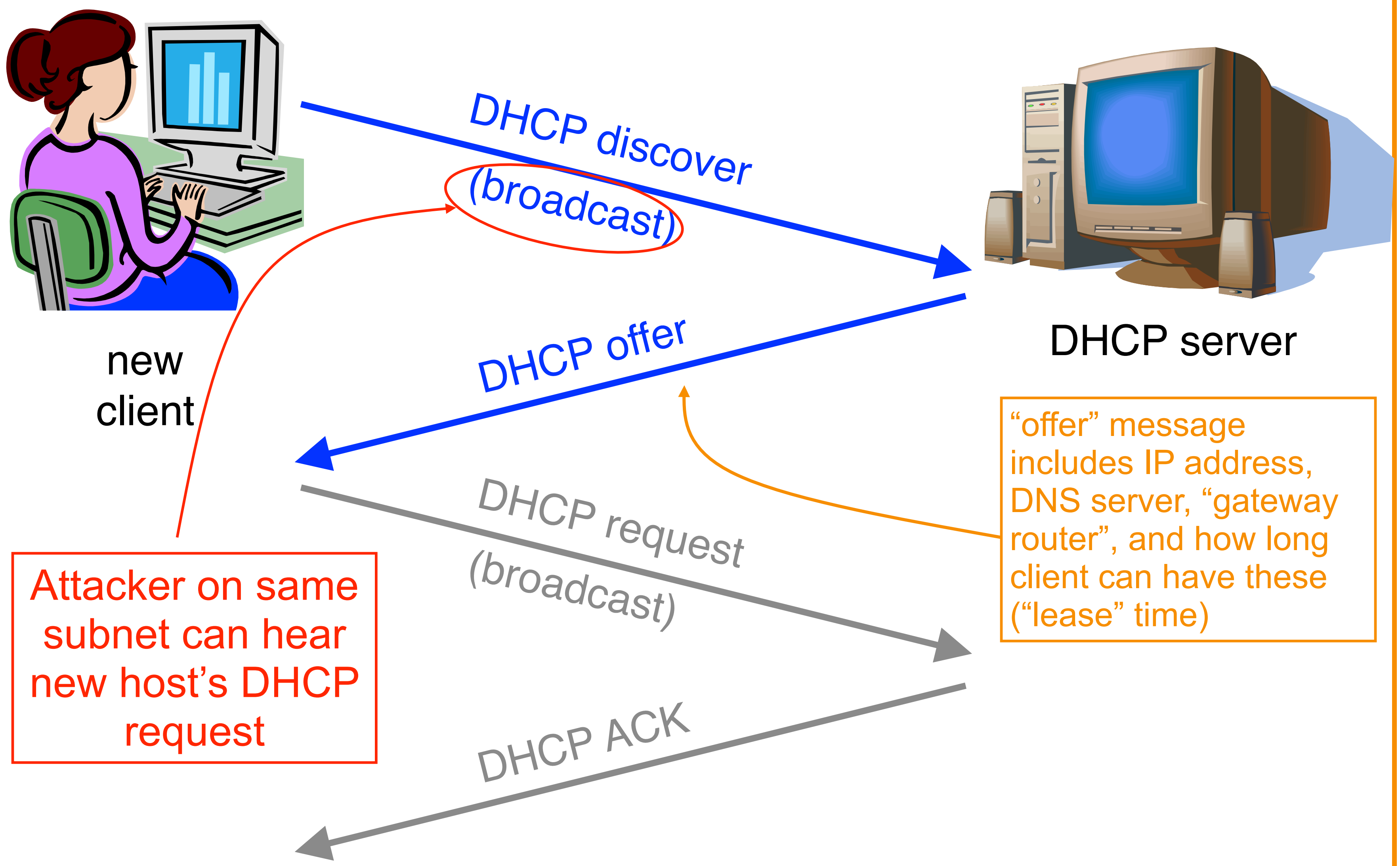
TCP Security Issues, con't

- TCP limits the rate at which senders transmit:
  - TCP relies on endpoints behaving properly to achieve "fairness" in how network capacity is used
  - Protocol lacks a mechanism to prevent cheating
  - Senders can cheat by just not abiding by the limits
    - Remains a significant threat: essentially nothing today prevents
- Receivers can manipulate honest senders into sending too fast because senders trust that receivers are honest
  - To a degree, sender can validate (e.g., partial acks)
  - A nonce can force receiver to only act on data they've seen
  - Rate manipulation remains a threat today
- General observation: tension between ease/power of protocols that assume everyone follows vs. violating
  - Security problems persist due to difficulties of retrofitting …
  - … coupled with investment in installed base

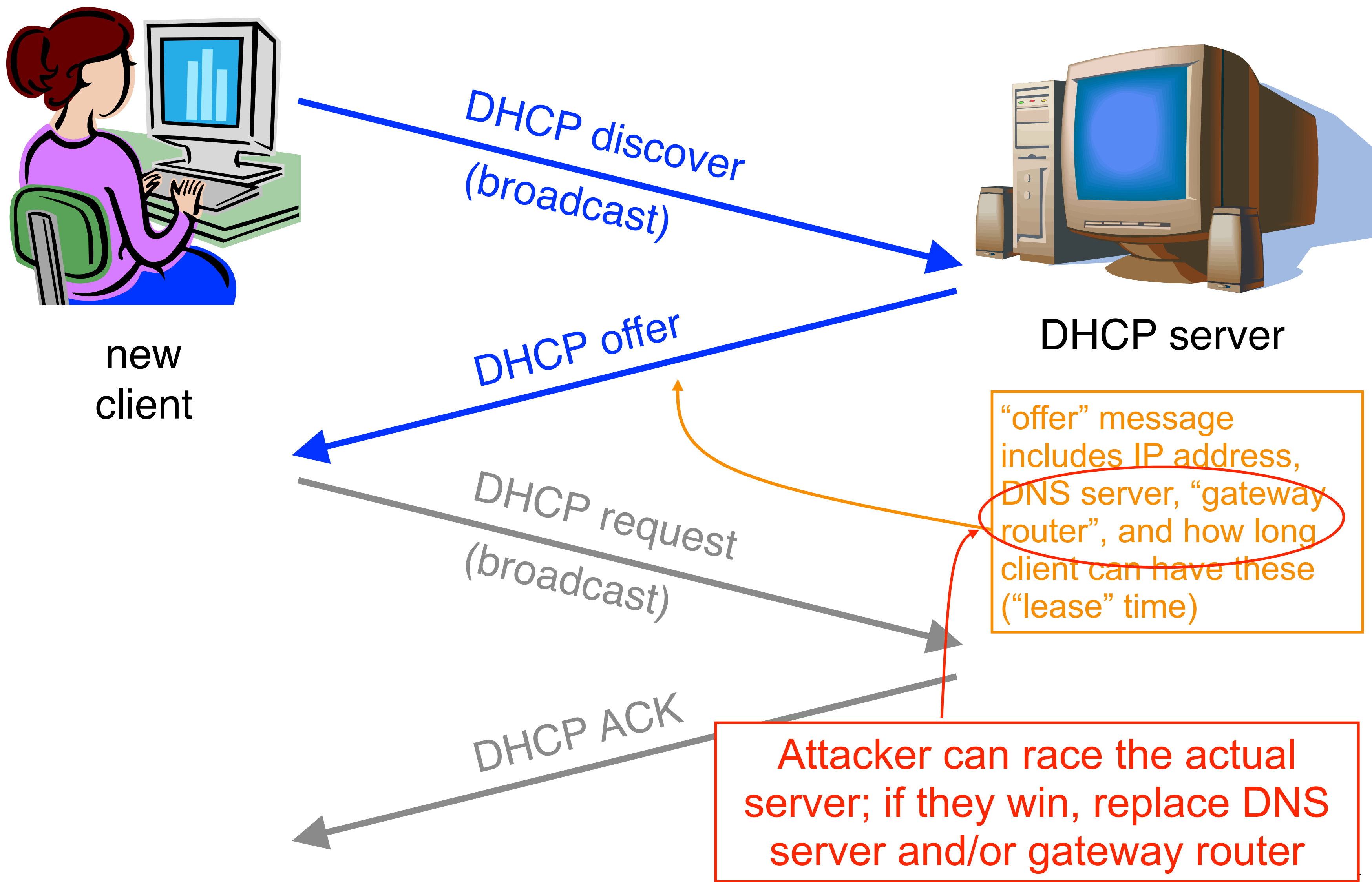
Dynamic Host Configuration Protocol



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"
  - 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
    - Modify however attacker chooses
- An invisible "Person In The Middle" (PITM)
  - 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?

Non-Eavesdropping Threats: DNS

- DHCP attacks show brutal power of attacker who can eavesdrop
- Consider attackers who *can't* eavesdrop - but still aim to manipulate us via how protocols function
- DNS: path-critical for just about everything we do
  - Maps hostnames ⇔ IP addresses
  - Design only **scales** if we can minimize lookup traffic
    - #1 way to do so: caching
    - #2 way to do so: return not only answers to queries, but additional info that will likely be needed shortly
- Directly interacting w/ DNS: **`dig`** program on Unix
  - Allows querying of DNS system
  - Dumps each field in DNS responses

dig eecs.mit.edu A

Use Unix "dig" utility to look up DNS address ("A") for hostname eecs.mit.edu

```
; ; <<>> DiG 9.6.0-APPLE-P2 <<>> eecs.mit.edu a
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 19901
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 3, ADDITIONAL: 3

;; QUESTION SECTION:
;eecs.mit.edu.                  IN      A

;; ANSWER SECTION:
eecs.mit.edu.           21600   IN      A       18.62.1.6

;; AUTHORITY SECTION:
mit.edu.                11088   IN      NS      BITSY.mit.edu.
mit.edu.                11088   IN      NS      W20NS.mit.edu.
mit.edu.                11088   IN      NS      STRAWB.mit.edu.

;; ADDITIONAL SECTION:
STRAWB.mit.edu.         126738  IN      A       18.71.0.151
BITSY.mit.edu.          166408  IN      A       18.72.0.3
W20NS.mit.edu.          126738  IN      A       18.70.0.160
```

dig eecs.mit.edu A

```
; ; <<>> DiG 9.6.0-APPLE-P2 <<>> eecs.mit.edu a
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 19901
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 3, ADDITIONAL: 3

;; QUESTION SECTION:
;eecs.mit.edu.                    IN      A

;; ANSWER SECTION:
eecs.mit.edu.         21600       IN      A       18.62.1.6

;; AUTHORITY SECTION:
mit.edu.              11088       IN      NS      BITSY.mit.edu.
mit.edu.              11088       IN      NS      W20NS.mit.edu.
mit.edu.              11088       IN      NS      STRAWB.mit.edu.

;; ADDITIONAL SECTION:
STRAWB.mit.edu.       126738      IN      A       18.71.0.151
BITSY.mit.edu.        166408      IN      A       18.72.0.3
W20NS.mit.edu.        126738      IN      A       18.70.0.160
```

These are just comments from dig itself with details of the request/response

dig eecs.mit.edu A

```
; ; <<>> DiG 9.6.0-APPLE-P2 <<>> eecs.mit.edu a
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 19901
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 3, ADDITIONAL: 3

;; QUESTION SECTION:
;eecs.mit.edu.                  IN      A

;; ANSWER SECTION:
eecs.mit.edu.           21600   IN      A       18.62.1.6

;; AUTHORITY SECTION:
mit.edu.                11088   IN      NS      BITSY.mit.edu.
mit.edu.                11088   IN      NS      W20NS.mit.edu.
mit.edu.                11088   IN      NS      STRAWB.mit.edu.

;; ADDITIONAL SECTION:
STRAWB.mit.edu.         126738  IN      A       18.71.0.151
BITSY.mit.edu.          166408  IN      A       18.72.0.3
W20NS.mit.edu.          126738  IN      A       18.70.0.160
```

Transaction identifier

dig eecs.mit.edu A

```
; ; <<>> DiG 9.6.0-APPLE-P2 <<>> eecs.mit.edu a
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 19901
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 3, ADDITIONAL: 3

;; QUESTION SECTION:
;eecs.mit.edu.                  IN      A

;; ANSWER SECTION:
eecs.mit.edu.           21600   IN      A       18.62.1.6

;; AUTHORITY SECTION:
mit.edu.                11088   IN      NS      BITSY.mit.edu.
mit.edu.                11088   IN      NS      W20NS.mit.edu.
mit.edu.                11088   IN      NS      STRAWB.mit.edu.

;; ADDITIONAL SECTION:
STRAWB.mit.edu.         126738  IN      A       18.71.0.151
BITSY.mit.edu.          166408  IN      A       18.72.0.3
W20NS.mit.edu.          126738  IN      A       18.70.0.160
```

Here the server echoes back the question that it is answering

dig eecs.mit.edu A

```
; ; <<>> DiG 9.6.0-APPLE-P2 <<>> eecs.mit.edu a
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 19901
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 3, ADDITIONAL: 3

;; QUESTION SECTION:
;eecs.mit.edu.                  IN      A

;; ANSWER SECTION:
eecs.mit.edu.           21600   IN      A       18.62.1.6

;; AUTHORITY SECTION:
mit.edu.                11088   IN      NS      BITSY.mit.edu.
mit.edu.                11088   IN      NS      W20NS.mit.edu.
mit.edu.                11088   IN      NS      STRAWB.mit.edu.

;; ADDITIONAL SECTION:
STRAWB.mit.edu.         126738  IN      A       18.71.0.151
BITSY.mit.edu.          166408  IN      A       18.72.0.3
W20NS.mit.edu.          126738  IN      A       18.70.0.160
```

"Answer" tells us its address is `18.62.1.6` and we can cache the result for 21,600 seconds

dig eecs.mit.edu A

```
; ; <<>> DiG 9.6.0-APPLE-P2 <<>> eecs.mit.edu a
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 19901
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 3, ADDITIONAL: 3

;; QUESTION SECTION:
;eecs.mit.edu.

;; ANSWER SECTION:
eecs.mit.edu.

;; AUTHORITY SECTION:
mit.edu.                11088   IN      NS      BITSY.mit.edu.
mit.edu.                11088   IN      NS      W20NS.mit.edu.
mit.edu.                11088   IN      NS      STRAWB.mit.edu.

;; ADDITIONAL SECTION:
STRAWB.mit.edu.         126738  IN      A       18.71.0.151
BITSY.mit.edu.          166408  IN      A       18.72.0.3
W20NS.mit.edu.          126738  IN      A       18.70.0.160
```

"Authority" tells us the *name servers* responsible for the answer. Each record gives the *hostname* of a different name server ("`NS`") for names in `mit.edu`.

We should cache each record for 11,088 seconds.

dig eecs.mit.edu A

```
; ; <<>> DiG 9.6.0-APPLE-P2 <<>> eecs.mit.edu a
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 19901
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 3, ADDITIONAL: 3

;; QUESTION SECTION:
;eecs.mit.edu.                  IN      A

;; ANSWER SECTION
eecs.mit.edu.

;; AUTHORITY SECTION:
mit.edu.                11088   IN      NS      BITSY.mit.edu.
mit.edu.                11088   IN      NS      W20NS.mit.edu.
mit.edu.                11088   IN      NS      STRAWB.mit.edu.

;; ADDITIONAL SECTION:
STRAWB.mit.edu.         126738  IN      A       18.71.0.151
BITSY.mit.edu.          166408  IN      A       18.72.0.3
W20NS.mit.edu.          126738  IN      A       18.70.0.160
```

"Additional" provides extra information to save us from making separate lookups for it, or helps with bootstrapping.

Here, it tells us the IP addresses for the hostnames of the name servers. We add these to our cache.

dig eecs.mit.edu A

```
; ; <<>> DiG 9.6.0-APPLE-P2 <<>> eecs.mit.edu a
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 19901
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 3, ADDITIONAL: 3

;; QUESTION SECTION:
;eecs.mit.edu.
```

What happens if the mit.edu server returns the following to us instead?

```
;; ANSWER SECTION:
eecs.mit.edu.          21600   IN      A       18.62.1.6

;; AUTHORITY SECTION:
mit.edu.               11088   IN      NS      BITSY.mit.edu.
mit.edu.               11088   IN      NS      W20NS.mit.edu.
mit.edu.               30      IN      NS      eecs.berkeley.edu.

;; ADDITIONAL SECTION:
eecs.berkeley.edu.     30      IN      A       18.6.6.6
BITSY.mit.edu.         166408  IN      A       18.72.0.3
W20NS.mit.edu.         126738  IN      A       18.70.0.160
```

dig eecs.mit.edu A

```
; ; <<>> DiG 9.6.0-APPLE-P2 <<>> eecs.mit.edu a
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 19901
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 3, ADDITIONAL: 3

;; QUESTION SECTION:
;eecs.mit.edu.                  IN      A

;; ANSWER SECTION:
eecs.mit.edu.

;; AUTHORITY SECTION:
mit.edu.                11088   IN      NS      BITSY.mit.edu.
mit.edu.                11088   IN      NS      W20NS.mit.edu.
mit.edu.                30      IN      NS      eecs.berkeley.edu.

;; ADDITIONAL SECTION:
eecs.berkeley.edu.      30      IN      A       18.6.6.6
BITSY.mit.edu.          166408  IN      A       18.72.0.3
W20NS.mit.edu.          126738  IN      A       18.70.0.160
```

We dutifully store in our cache a mapping of `eecs.berkeley.edu` to an IP address under MIT's control. (It could have been any IP address they wanted, not just one of theirs.)

dig eecs.mit.edu A

```
; ; <<>> DiG 9.6.0-APPLE-P2 <<>> eecs.mit.edu a
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 19901
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 3, ADDITIONAL: 3

;; QUESTION SECTION:
;eecs.mit.edu.                  IN      A

;; ANSWER SECTION:
eecs.mit.edu.                                            6

;; AUTHORITY SECTION:
mit.edu.                11088   IN      NS      BITSY.mit.edu.
mit.edu.                11088   IN      NS      W20NS.mit.edu.
mit.edu.                30      IN      NS      eecs.berkeley.edu.

;; ADDITIONAL SECTION:
eecs.berkeley.edu.      30      IN      A       18.6.6.6
BITSY.mit.edu.          166408  IN      A       18.72.0.3
W20NS.mit.edu.          126738  IN      A       18.70.0.160
```

In this case they chose to make the mapping *disappear* after 30 seconds. They could have made it persist for weeks, or disappear even quicker.

dig eecs.mit.edu A

```
; ; <<>> DiG 9.6.0-APPLE-P2 <<>> eecs.mit.edu a
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 19901
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 3, ADDITIONAL: 3

;; QUESTION SECTION:
;eecs.mit.edu.                    IN      A

;; ANSWER SECTION:
eecs.mit.edu.

;; AUTHORITY SECTION:
mit.edu.                11088     IN      NS      BITSY.mit.edu.
mit.edu.                11088     IN      NS      W20NS.mit.edu.
mit.edu.                30        IN      NS      eecs.berkeley.edu.

;; ADDITIONAL SECTION:
eecs.berkeley.edu.      30        IN      A       18.6.6.6
BITSY.mit.edu.          166408    IN      A       18.72.0.3
W20NS.mit.edu.          126738    IN      A       18.70.0.160
```

How do we fix such *cache poisoning*?

dig eecs.mit.edu A

```
; ; <<>> DiG 9.6.0-APPLE-P2 <<>> eecs.mit.edu a
;; global options: +cr
;; Got answer:
;; ->>HEADER<<- opcode
;; flags: qr rd ra; QU

;; QUESTION SECTION:
;eecs.mit.edu.

;; ANSWER SECTION:
eecs.mit.edu.

;; AUTHORITY SECTION:
mit.edu.                 11088   IN      NS      BITSY.mit.edu.
mit.edu.                 11088   IN      NS      W20NS.mit.edu.
mit.edu.                 30      IN      NS      eecs.berkeley.edu.

;; ADDITIONAL SECTION:
```

~~eecs.berkeley.edu. 30 IN A 18.6.6.6~~

```
BITSY.mit.edu.           166408  IN      A       18.72.0.3
W20NS.mit.edu.           126738  IN      A       18.70.0.160
```

Don't accept Additional records unless they're for the domain we're looking up

- E.g., looking up `eecs.mit.edu` ⇒ only accept additional records from `*.mit.edu`

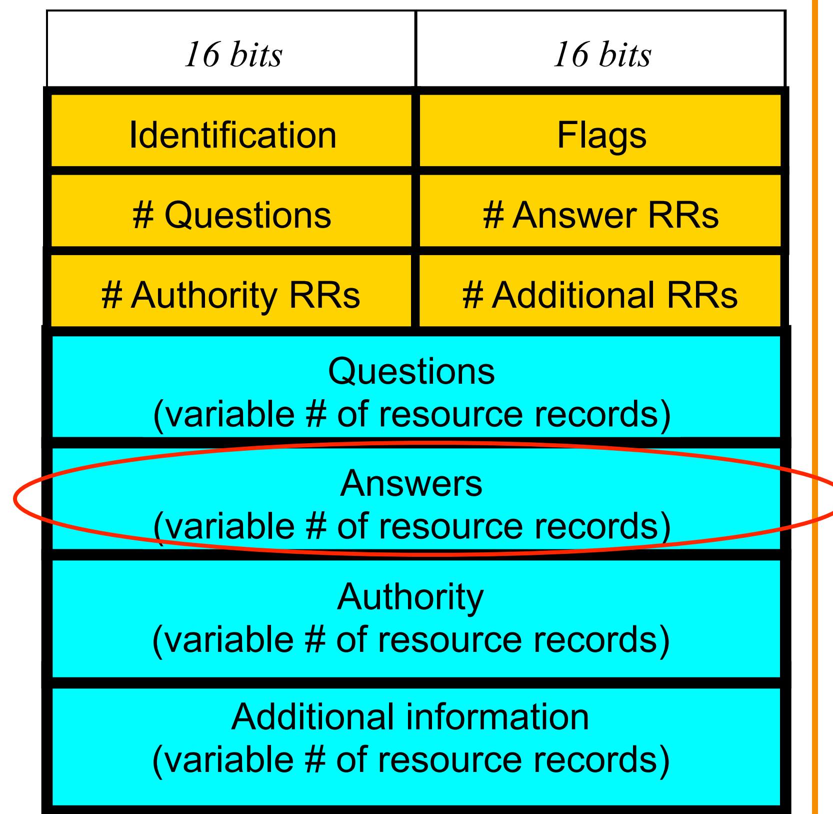
No extra risk in accepting these since server could return them to us directly in an Answer anyway.

DNS Threats, con’t

What about *blind spoofing*?

- Say we look up `mail.google.com`; how can an off-path attacker feed us a bogus A answer before the legitimate server replies?

- How can such an attacker even know we are looking up `mail.google.com`?




```
<img src="http://mail.google.com" …>
```

DNS Blind Spoofing, con’t

Fix?

Once they know we’re looking it up, they just have to guess the Identification field and reply before legit server.

How hard is that?

Originally, identification field incremented by 1 for each request. How does attacker guess it?

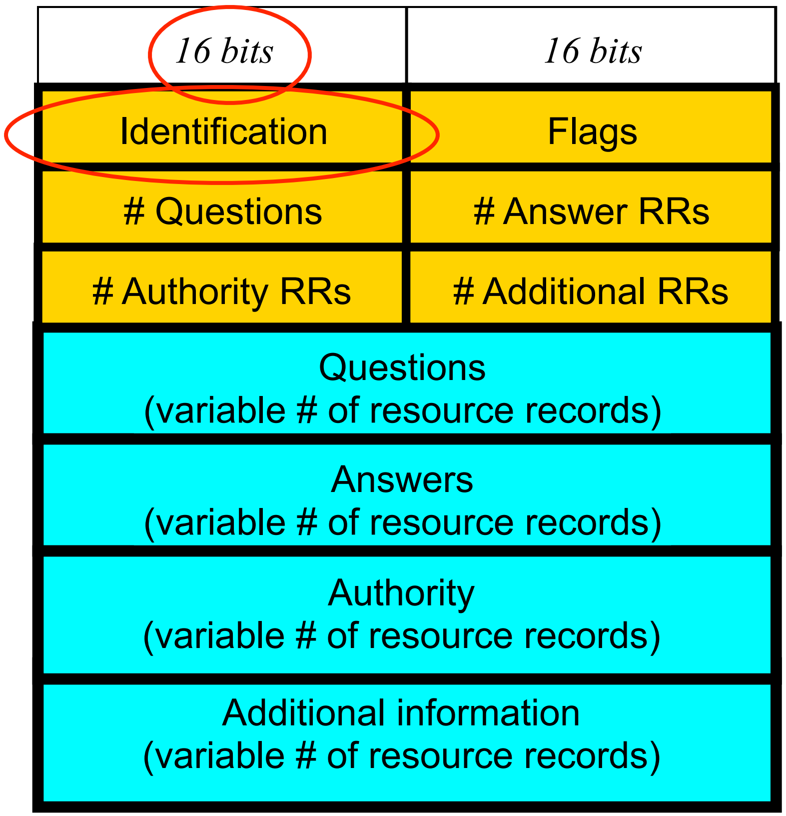
16 bits	*16 bits*
Identification	Flags
# Questions	# Answer RRs
# Authority RRs	# Additional RRs
Questions (variable # of resource records)	
Answers (variable # of resource records)	
Authority (variable # of resource records)	
Additional information (variable # of resource records)	

<img src="http://badguy.com" …> ← They observe ID k here

<img src="http://mail.google.com" …> ← So this will be k+1

DNS Blind Spoofing, con't

Once we randomize the Identification, attacker has a 1/65536 chance of guessing it correctly.
Are we pretty much safe?



Attacker can send *lots* of replies, not just one …

However: once reply from legit server arrives (with correct Identification), it's cached and no more opportunity to poison it. Victim is innoculated!

Unless attacker can send 1000s of replies before legit arrives, we're likely safe - phew! ?

DNS Blind Spoofing (Kaminsky 2008)

- Two key ideas:
  - Spoof uses Additional field (rather than Answer)
  - Attacker can get around caching of legit replies by generating a series of different name lookups:

```
<img src="http://random1.google.com" …>
<img src="http://random2.google.com" …>
<img src="http://random3.google.com" …>
                    . . .
<img src="http://randomN.google.com" …>
```

Kaminsky Blind Spoofing, con't

For each lookup of `randomk.google.com`, attacker returns a bunch of records like this, each with a different Identifier

```
;; QUESTION SECTION:
;randomk.google.com.               IN      A

;; ANSWER SECTION:
randomk.google.com        21600    IN      A       doesn't matter

;; AUTHORITY SECTION:
google.com.               11088    IN      NS      mail.google.com

;; ADDITIONAL SECTION:
mail.google.com           126738   IN      A       6.6.6.6
```

Once they win the race, not only have they poisoned `mail.google.com` …

Kaminsky Blind Spoofing, con't

For each lookup of `randomk.google.com`, attacker returns a bunch of records like this, each with a different Identifier

```
;; QUESTION SECTION:
;randomk.google.com.                 IN      A

;; ANSWER SECTION:
randomk.google.com        21600      IN      A       doesn't matter

;; AUTHORITY SECTION:
google.com.               11088      IN      NS      mail.google.com

;; ADDITIONAL SECTION:
mail.google.com           126738     IN      A       6.6.6.6
```

Once they win the race, not only have they poisoned `mail.google.com` … but also the cached NS record for `google.com`'s name server - so any future *X*`.google.com` lookups *go through the attacker's machine*

Note: It's not a matter of being lucky!

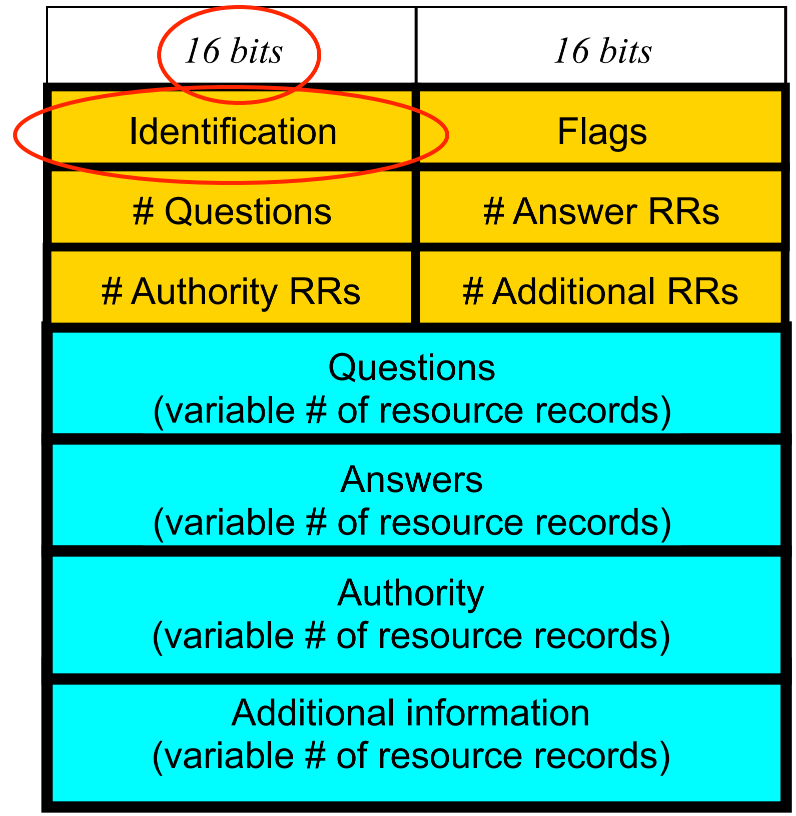
- The adversary know that all of these DNS requests are generated
- It also knows that the Query IDS are pseudorandomly generated.
- If it sees enough of these quickly enough, it can determine the parameters of the pseudorandom number generator!
- Then it knows what is coming next!

Defending Against Blind Spoofing

Central problem: all that tells a client they should accept a response is that it matches the Identification field.

With only 16 bits, it lacks sufficient entropy: even if truly random, the *search space* an attacker must *brute force* is too small.

Where can we get more entropy? (*Without* requiring a protocol change.)

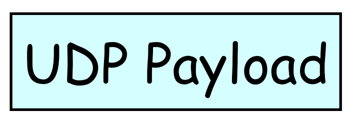


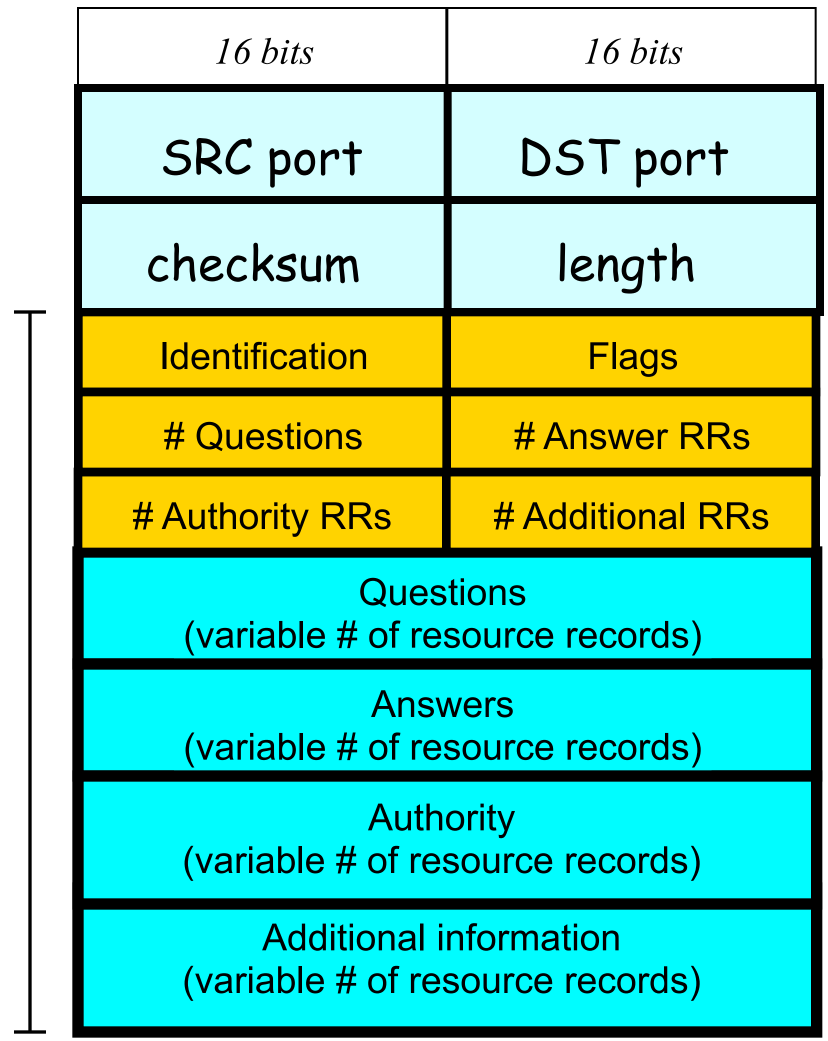
Defending Against Blind Spoofing

DNS (primarily) uses UDP for transport rather than TCP.

UDP header has:

- 16-bit Source & Destination ports (identify processes, like w/ TCP)
- 16-bit checksum, 16-bit length





Defending Against Blind Spoofing

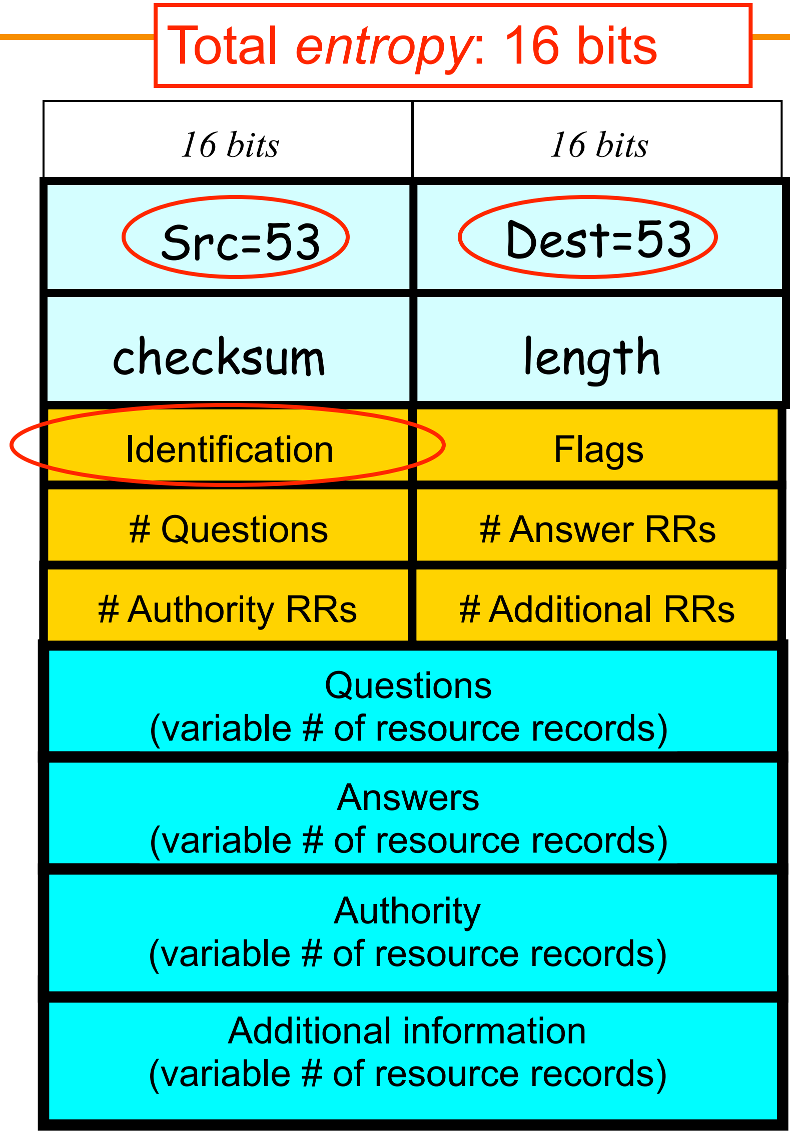
DNS (primarily) uses UDP for transport rather than TCP.

UDP header has:

- 16-bit Source & Destination ports (identify processes, like w/ TCP)
- 16-bit checksum, 16-bit length

For requestor to receive DNS reply, needs both correct Identification and correct ports.

On a request, DST port = 53. SRC port usually also 53 - but not fundamental, just convenient

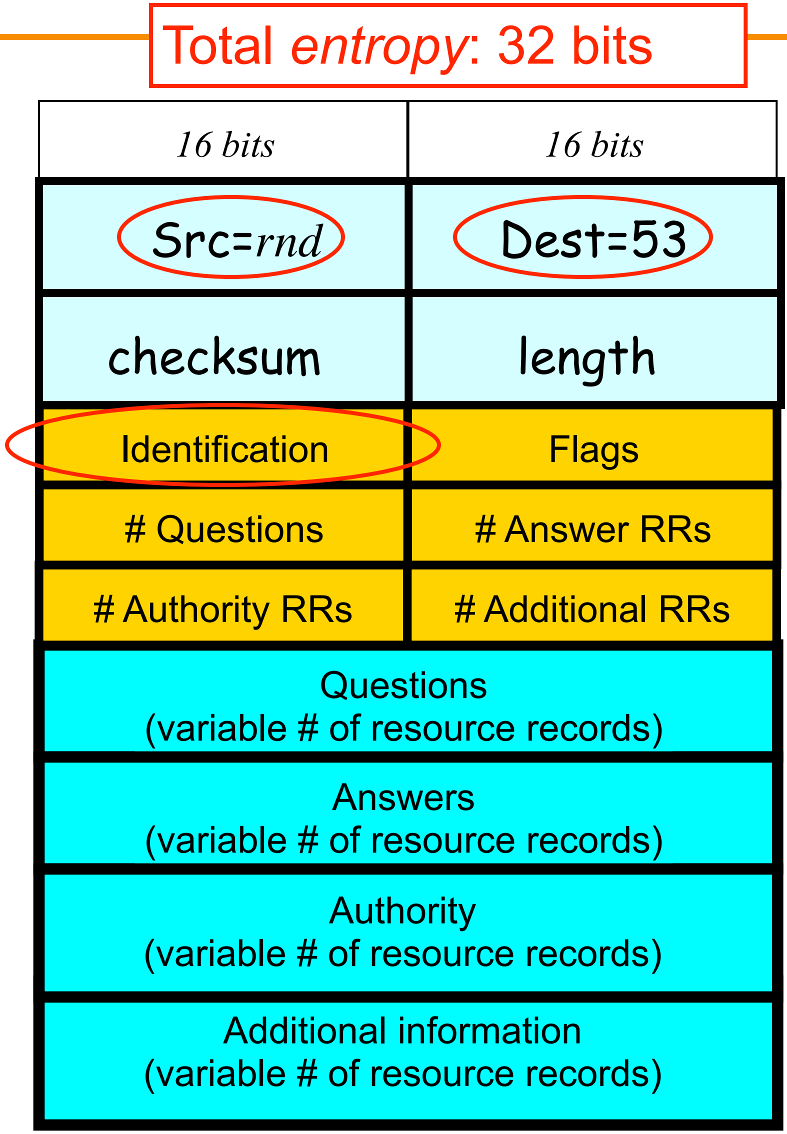


Defending Against Blind Spoofing

"Fix": use random source port

32 bits of entropy makes it orders of magnitude harder for attacker to guess all the necessary fields and dupe victim into accepting spoof response.

This is what primarily "secures" DNS today. (Note: not all resolvers have implemented random source ports!)



Summary of DHCP/DNS Security Issues

- DHCP threats highlight:
  - Broadcast protocols inherently at risk of attacker spoofing
    - Attacker knows exactly when to try it
  - When initializing, systems are particularly vulnerable because they can *lack a trusted foundation* to build upon
  - Tension between wiring in trust vs. flexibility/convenience
  - PITM attacks insidious because no indicators they're occurring

Summary of DHCP/DNS Security Issues

- DHCP threats highlight:
  - Broadcast protocols inherently at risk of attacker spoofing
    - Attacker knows exactly when to try it
  - When initializing, systems are particularly vulnerable because they can *lack a trusted foundation* to build upon
  - Tension between wiring in trust vs. flexibility/convenience
  - MITM attacks insidious because no indicators they're occurring
- DNS threats highlight:
  - Attackers can attack opportunistically rather than eavesdropping
    - Cache poisoning only requires victim to look up some name under attacker's control
  - Attackers can often manipulate victims into vulnerable activity
    - E.g., `IMG SRC` in web page to force DNS lookups
  - Crucial for identifiers associated with communication to have sufficient entropy (= a lot of bits of unpredictability)
  - "Attacks only get better": threats that appears technically remote can become practical due to unforeseen cleverness

Questions?