Internet Protocol Security (Contd.)

CS155 Computer and Network Security

Stanford University

L2: Ethernet

Provides connectivity between hosts on a single *Local Area Network*

Data is split into ~1500 byte **Frames,** which are addressed to a device's physical (MAC) address — assigned by manufacturer

Switches forward frames based on *learning* where different MACs are located. *No guarantees not sent to other hosts!*

No security (confidentiality, authentication, or integrity)

ARP: Address Resolution Protocol

ARP lets hosts to find each others' MAC addresses on a local network. For example, when you need to send packets to the upstream router to reach Internet hosts

Client: Broadcast (all MACs): Which MAC address has IP 192.168.1.1?
Response: I have this IP address (sent from correct MAC)

No built-in security. Attacker can impersonate a host by faking its identity and responding to ARP requests or sending gratuitous ARP announcements

IP: Internet Protocol

Provides routing between hosts on the Internet. Unreliable. Best Effort.

- Packets can be dropped, corrupted, repeated, reordered

Routers simply route IP packets based on their destination address.

- Must be simple in order to be fast — insane number packets FWD'ed

No inherent security. Packets have a checksum, but it's non-cryptographic. Attackers can change any packet.

Source address is set by sender—can be faked by an attacker

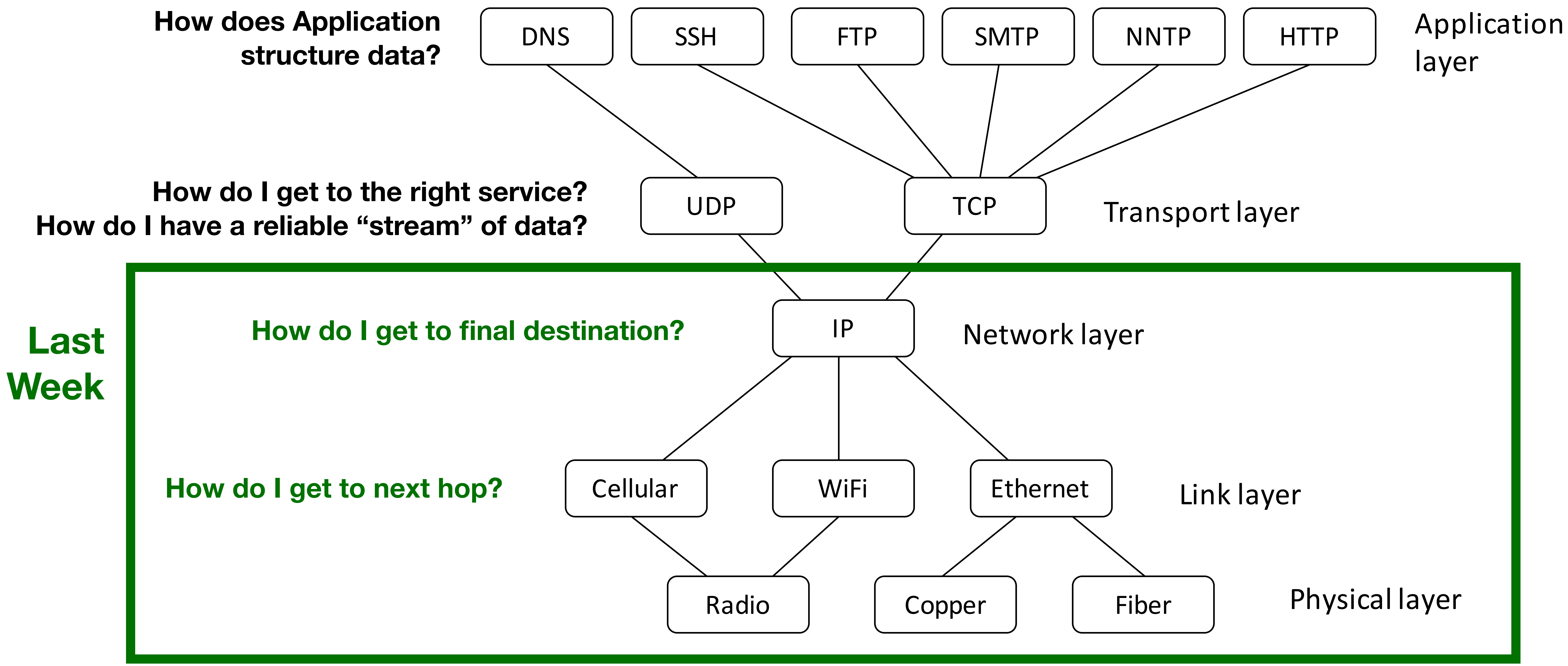
BGP (Border Gateway Protocol)

Internet Service Providers (ISPs) announce their presence on the Internet via BGP. Each router maintains list of routes to get to different announced prefixes

No authentication—possible to announce someone else's network

Commonly occurs (often due to operator error but also due to attacks)

Protocol Layering



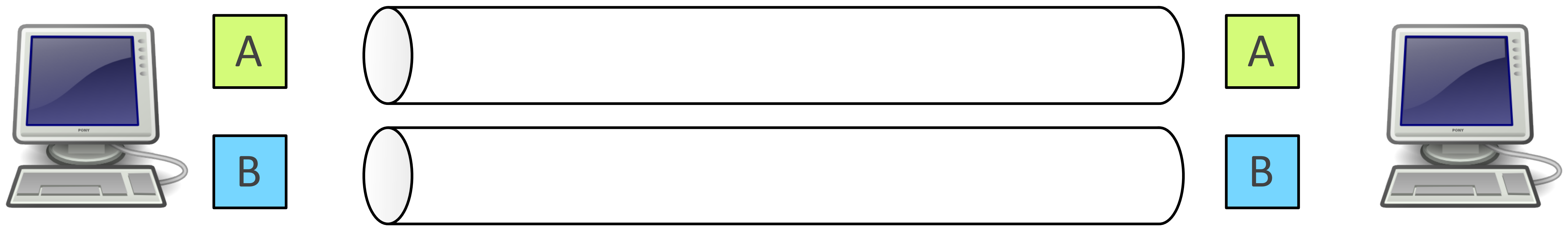
Ports

Each application (e.g., HTTP server) on a host is identified by a *port number*

TCP connection established between port *A* on host *X* to port *B* on host *Y*

- Ports are 1–65535 (16 bits)

Some destination port numbers used for specific applications by convention



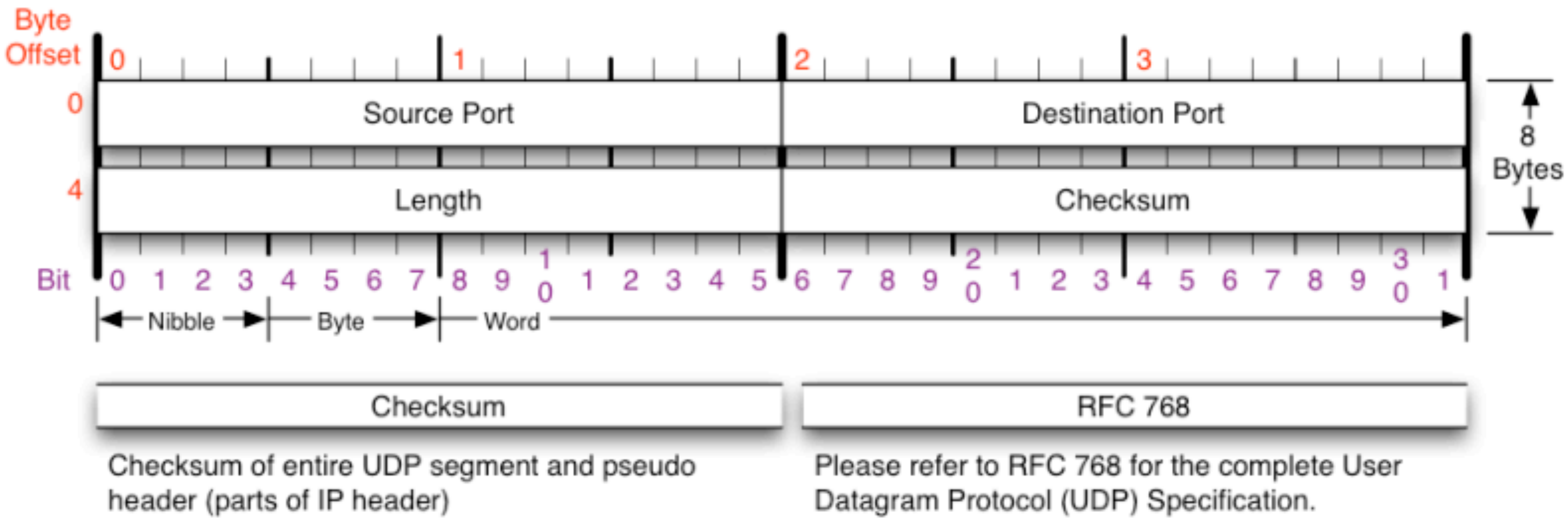
Common Ports

Port	Application
80	HTTP (Web)
443	HTTPS (Secure Web)
25	SMTP (mail delivery)
67	DHCP (host config)
22	SSH (secure shell)
23	Telnet

UDP (User Datagram Protocol)

User Datagram Protocol (UDP) is a transport layer protocol that is essentially a wrapper around IP

Adds ports to demultiplex traffic by application



From Packets to Streams

Most applications want a stream of bytes delivered reliably and in-order between applications on different hosts

Transmission Control Protocol (TCP) provides…

- Connection-oriented protocol with explicit setup/teardown
- Reliable in-order byte stream
- Congestion control

Despite IP packets being dropped, re-ordered, and duplicated

TCP Sequence Numbers

Two data streams in a TCP session, one in each direction

Bytes in data stream numbered with a 32-bit *sequence number*

Every packet has sequence number that indicates where data belongs

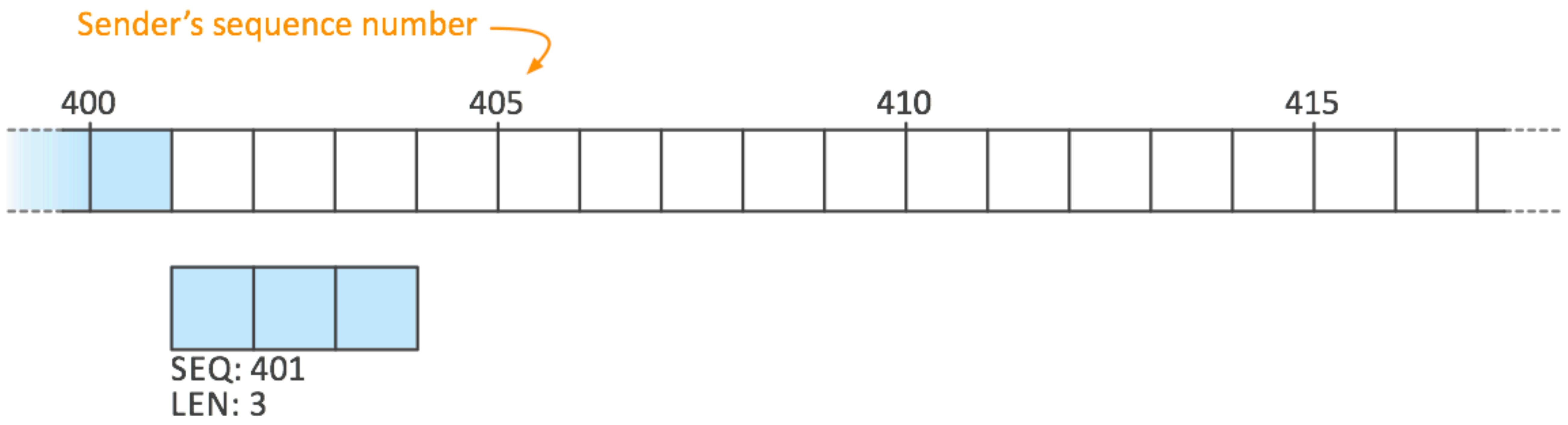
Receiver sends acknowledgement number that indicates data received

TCP Packet

Byte Offset
0
1
2
3
0
Source Port
Destination Port
4
Sequence Number
8
Acknowledgment Number
12
Offset
Reserved
TCP Flags
C E U A P R S F
Window
16
Checksum
Urgent Pointer
20
TCP Options (variable length, optional)
20 Bytes
Offse
Bit
0 1 2 3 4 5 6 7 8 9 10 1 2 3 4 5 6 7 8 9 20 1 2 3 4 5 6 7 8 9 30 1
Nibble
Byte
Word

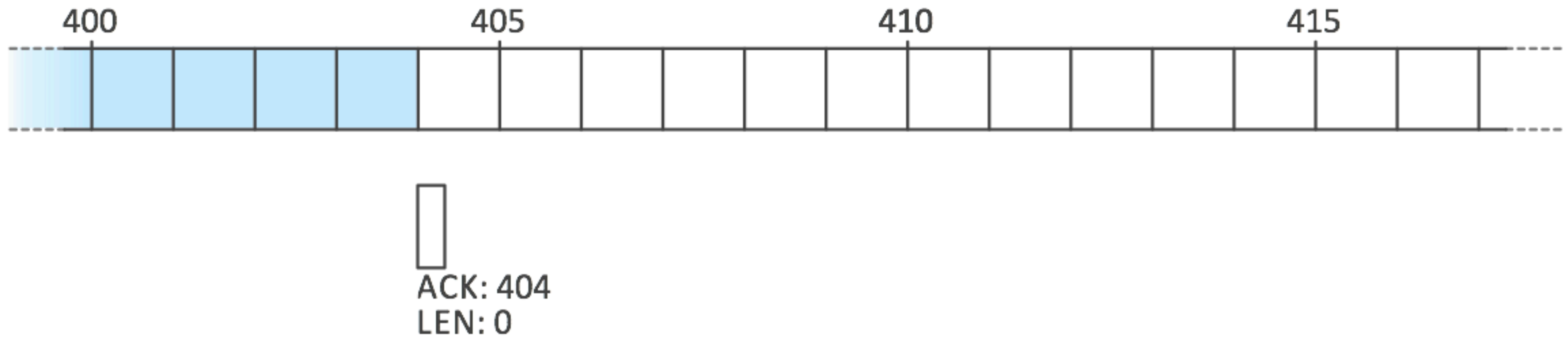
Transmission Control Protocol

- Sender sends 3 byte segment
- Sequence number indicates where data belongs in byte sequence (at byte 401)
  - *Note:* Wireshark shows *relative* sequence numbers



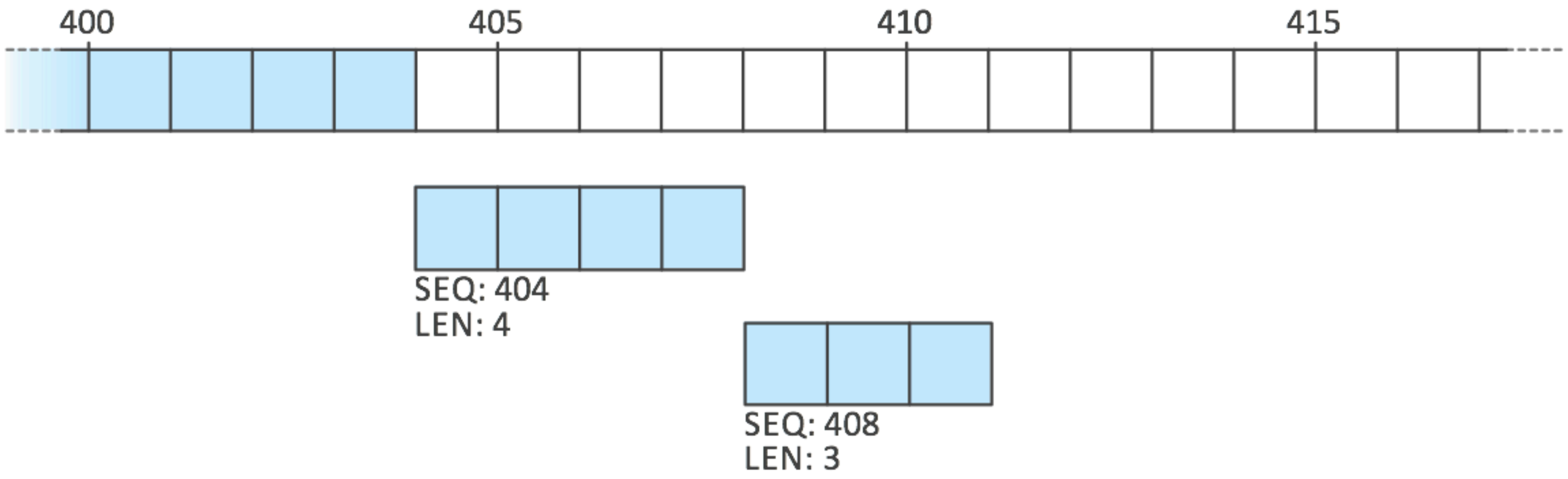
TCP Acknowledgement Numbers

- Receiver acknowledges received data
  - Sets ACK flag in TCP header
  - Sets acknowledgement number to indicate next expected byte in sequence



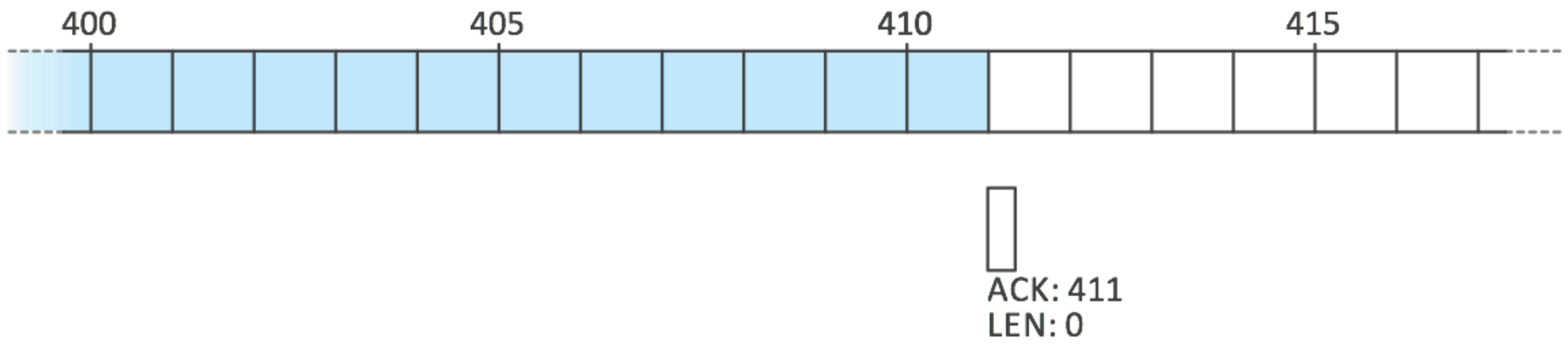
ACKing Multiple Segments

- Sender may send several segments before receiving acknowledgement



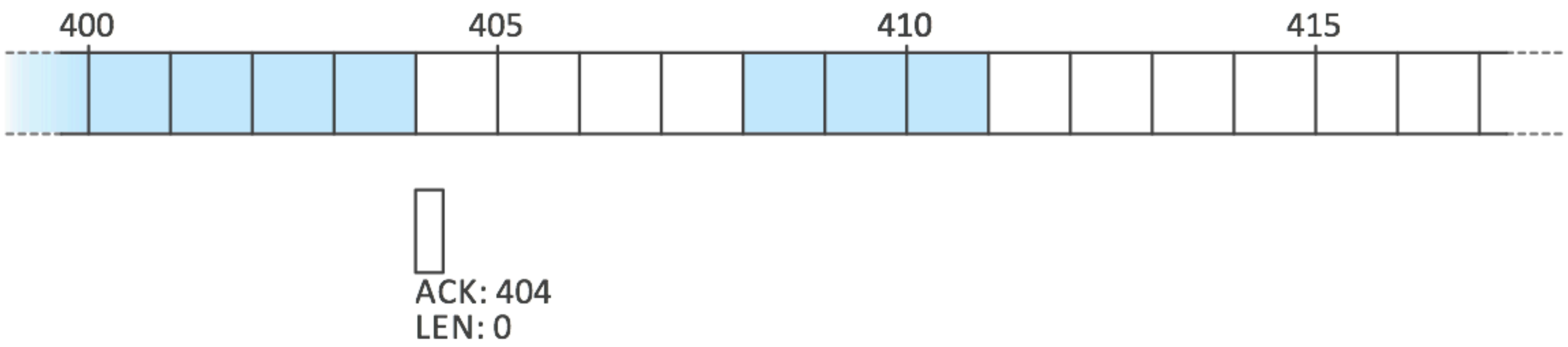
ACKing Multiple Segments

- Sender may send several segments before receiving acknowledgement
- Receiver always acknowledges with seq. no. of next expected byte



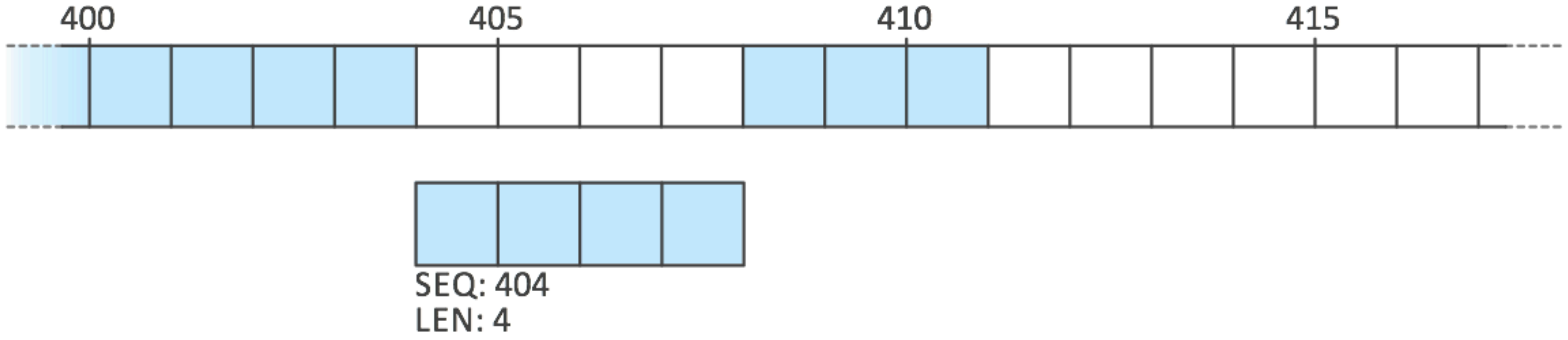
Transmission Control Protocol

- *What if the first packet is dropped in network?*
- Receiver always acknowledges with seq. no. of next expected byte



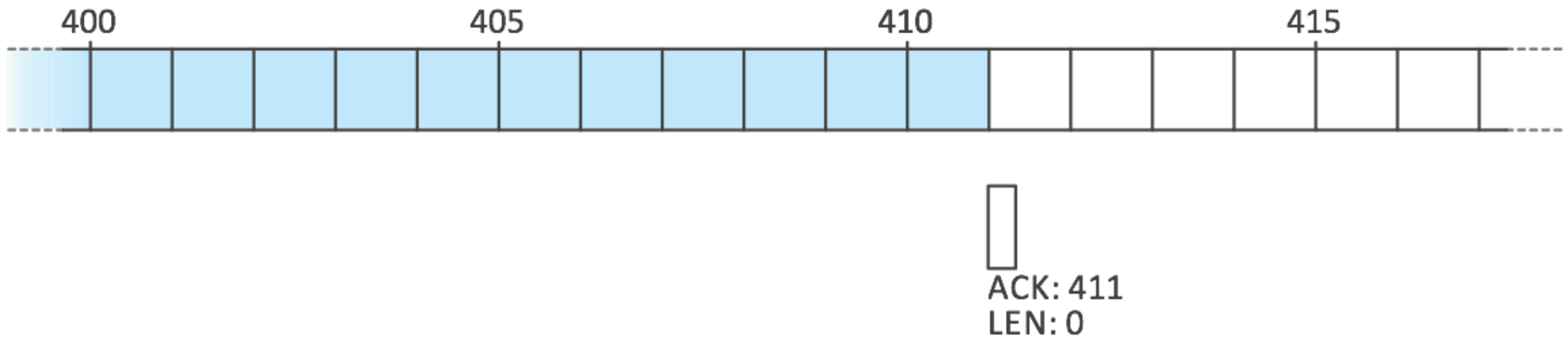
Transmission Control Protocol

- *What if the first packet is dropped in network?*
- Receiver always acknowledges with seq. no. of next expected byte
- Sender retransmits lost segment

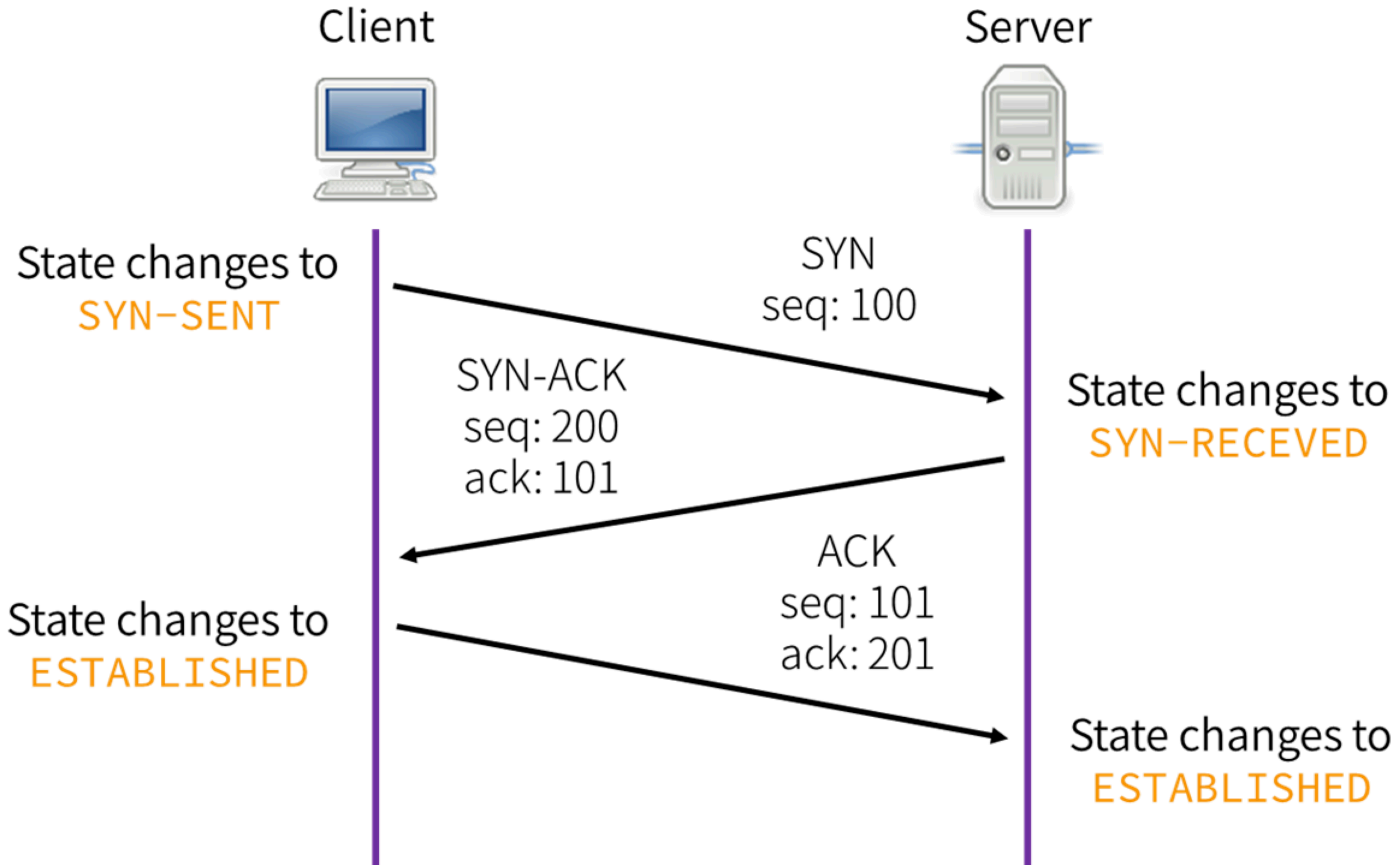


Transmission Control Protocol

- *What if the first packet is dropped in network?*
- Sender retransmits lost segment
- Receiver always acknowledges with seq. no. of next expected byte



TCP Three Way Handshake



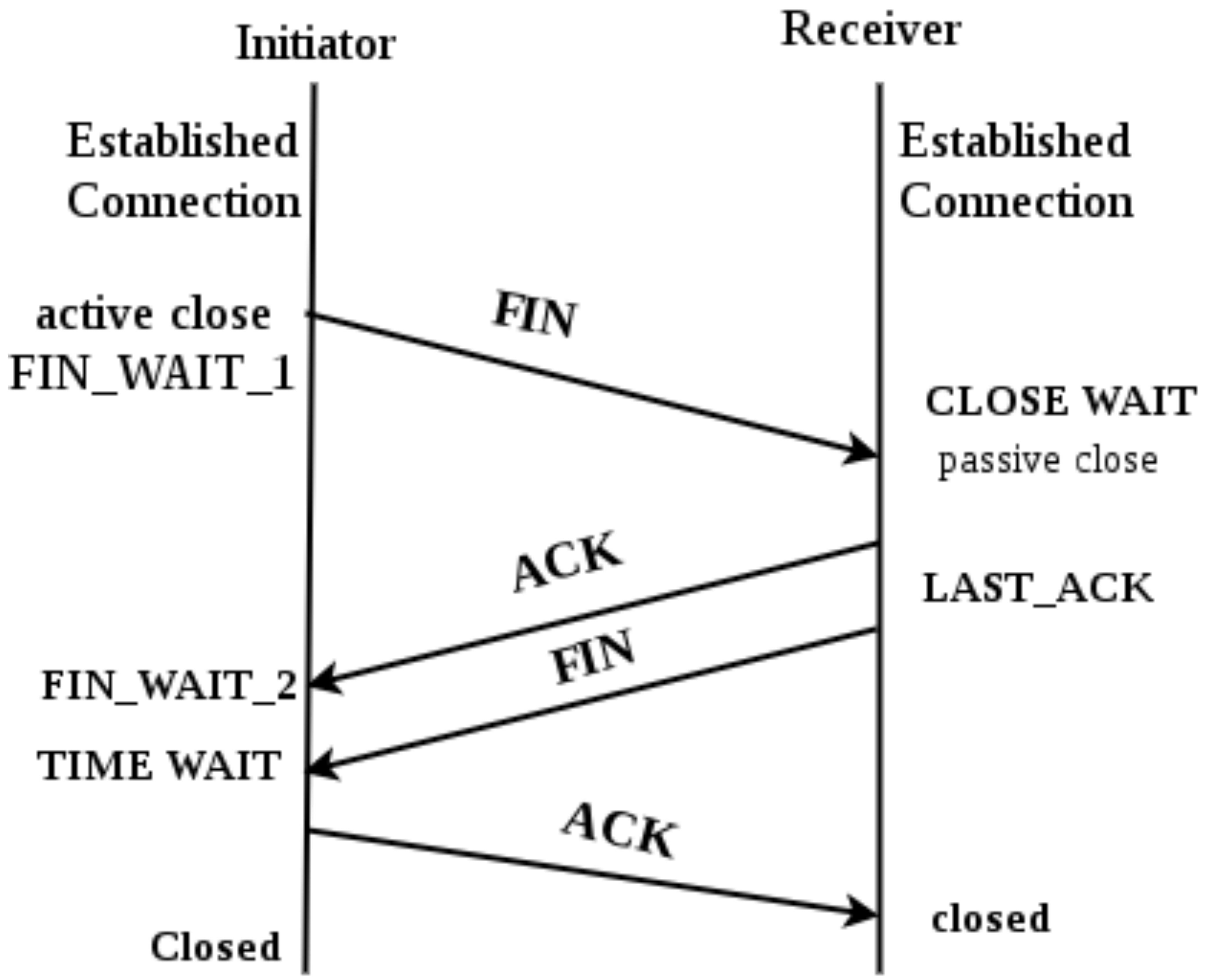
Ending a Connection

Sends packet with FIN flag set
  Must have ACK flag with valid seqnum

Peer receiving FIN packet acknowledges receipt of FIN packet with ACK

FIN "consumes" one byte of seq. number

Eventually other side sends packet with FIN flag set — terminates session



TCP Connection Reset

TCP designed to handle possibility of spurious TCP packets (e.g. from previous connections)

Packets that are invalid given current state of session generate a reset

- If a connection exists, it is torn down
- Packet with RST flag sent in response

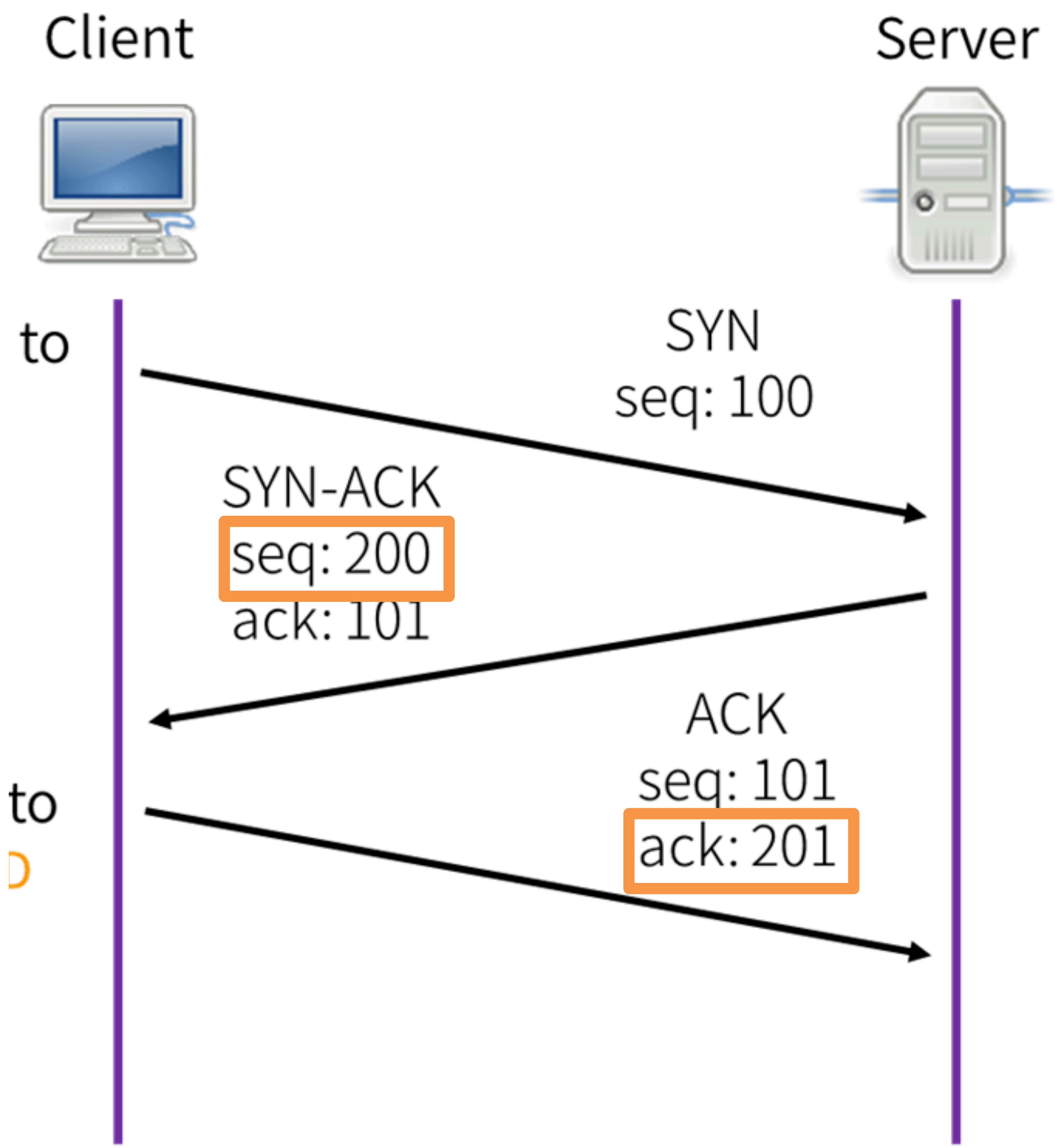
If a host receives a TCP packet with RST flag, it tears down the connection

TCP Connection Spoofing

Can we impersonate another host when *initiating* a connection?

Off-path attacker can send initial SYN to server …
… but cannot complete three-way handshake without seeing the server's sequence number

1 in 2^{32} chance to guess right if initial sequence number chosen uniformly at random



TCP Reset Attack

Can we reset an *existing* TCP connection?

Need to know port numbers (16 bits)
  Initiator's port number usually chosen random by OS
  Responder's port number may be well-known port of service

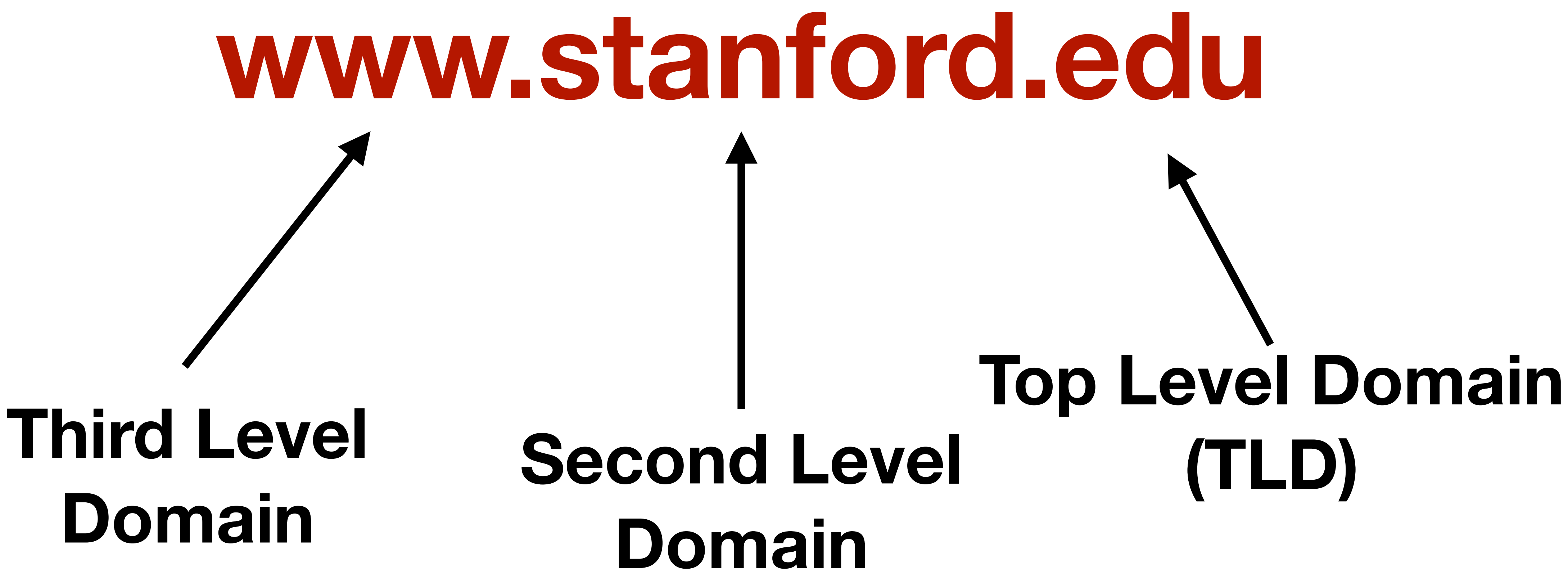
There is leeway in sequence numbers B will accept
  Must be within window size (32-64K on most modern OSes)

1 in $2^{16+32}/W$ (where W is window size) chance to guess right

DNS (Domain Name System)

Application-layer protocols (and people) usually refer to Internet host by host name (e.g., google.com)

DNS is a delegatable, hierarchical name space



DNS Record

A DNS server has a set of records it authoritatively knows about

```
$ dig bob.ucsd.edu

;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 30439
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 3, ADDITIONAL: 6

;; QUESTION SECTION:
;bob.ucsd.edu.          IN  A

;; ANSWER SECTION:
bob.ucsd.edu.     3600  IN  A   132.239.80.176

;; AUTHORITY SECTION:
ucsd.edu.         3600  IN  NS  ns0.ucsd.edu.
ucsd.edu.         3600  IN  NS  ns1.ucsd.edu.
ucsd.edu.         3600  IN  NS  ns2.ucsd.edu.
```

DNS Root Name Servers

In total, there are 13 main **DNS root servers**, each of which is named with the letters 'A' to 'M'.

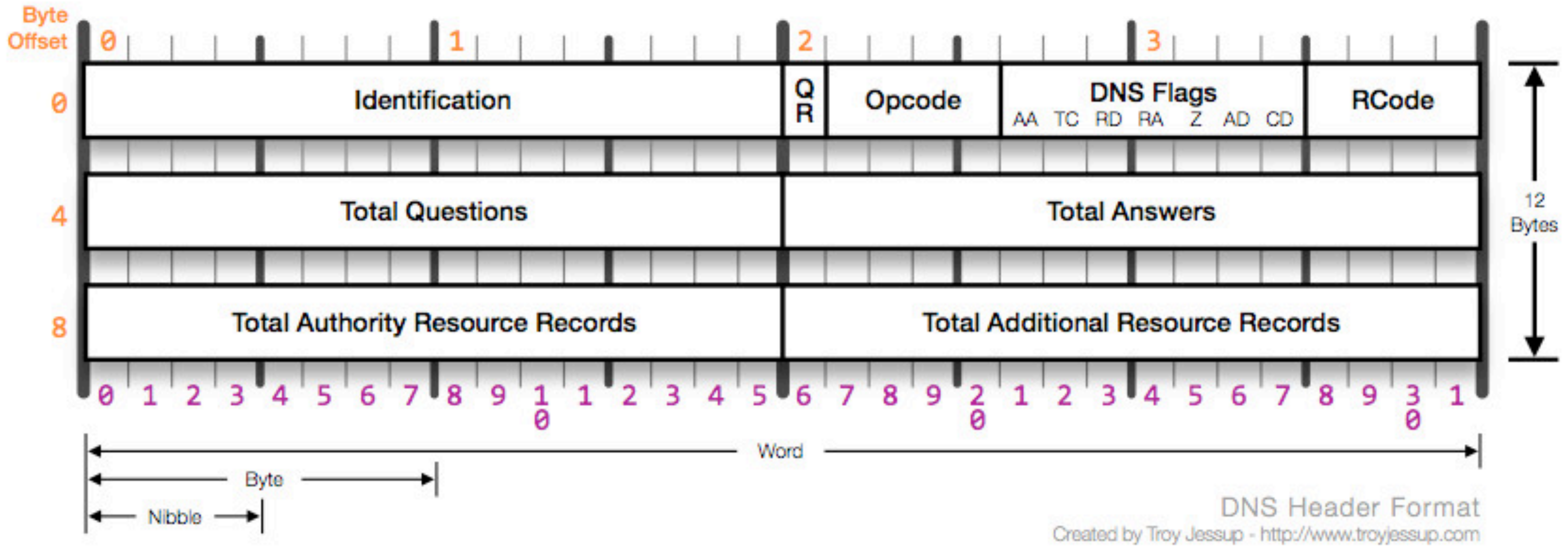
HOSTNAME	IP ADDRESSES	MANAGER
a.root-servers.net	198.41.0.4, 2001:503:ba3e::2:30	VeriSign, Inc.
b.root-servers.net	199.9.14.201, 2001:500:200::b	University of Southern California (ISI)
c.root-servers.net	192.33.4.12, 2001:500:2::c	Cogent Communications
d.root-servers.net	199.7.91.13, 2001:500:2d::d	University of Maryland
e.root-servers.net	192.203.230.10, 2001:500:a8::e	NASA (Ames Research Center)
f.root-servers.net	192.5.5.241, 2001:500:2f::f	Internet Systems Consortium, Inc.
g.root-servers.net	192.112.36.4, 2001:500:12::d0d	US Department of Defense (NIC)
h.root-servers.net	198.97.190.53, 2001:500:1::53	US Army (Research Lab)
i.root-servers.net	192.36.148.17, 2001:7fe::53	Netnod
j.root-servers.net	192.58.128.30, 2001:503:c27::2:30	VeriSign, Inc.
k.root-servers.net	193.0.14.129, 2001:7fd::1	RIPE NCC
l.root-servers.net	199.7.83.42, 2001:500:9f::42	ICANN
m.root-servers.net	202.12.27.33, 2001:dc3::35	WIDE Project

DNS Packet

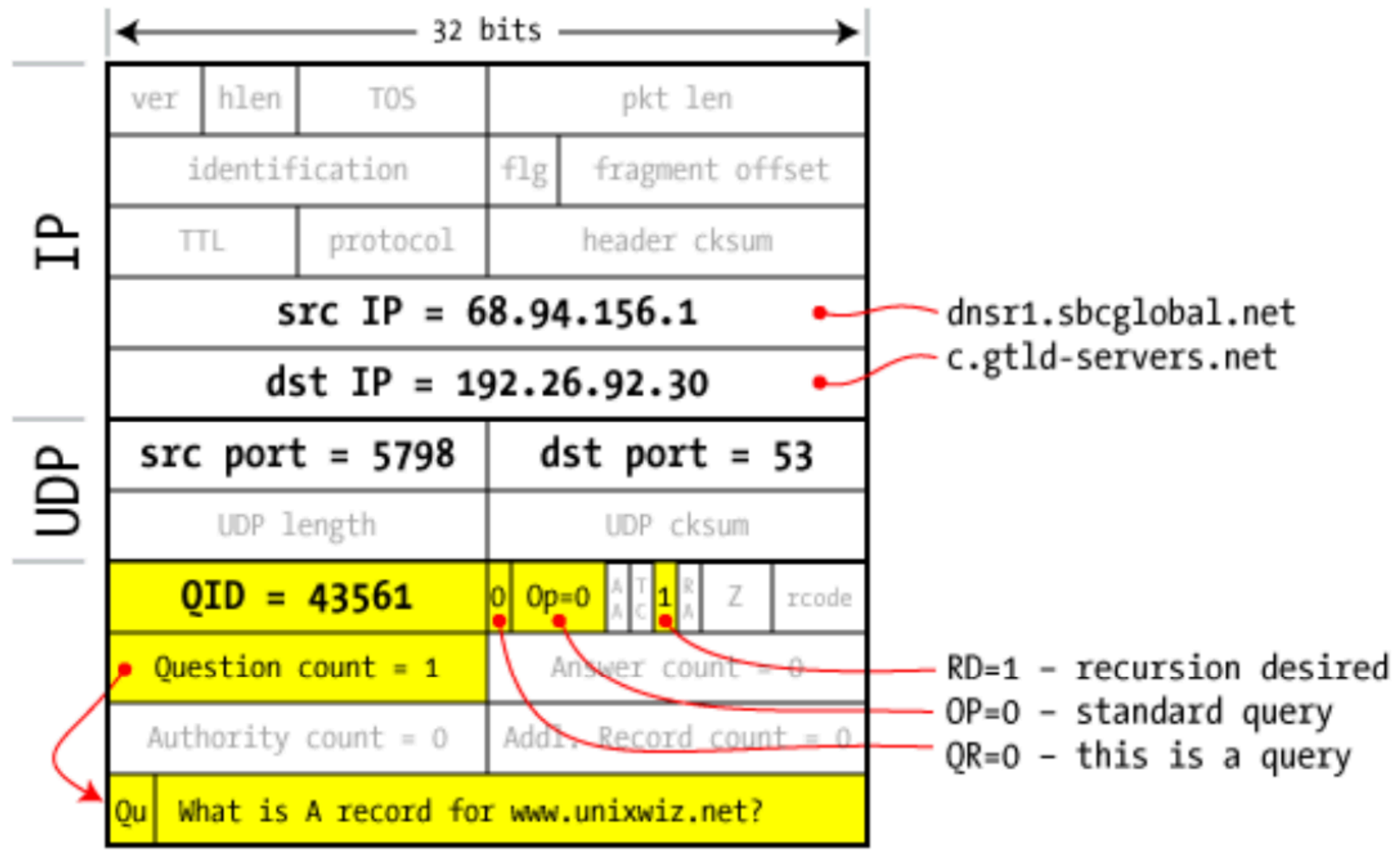
DNS requests sent over UDP

Four sections: questions, answers, authority, additional records

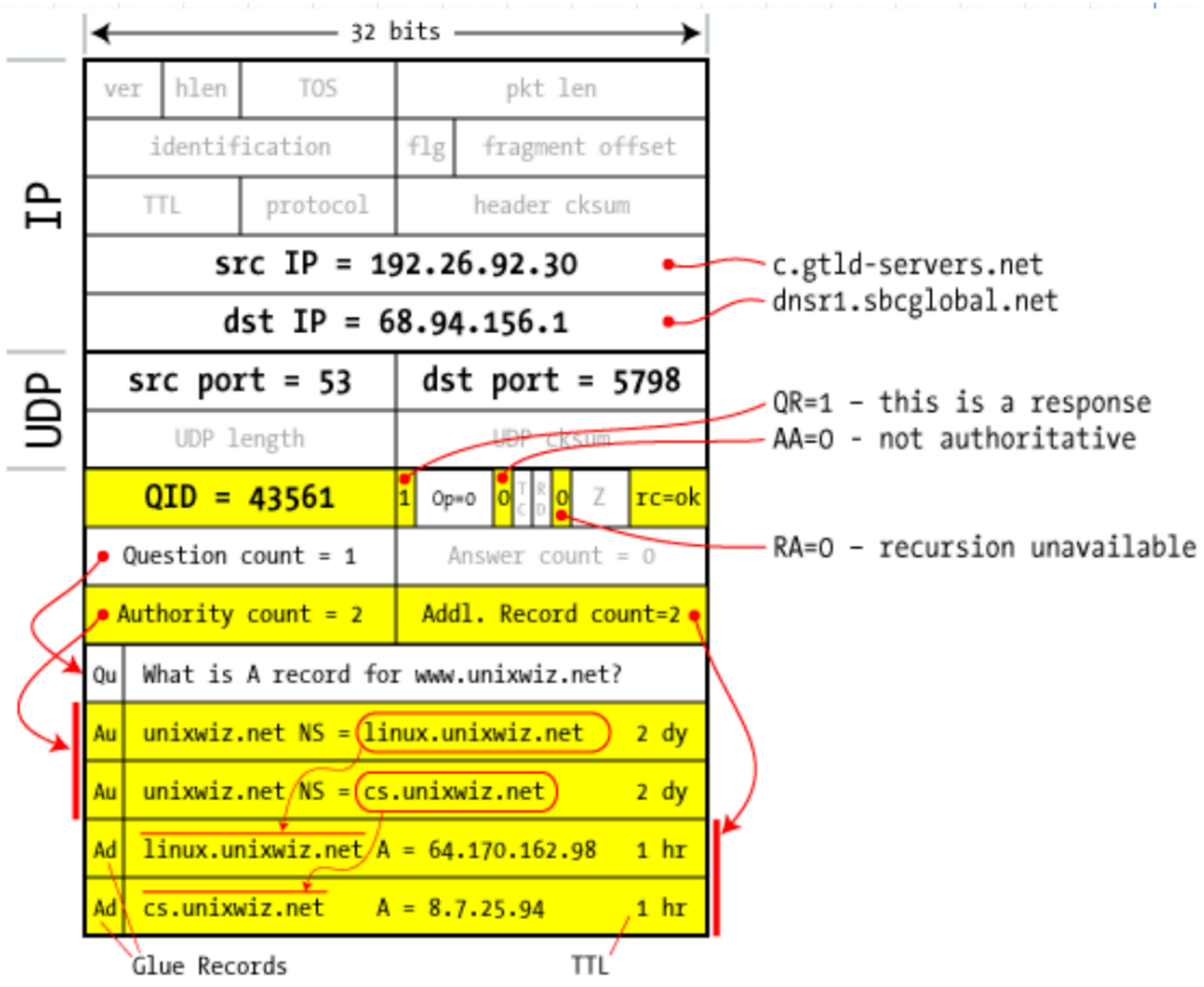
Query ID:
16 bit random value
Links response to query



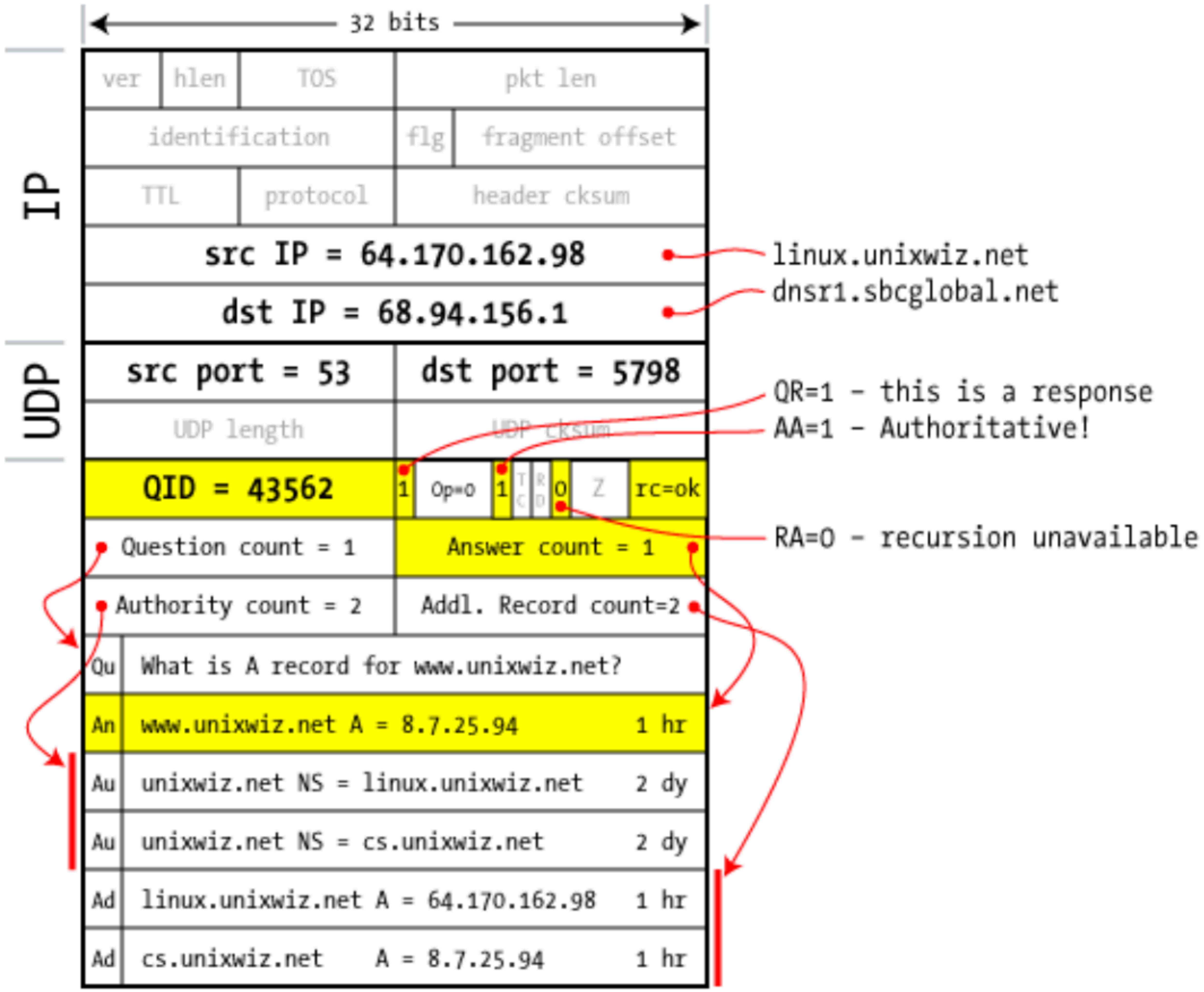
Request


32 bits
IP
UDP
ver
hlen
TOS
pkt len
identification
flg
fragment offset
TTL
protocol
header cksum
src IP = 68.94.156.1
dst IP = 192.26.92.30
dnsr1.sbcglobal.net
c.gtld-servers.net
src port = 5798
dst port = 53
UDP length
UDP cksum
QID = 43561
0
Op=0
AA
TC
1
RA
Z
rcode
Question count = 1
Answer count = 0
Authority count = 0
Addl. Record count = 0
Qu
What is A record for www.unixwiz.net?
RD=1 - recursion desired
OP=0 - standard query
QR=0 - this is a query

Response


32 bits
IP
UDP
ver
hlen
TOS
pkt len
identification
flg
fragment offset
TTL
protocol
header cksum
src IP = 192.26.92.30
dst IP = 68.94.156.1
src port = 53
dst port = 5798
UDP length
UDP cksum
QID = 43561
1
Op=0
0
0
Z
rc=ok
Question count = 1
Answer count = 0
Authority count = 2
Addl. Record count=2
Qu
What is A record for www.unixwiz.net?
Au
unixwiz.net NS = linux.unixwiz.net
2 dy
Au
unixwiz.net NS = cs.unixwiz.net
2 dy
Ad
linux.unixwiz.net A = 64.170.162.98
1 hr
Ad
cs.unixwiz.net A = 8.7.25.94
1 hr
Glue Records
TTL
c.gtld-servers.net
dnsr1.sbcglobal.net
QR=1 - this is a response
AA=0 - not authoritative
RA=0 - recursion unavailable

Authoritative Response



DNS Security

Users/hosts trust the host-address mapping provided by DNS

- Used as basis for many security policies:
  - Browser same origin policy, URL address bar

Interception of requests or compromise of DNS servers can result in incorrect or malicious responses

Caching

DNS responses are cached

- Quick response for repeated translations
- NS records for domains also cached

DNS negative queries are cached

- Save time for nonexistent sites, e.g. misspelling

Cached data periodically times out

- Lifetime (TTL) of data controlled by owner of data
- TTL passed with every record

DNS Spoofing

Scenario: DNS client issues query to server

Attacker would like to inject a fake reply
  Attacker does not see query or real response

How does client authenticate response?

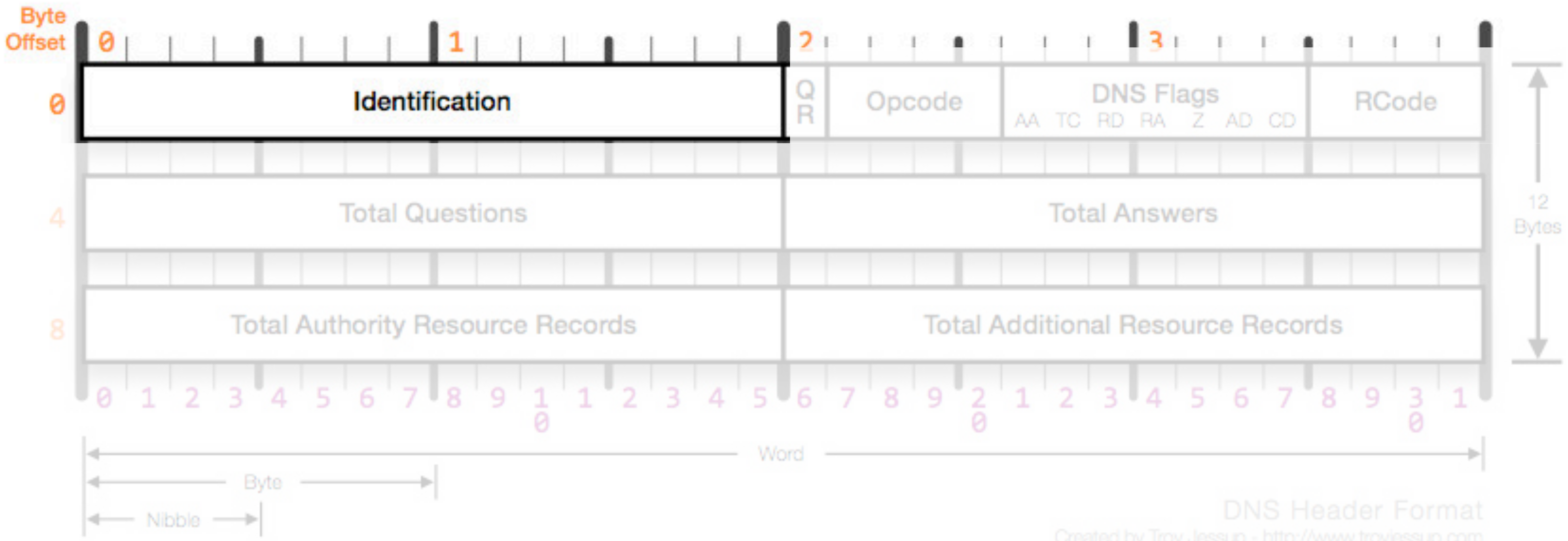
DNS Spoofing

How does client authenticate response?

UDP port numbers must match

Destination port usually port 53 by convention

16-bit query ID must match

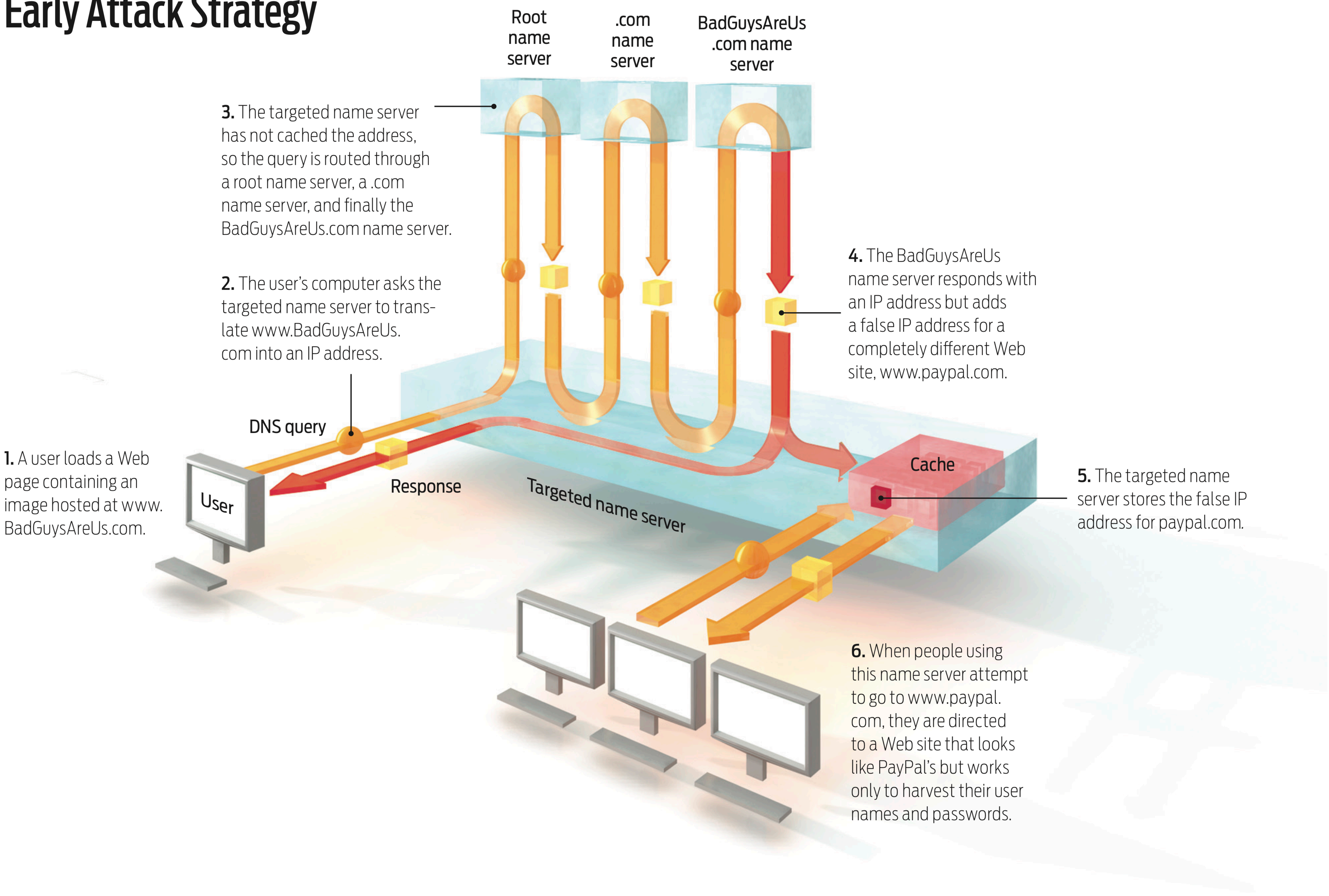


DNS Cache Poisoning

DNS query results include Additional Records section
- Provide records for anticipated next resolution step

Early servers accepted and cached all additional records provided in query response

Early Attack Strategy



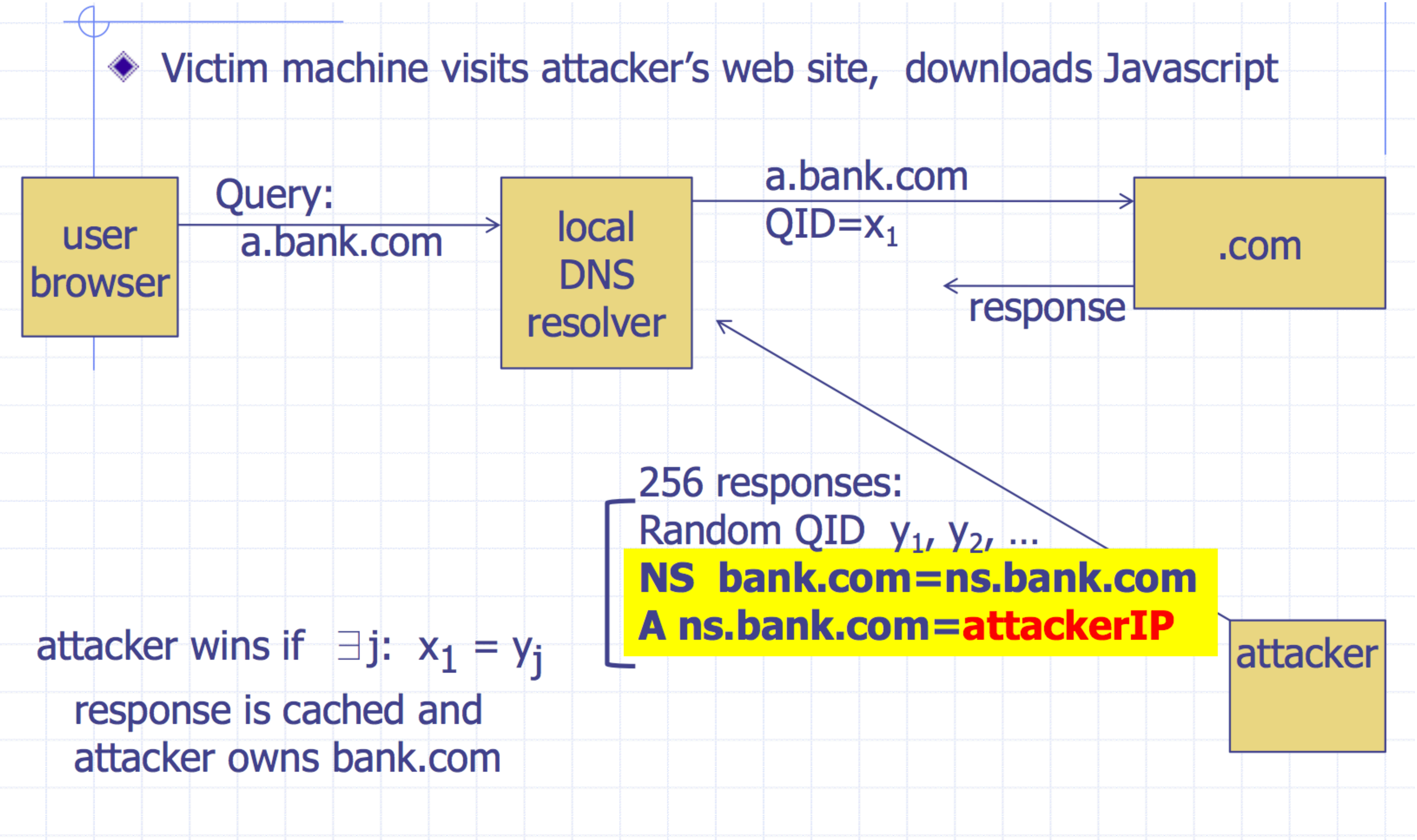
Glue Records

Can we just stop using additional section?
– Only accept answers from authoritative servers?

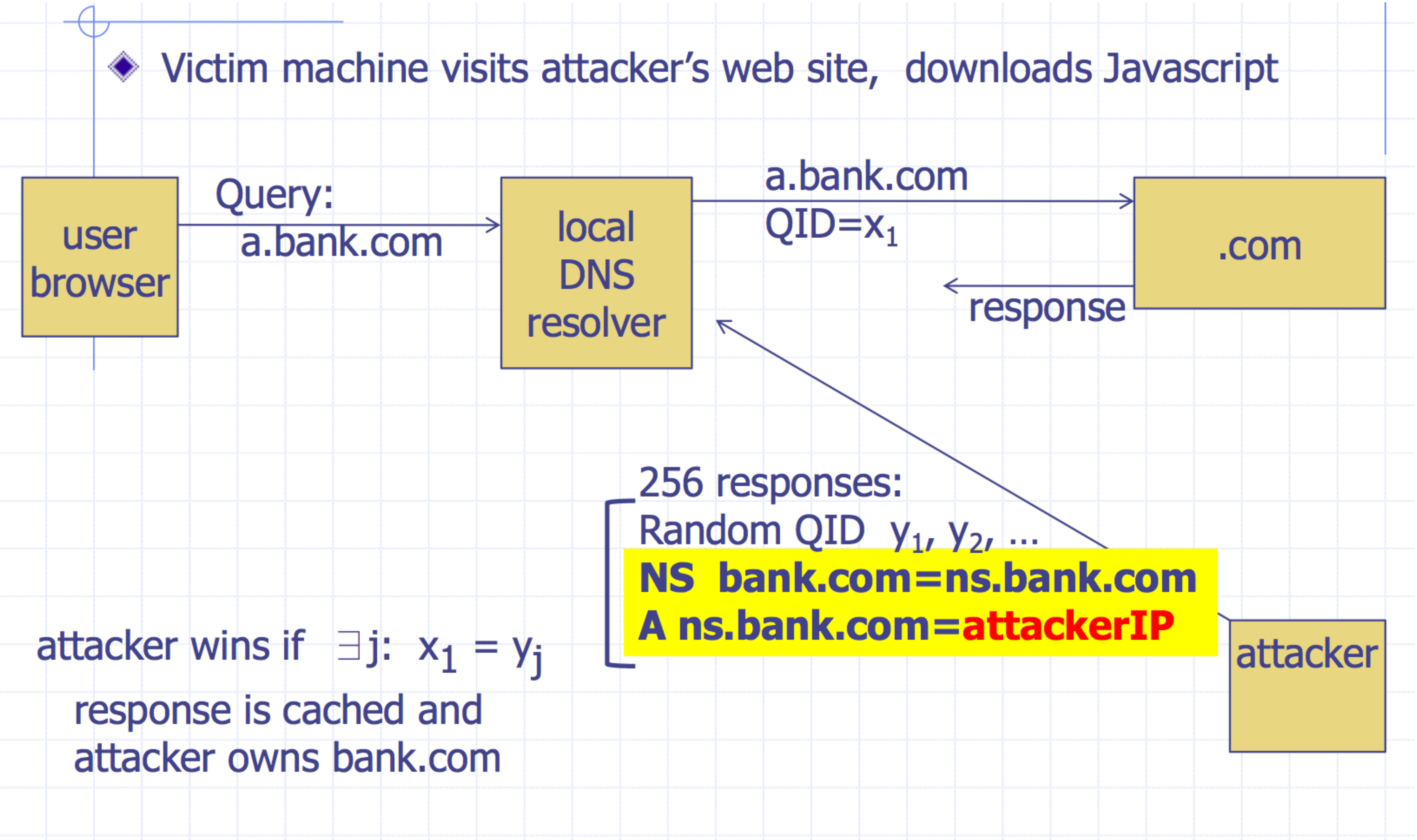
Glue records: non-authoritative are records necessary to contact next hop in resolution chain
– Necessary given current design of DNS

Bailiwick Checking: Only accept additional records that are for a domain in the original question.

Kaminsky Attack



Try Again!


Victim machine visits attacker's web site, downloads Javascript
user browser
Query:
a.bank.com
local DNS resolver
a.bank.com
QID=x_1
.com
response
256 responses:
Random QID y_1, y_2, …
NS bank.com=ns.bank.com
A ns.bank.com=attackerIP
attacker
attacker wins if $\exists j$: $x_1 = y_j$
response is cached and
attacker owns bank.com

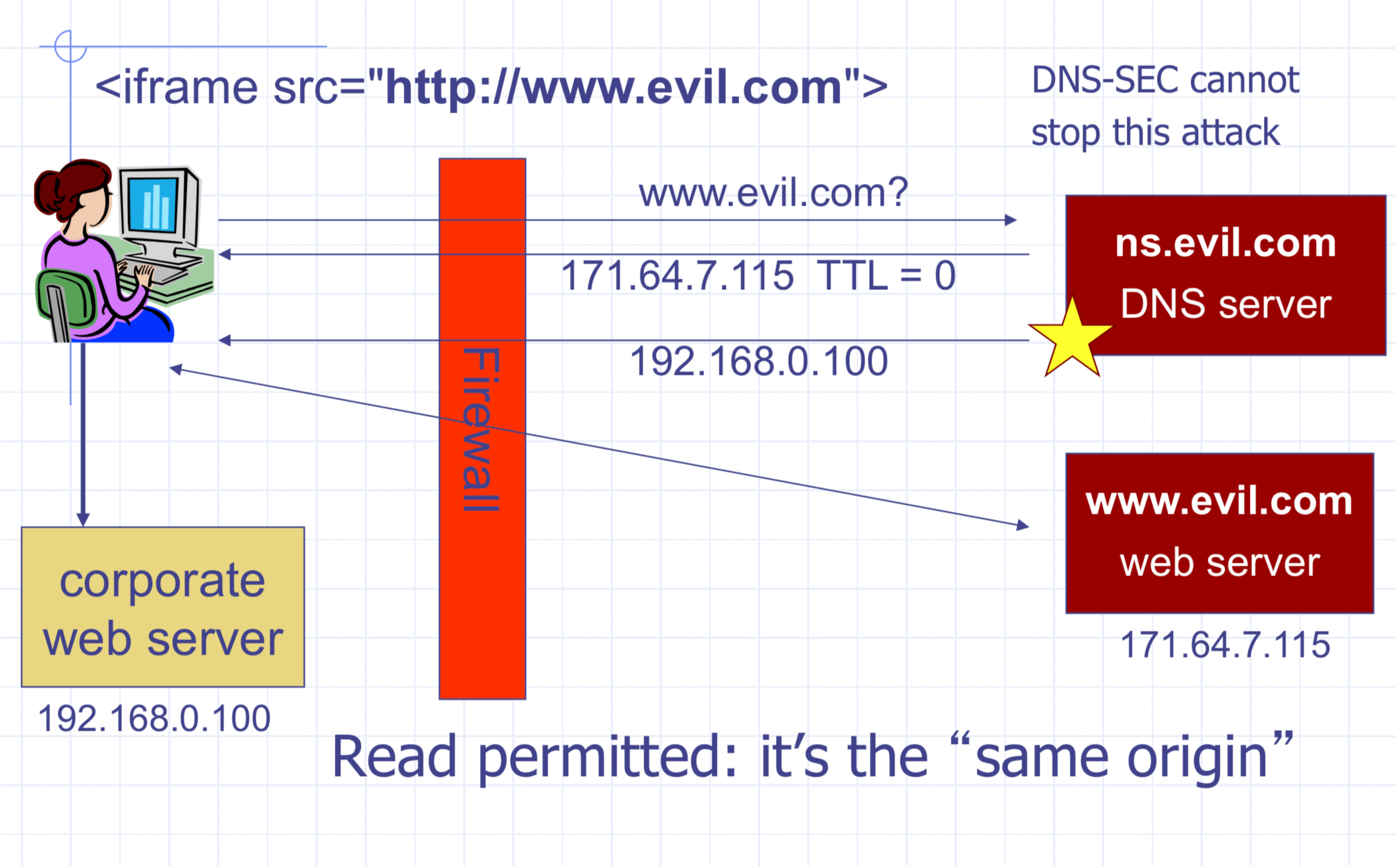
Defenses

Increase QueryID space. But how? Don’t want to change packet.

Randomize src port, additional 11 bits of entropy

 - Attack now takes several hours

DNS Rebinding



Rebinding Defenses

Browser Mitigations:

- Refuse to switch IPs mid session
- Interacts poorly with proxies, VPNs, CDNs, etc
- Not consistently implemented in any browser

Server Defenses

- Check Host header for unrecognized domains
- Authenticate users with something else beyond IP address

DNSSEC

Adds authentication and integrity to DNS responses

Authoritative DNS servers sign DNS responses using cryptographic key

Clients can verify that a response is legitimate by checking signature through PKI similar to HTTPS

Most people don’t use DNSSEC and never will. Use TLS instead.

Network Security Takeaway

Assume the network is out to get you.

If you want any guarantee of any security, use TLS.

Denial of Service Attacks

Goal: take large site offline by overwhelming it with network traffic such that they can't process real requests

How: find mechanism where attacker doesn't have to spend a lot of effort, but requests are difficult/expensive for victim to process

Types of Attacks

DoS Bug: design flaw that allows one machine to disrupt a service. Generally a protocol asymmetry, e.g., easy to send request, difficult to create response. Or requires server state.

DoS Flood: control a large number of requests from a botnet of machines you control

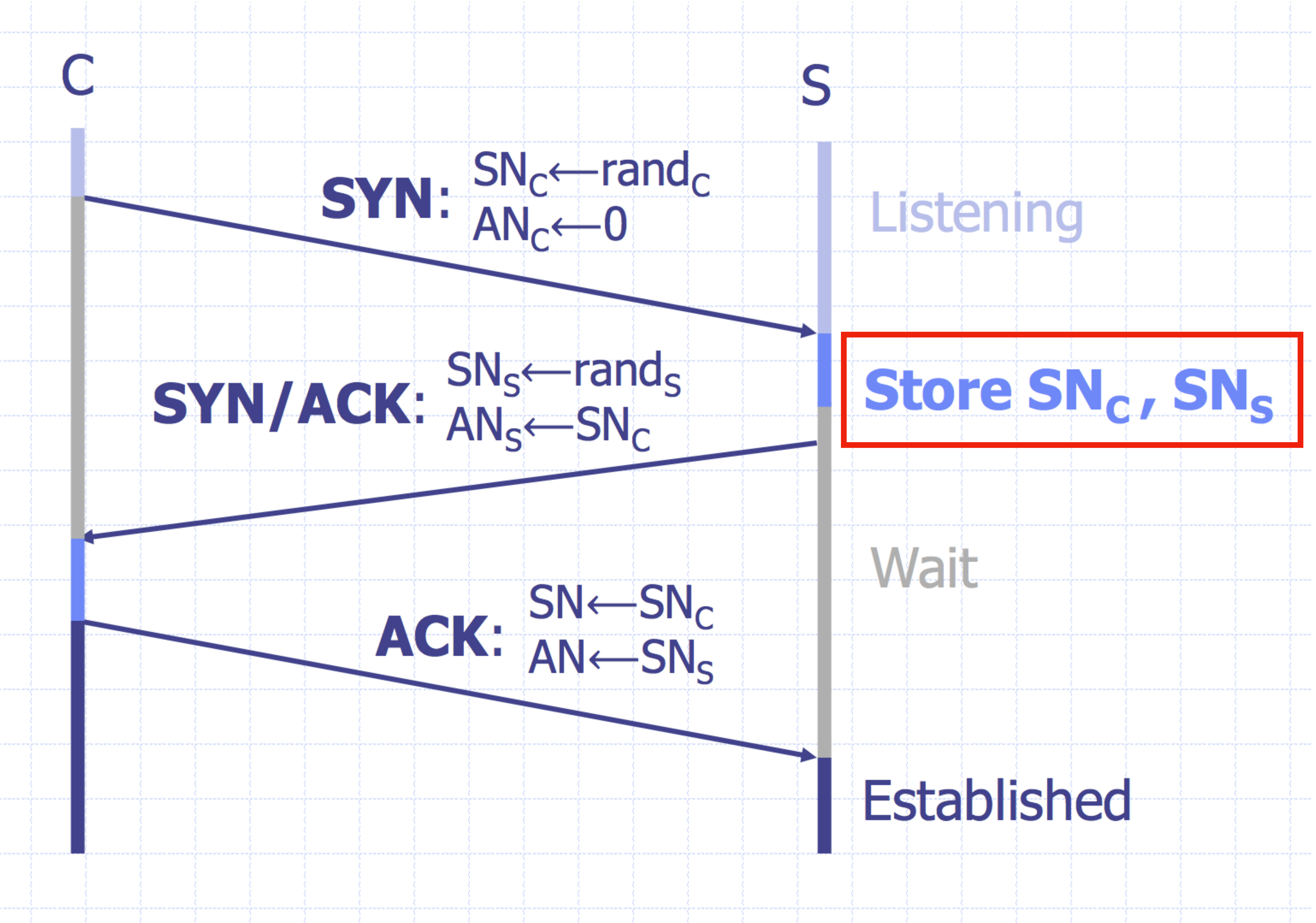
Possible at Every Layer

Link Layer: send too much traffic for switches/routers to handle

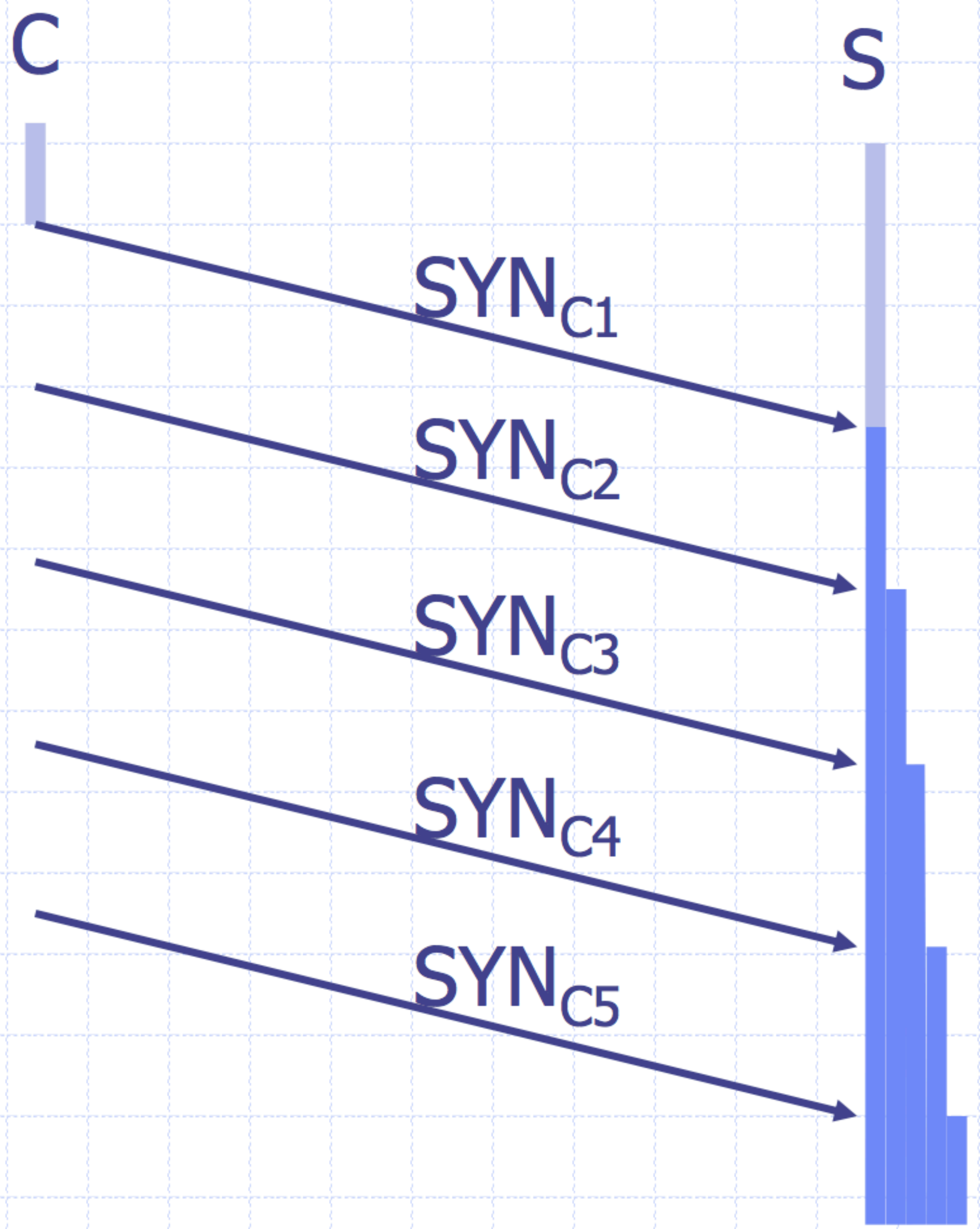
TCP/UDP: require servers to maintain large number of concurrent connections or state

Application Layer: require servers to perform expensive queries or cryptographic operations

TCP Handshake



SYN Floods



Single machine:

- SYN Packets with **random source IP addresses**
- Fills up backlog queue on server
- No further connections possible

Core Problem

Problem: server commits resources (memory) before confirming identify of client (when client responds)

Bad Solution:

- Increase backlog queue size
- Decrease timeout

Real Solution: Avoid state until 3-way handshake completes

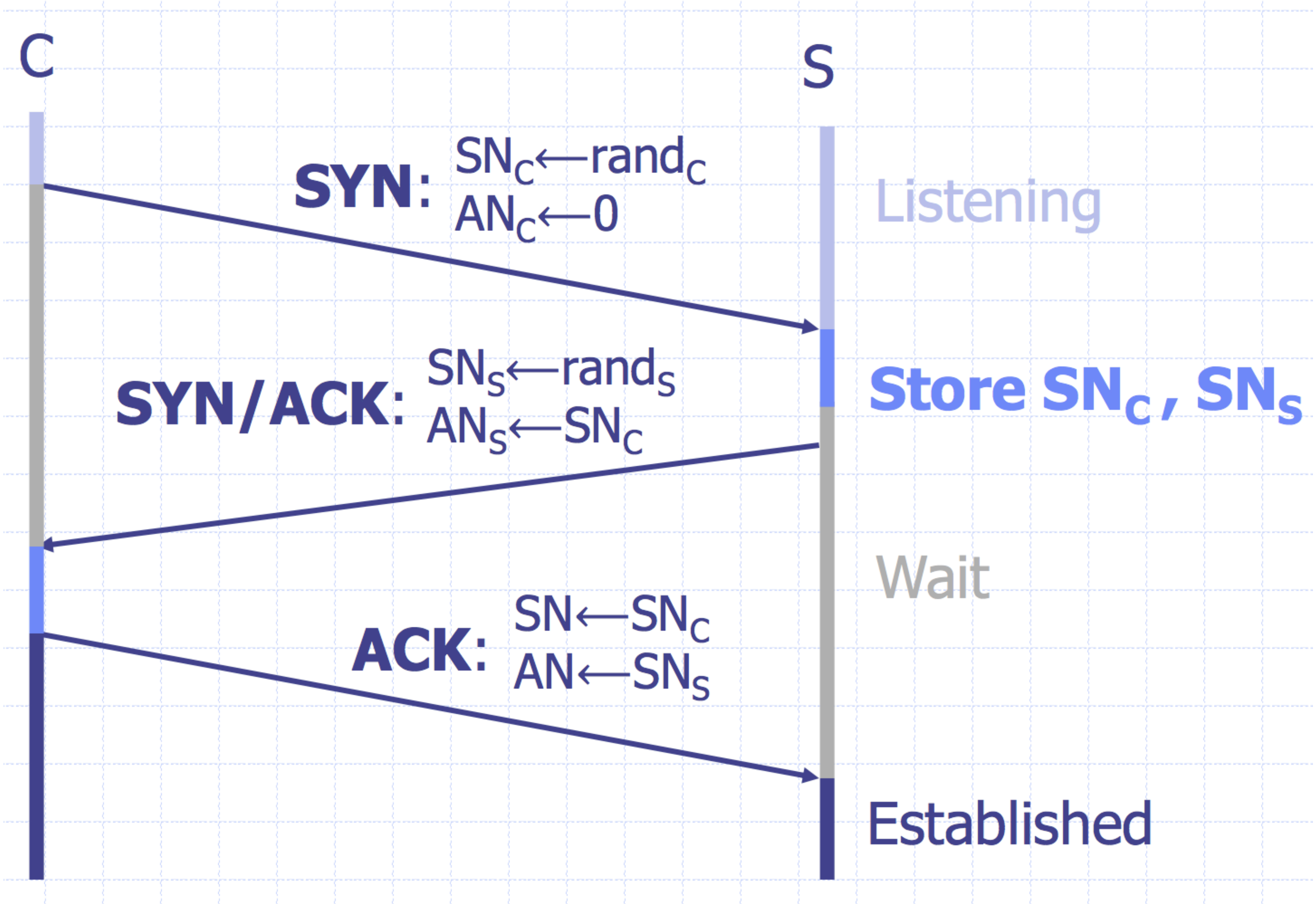
SYN Cookies

Idea: Instead of storing SNc and SNs…
send a cookie back to the client.

L = MAC_{key} (SAddr, SPort, DAddr, DPort, SN_C, T)
key: picked at random during boot

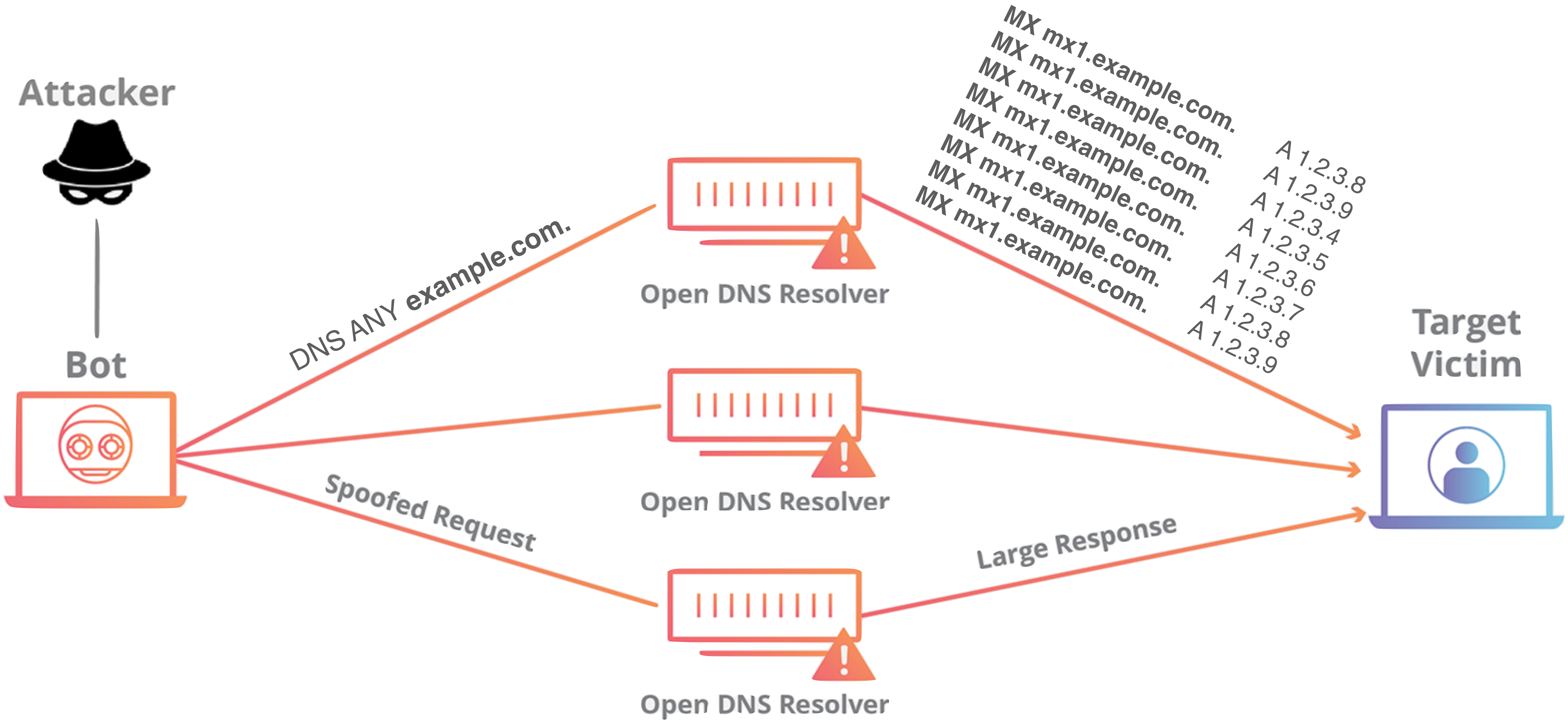
T = 5-bit counter incremented every 64 secs.
SN_s = (T || mss || L)

Honest client sends ACK (AN=SN_s , SN=SN_C+1)
Server allocates space for socket only if valid SNs



Server does not save state
(loses TCP options)

Amplification Attacks



60-70x Increase in Size

Image: Cloudflare

Common UDP Amplifiers

DNS: ANY query returns *all* records server has about a domain

NTP: MONLIST returns list of last 600 clients who asked for the time recently

Only works if you can receive a big response by sending a single packet — otherwise spoofing doesn’t help you.

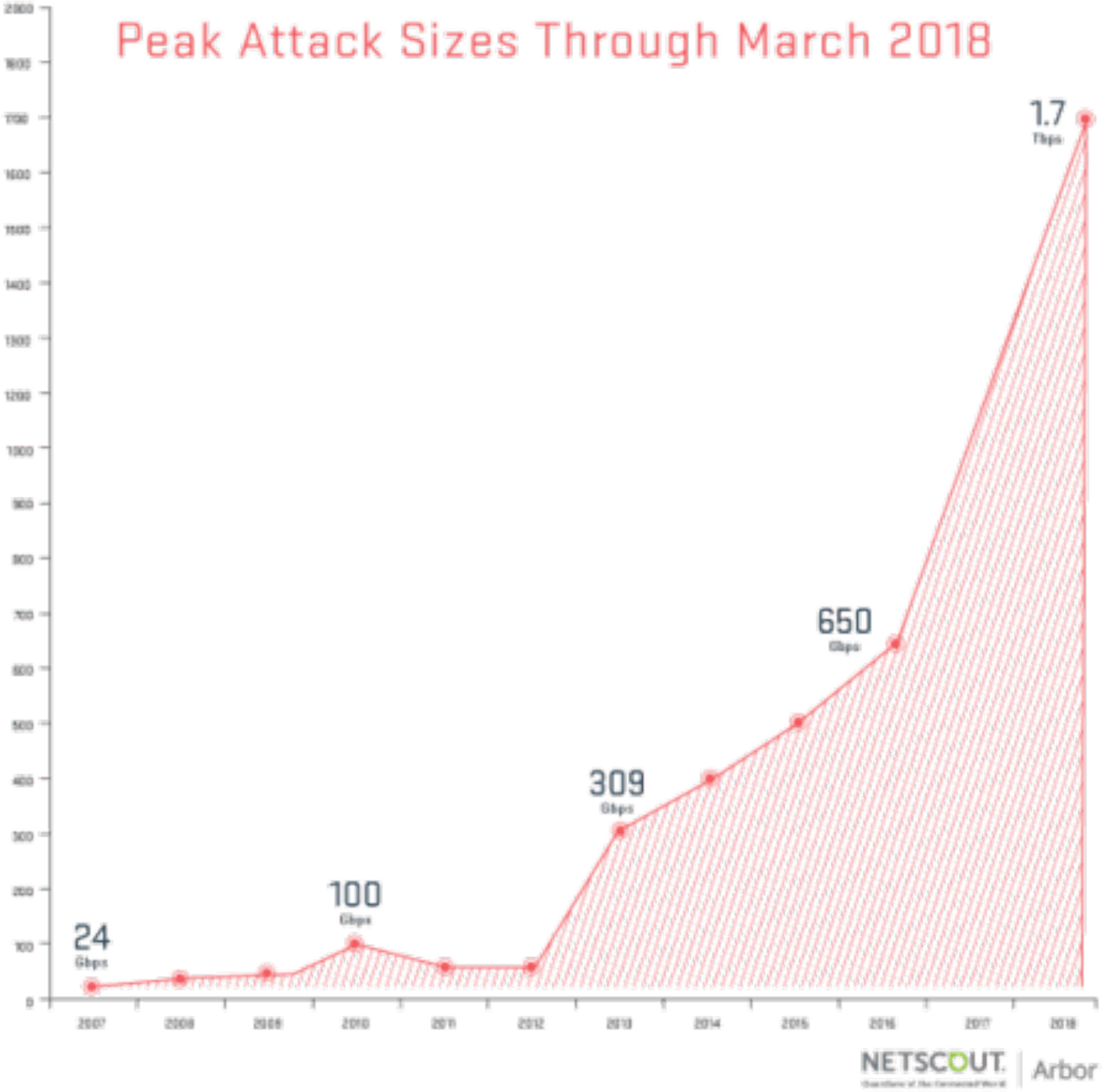
Amplification Attacks

2013: DDoS attack generated 300 Gbps (DNS)

- 31,000 misconfigured open resolvers, each at 10 Mbps
- Source: 3 networks that allowed IP spoofing

2014: 400 Gbps DDoS attacked used 4500 NTP servers

Memcache



Memcache: retrieve large record

The server responds by firing back as much as 50,000 times the data it received.

THE WALL STREET JOURNAL.

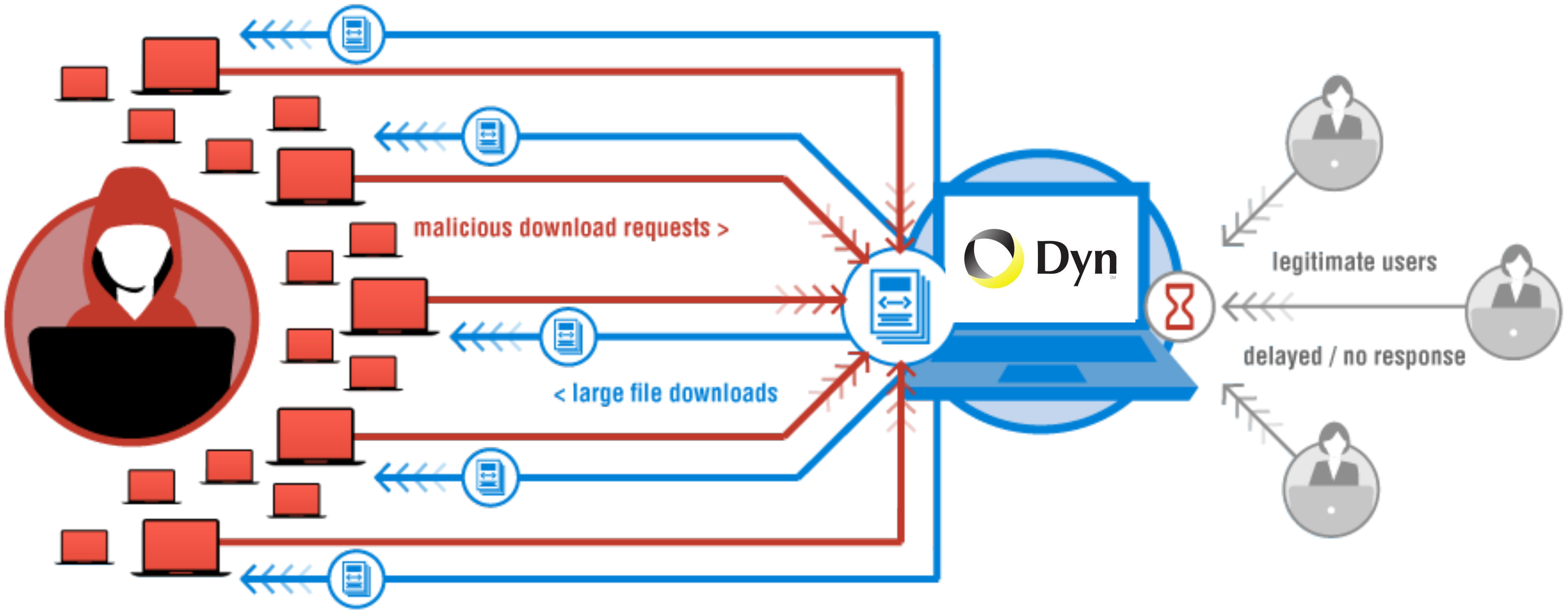
October 21, 2016

Cyberattack Knocks Out Access to Websites

Popular sites such as Twitter, Netflix and PayPal were unreachable for part of the day

twitter amazon web services PayPal NETFLIX SOUNDCLOUD

Spotify GitHub reddit New York Times



> “We are still working on analyzing the data but the estimate at the time of this report is up to 100,000 malicious endpoints. […] There have been some reports of a magnitude in the 1.2 Tbps range; at this time we are unable to verify that claim.”

Image: Verisign

A Botnet of IoT Devices

≈ ~~200K Hosts~~

200K IoT devices

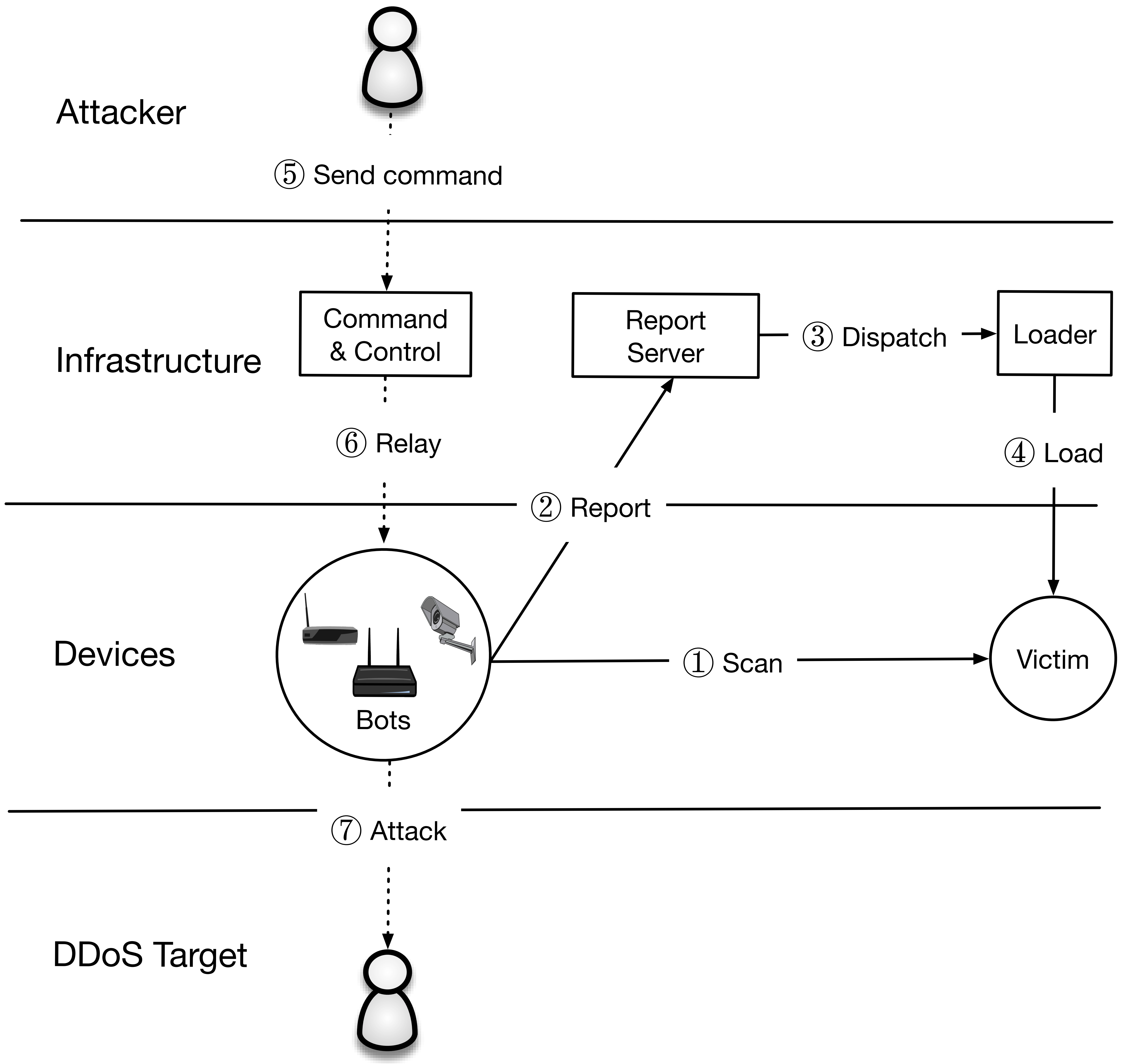
Not Amplification.
Flood with SYN, ACK, UDP, and GRE packets

The Mirai Malware

5-7. Later, the **bot master** will issue commands to pause scanning and to start an attack

Attack Command:

- Action (e.g., START, STOP)
- Target IP(s)
- Attack Type (e.g., GRE, DNS, TCP)
- Attack Duration

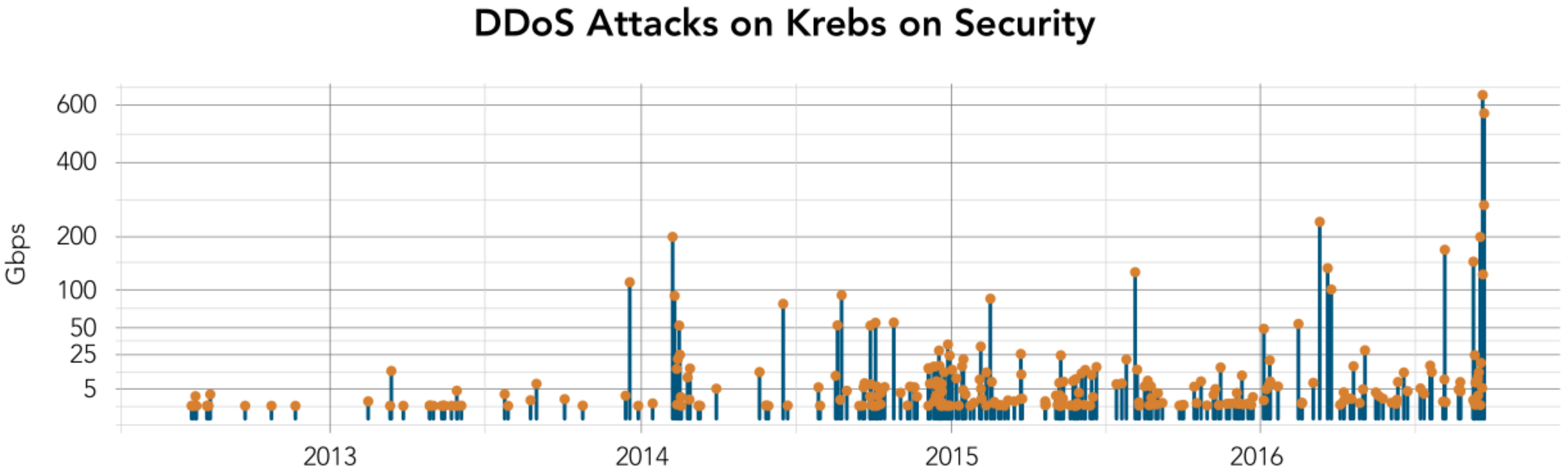


Password Guessing

Password	Device Type	Password	Device Type	Password	Device Type
123456	ACTi IP Camera	klv1234	HiSilicon IP Camera	1111	Xerox Printer
anko	ANKO Products DVR	jvbzd	HiSilicon IP Camera	Zte521	ZTE Router
pass	Axis IP Camera	admin	IPX-DDK Network Camera	1234	Unknown
888888	Dahua DVR	system	IQinVision Cameras	12345	Unknown
666666	Dahua DVR	meinsm	Mobotix Network Camera	admin1234	Unknown
vizxv	Dahua IP Camera	54321	Packet8 VOIP Phone	default	Unknown
7ujMko0vizxv	Dahua IP Camera	00000000	Panasonic Printer	fucker	Unknown
7ujMko0admin	Dahua IP Camera	realtek	RealTek Routers	guest	Unknown
666666	Dahua IP Camera	1111111	Samsung IP Camera	password	Unknown
dreambox	Dreambox TV Receiver	xmhdipc	Shenzhen Anran Camera	root	Unknown
juantech	Guangzhou Juan Optical	smcadmin	SMC Routers	service	Unknown
xc3511	H.264 Chinese DVR	ikwb	Toshiba Network Camera	support	Unknown
OxhlwSG8	HiSilicon IP Camera	ubnt	Ubiquiti AirOS Router	tech	Unknown
cat1029	HiSilicon IP Camera	supervisor	VideoIQ	user	Unknown
hi3518	HiSilicon IP Camera	<none>	Vivotek IP Camera	zlxx.	Unknown
klv123	HiSilicon IP Camera				

Mirai Population

~600K devices compromised



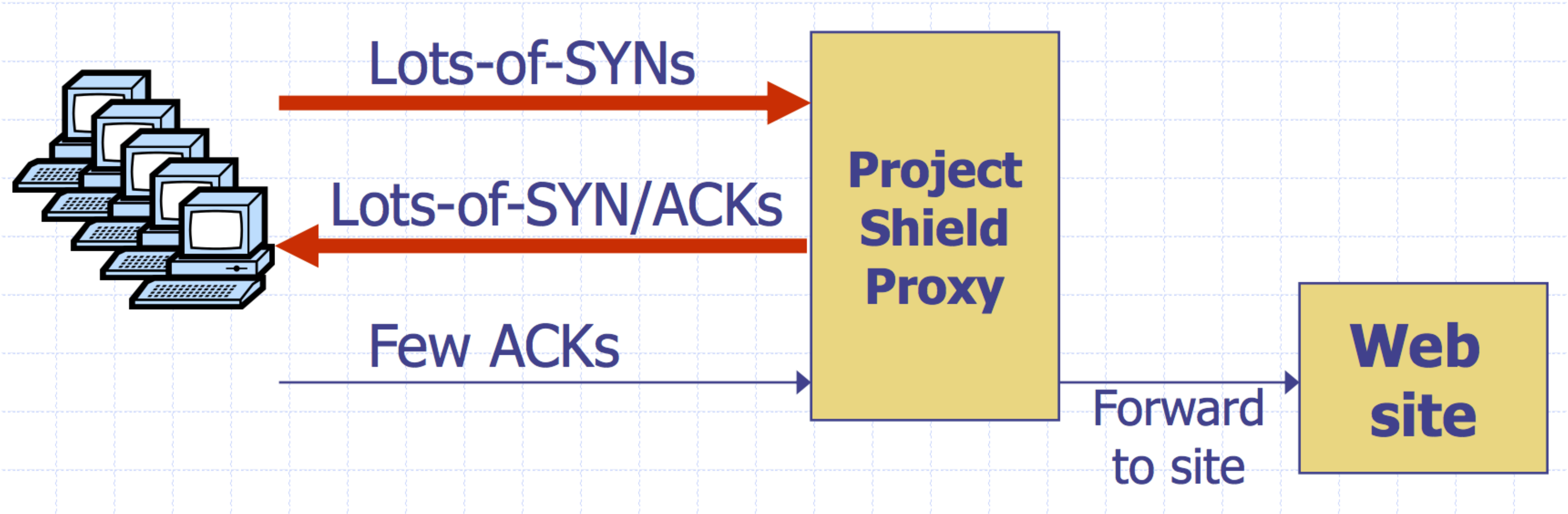
> “The magnitude of the attacks seen during the final week were significantly larger than the majority of attacks Akamai sees on a regular basis. […] In fact, while the attack on September 20 was the largest attack ever mitigated by Akamai, the attack on September 22 would have qualified for the record at any other time, peaking at 555 Gbps.”

Source: 2017 Akamai State of the Internet

Google Project Shield

DDoS Attacks are often used to censor content. In the case of Mirai, Brian Kreb's blog was under attack.

Google Project shield uses Google bandwidth to shield vulnerable websites (e.g., news, blogs, human rights orgs)



Moving Up Stack: GET Floods

Command bot army to:
* Complete real TCP connection
* Complete TLS Handshake
* GET large image or other content

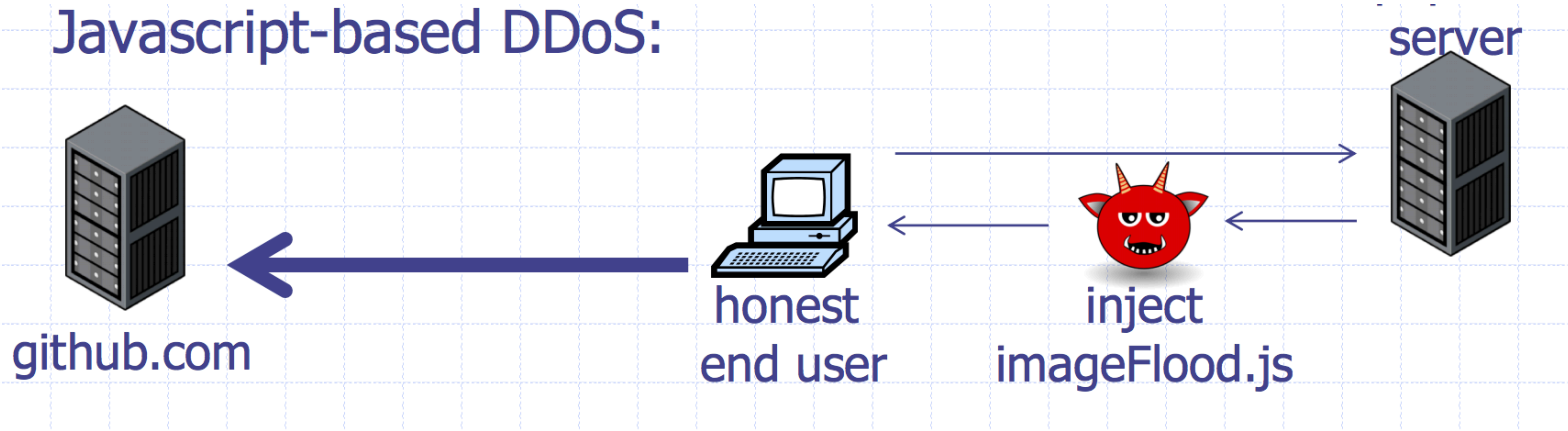
Will bypass flood protections…. but attacker can no longer use random source IPs

Victim site can block or rate limit bots

Github Attacks

1.35 Tbps attack against Github caused by javascript injected into HTTP web requests

The Chinese government was widely suspected to be behind the attack



Client Puzzles

Idea: What if we force every client to do moderate amount of work for every connection they make?

Example:

1) Server Sends: C

2) Client: find X s.t. $LSB_n(SHA\text{-}1(C||X)) = 0^n$

Assumption:

Puzzle takes 2^n for the client to compute (0.3 s on 1Ghz core)

Solution is trivial for server to check (single SHA-1)

Client Puzzles

Not frequently used in the real world

Benefits:

* Can change *n* based on amount of attack traffic

Limitations:

* Requires changes to both protocols, clients, and servers
* Hurts low power legitimate clients during attack (e.g., phones)