

CS165 – Computer Security

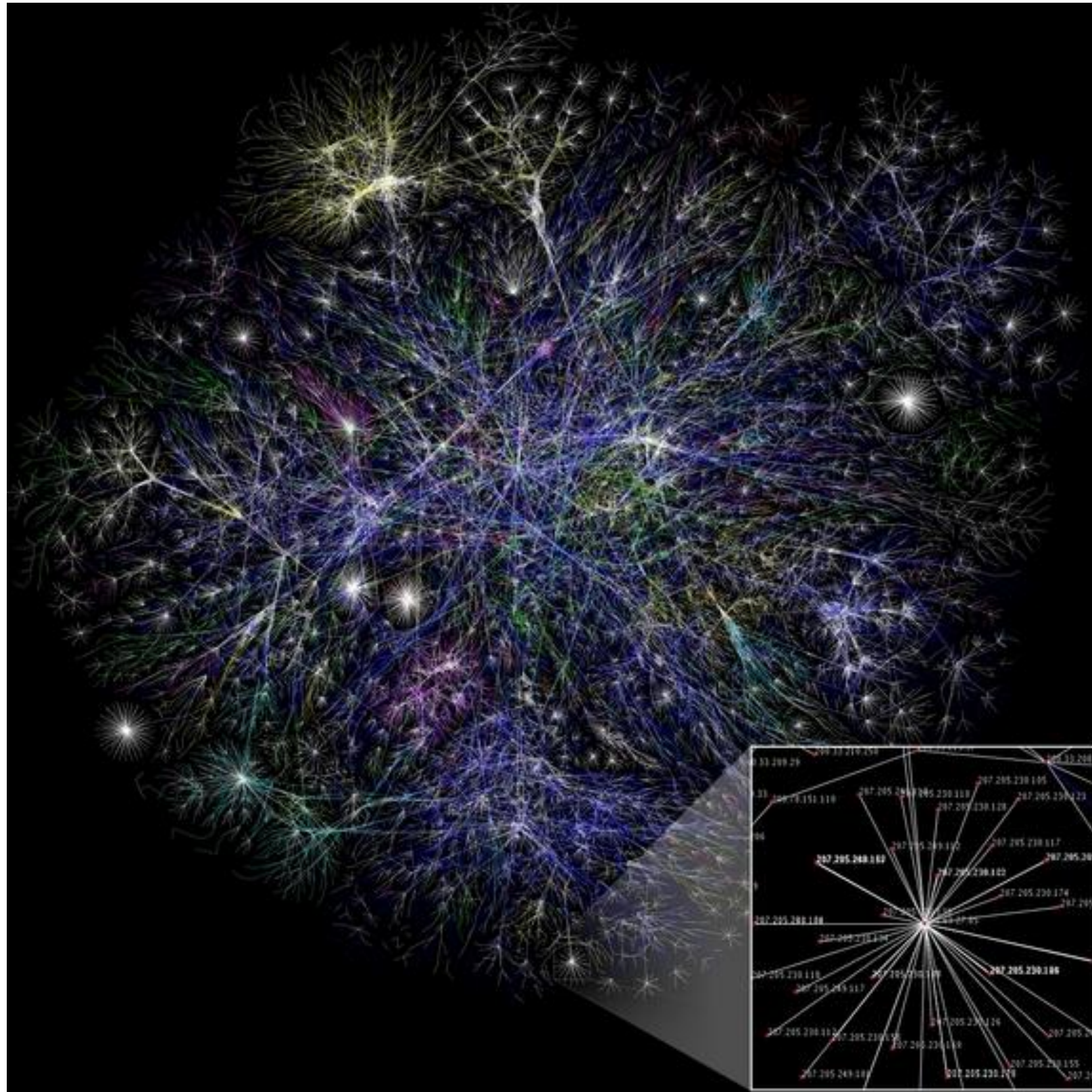
Network security

Nov 18, 2021

History of Network Security

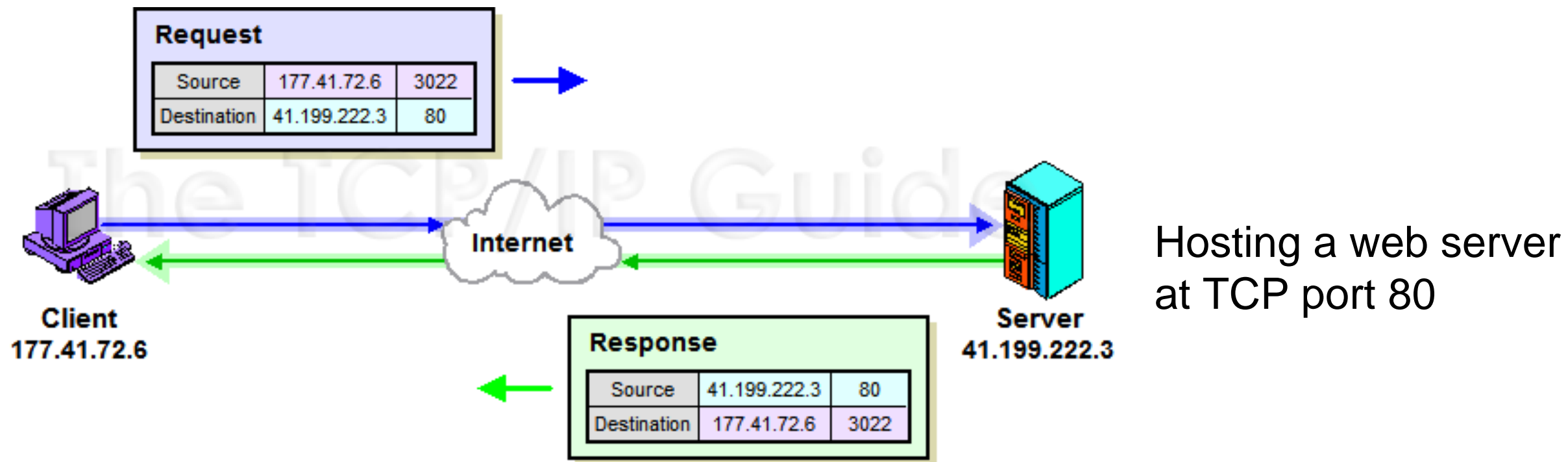
- Initially built for communication between research institutions
 - ARPANET (TCP/IP)
 - First packet sent from UCLA to SRI
- Internet designed without security in mind
 - Including key protocols such as TCP/IP
 - Getting it to work is already an amazing job
- Hard to retrofit security into existing protocols
 - Have to remain backward-compatible
 - E.g., TCP/IP used by every machine now
 - Solutions often are patches or require an additional layer of indirection

How the Internet looks like in 2005



Quick overview of TCP/IP

- Example:



- Network traffic is broken down into “packets” containing information at 4 main layers

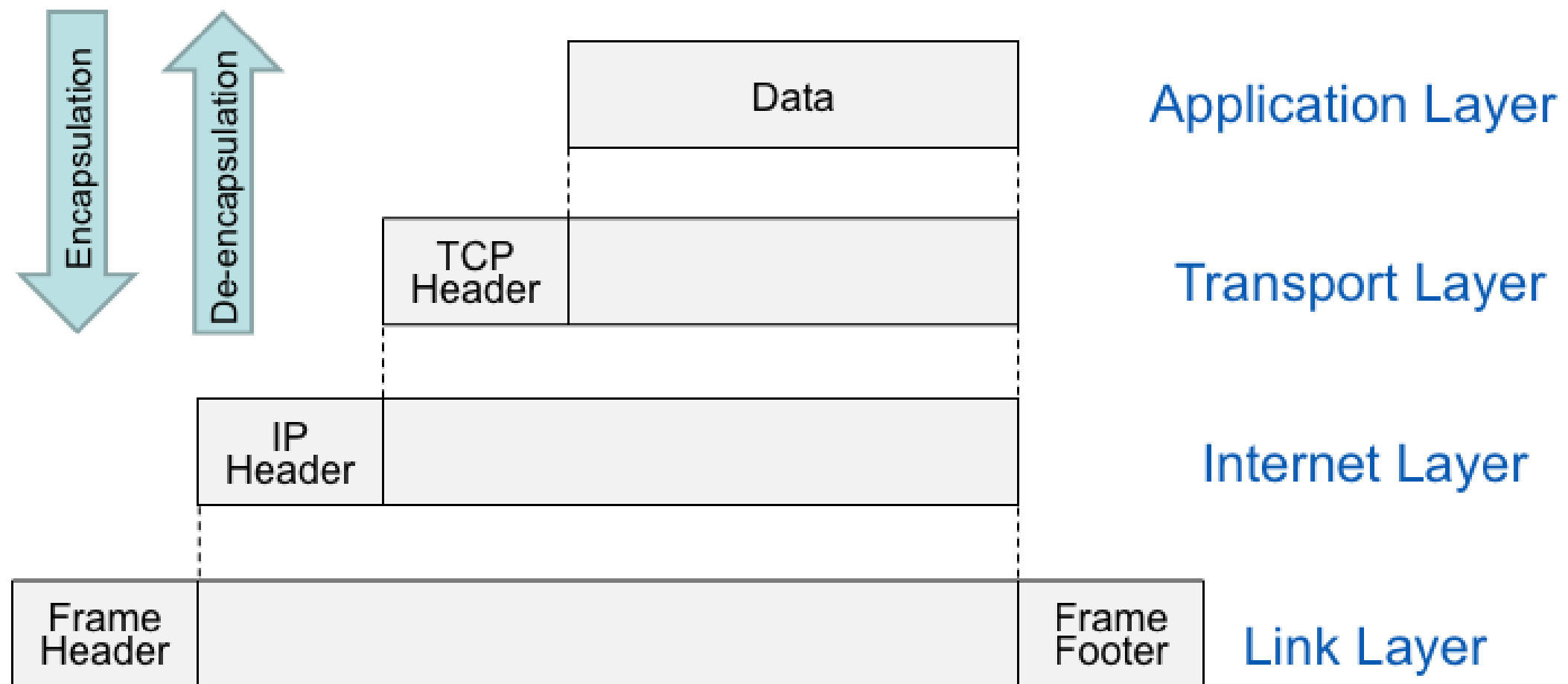
Layer 4 : Application

Layer 3 : Transport (TCP)

Layer 2 : Network (IP)

Layer 1 : Link/Physical

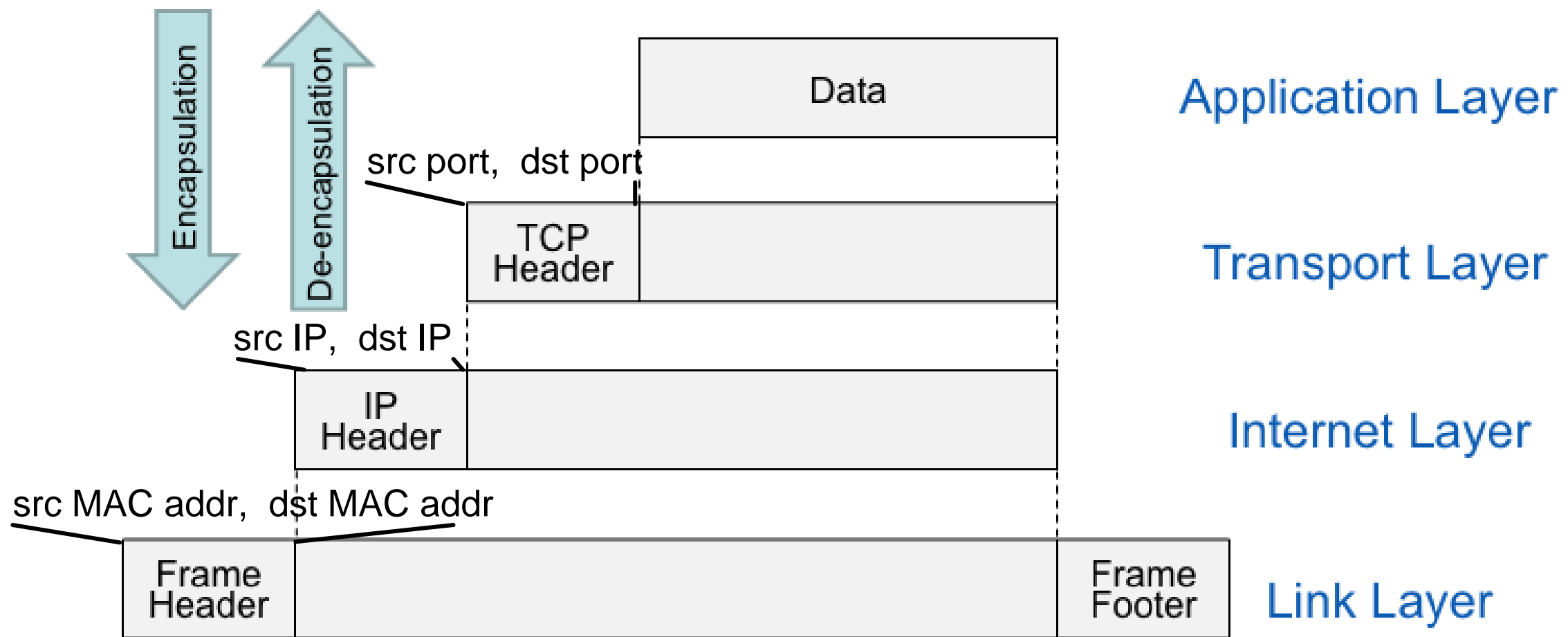
TCP/IP network Layers



Headers at higher layers become data at lower layers

Source: IETF RFC 1122

TCP/IP network Layers

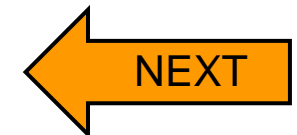


Headers at higher layers become data at lower layers

Source: IETF RFC 1122

Common network security attacks and their countermeasures

- Packet sniffing and spoofing
 - Encryption (SSH, SSL, HTTPS)
- Finding a way into the network
 - Firewalls
- Exploiting software bugs, buffer overflows
 - Intrusion Detection Systems
- Denial of Service
 - Ingress filtering, IDS



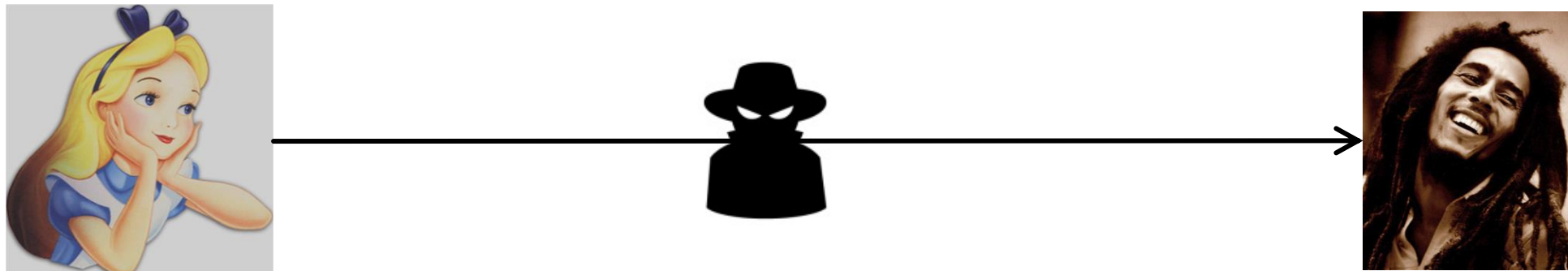
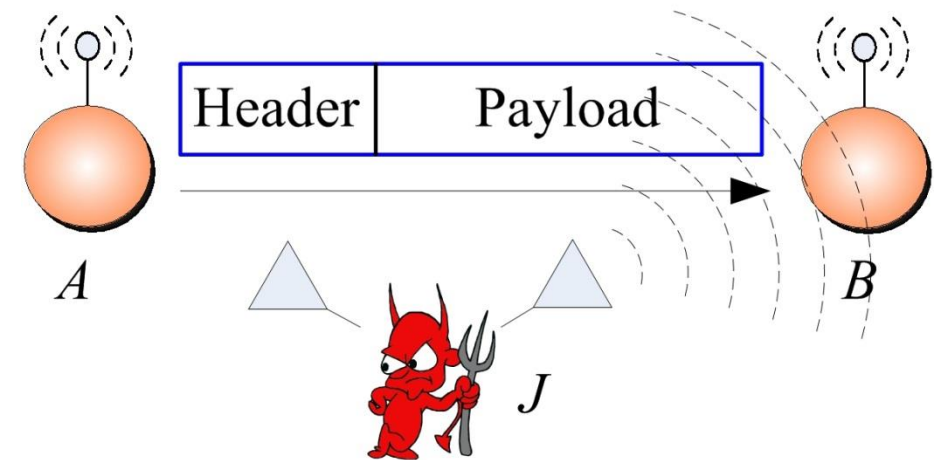
Common Threat Models in Networks

- **Passive Eavesdropper**

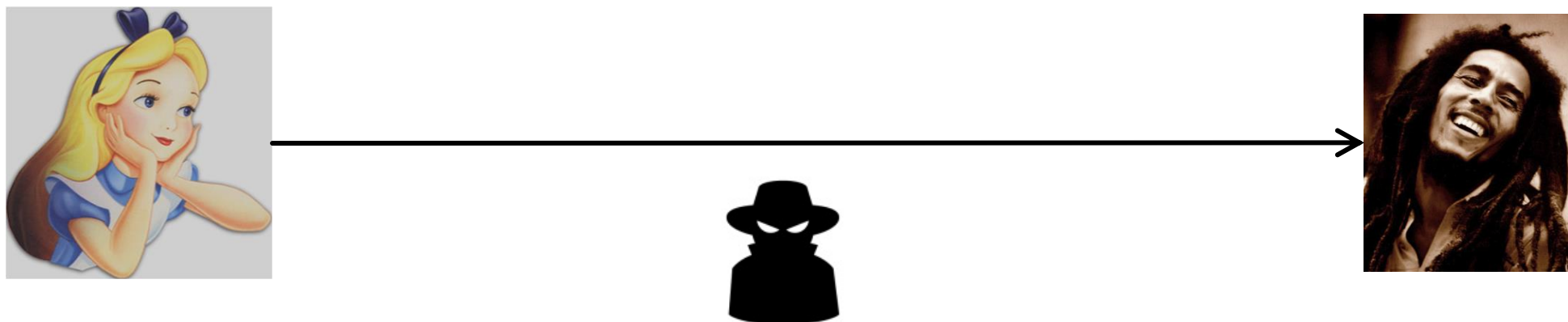
- Read (and Insert)

- **Man-in-the-middle (MITM)**

- On the communication path (compromised router)
- Arbitrary Read/Write capability (modify, drop, etc.)

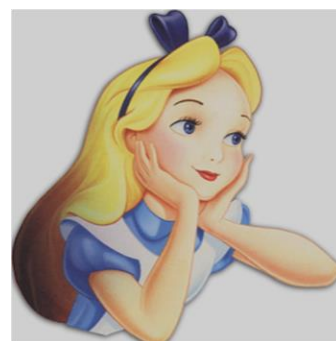
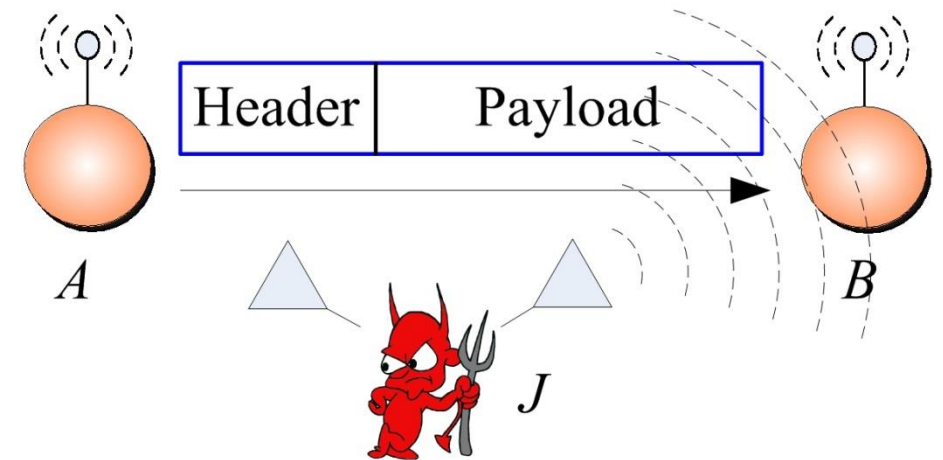


- **Off-Path attacker (no read capability)**



Passive Eavesdropper

- Read (and Insert)

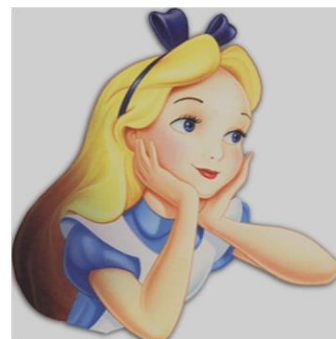


Let's hang out tomorrow night?



Man-in-the-middle

- Read/Write (drop, modify, inject)



Let's hang out tomorrow night?

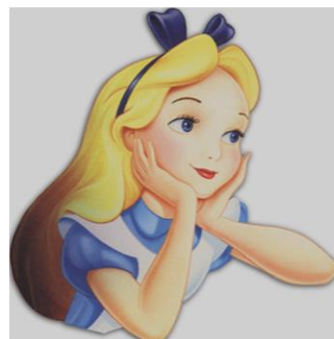


Bye



Off-Path

- Inject only



Let's hang out tomorrow night?



I hate you



IP Layer

- Responsible for end to end transmission
- Sends data in individual packets
- Maximum size of packet is determined by the networks
 - Fragmented if too large
- Unreliable
 - Packets might be lost, corrupted, duplicated, delivered out of order

IP addresses

- 4 bytes (IPv4)
 - e.g. 163.1.125.98
 - Each device normally gets one (or more)
 - In theory there are about 4 billion available
- But...
 - Basically used up today
 - Therefore, 16 bytes are now used in IPv6 (still not fully deployed today)

Routing

- How does a device know where to send a packet?
 - All devices need to know what IP addresses are on directly attached networks
 - If the destination is on a local network, send it directly there

Routing (cont)

- If the destination address isn't local
 - Most non-router devices just send everything to a single local router (gateway)
 - Routers need to know which network corresponds to each possible IP address

Allocation of addresses

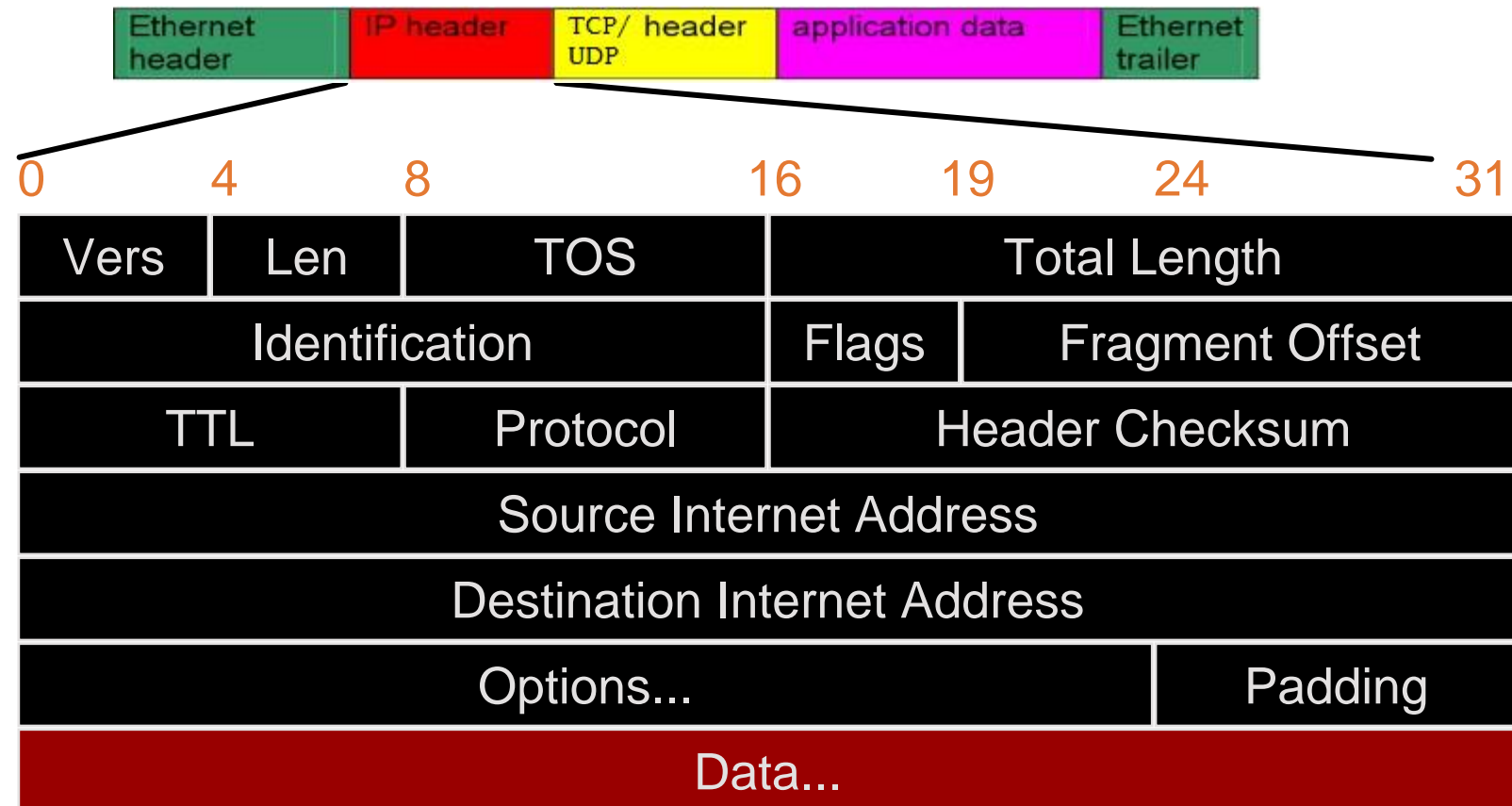
- Controlled centrally by ICANN
 - Fairly strict rules on further delegation to avoid wastage
 - Have to demonstrate actual need
- Organizations that got in early have bigger allocations than they really need

MIT	18.0.0.0/9	= 8,388,608 IPs
UCR	169.235.0.0/16	= 65536
UCI	169.234.0.0/16	
UCSC	169.233.0.0/16	
UCLA	169.232.0.0/16	

IP packets

- Source and destination addresses
- Protocol number
 - 1 = ICMP, 6 = TCP, 17 = UDP
- Various options
 - e.g. to control fragmentation
- Time to live (TTL)
 - Prevent routing loops

IP Datagram



Field Purpose

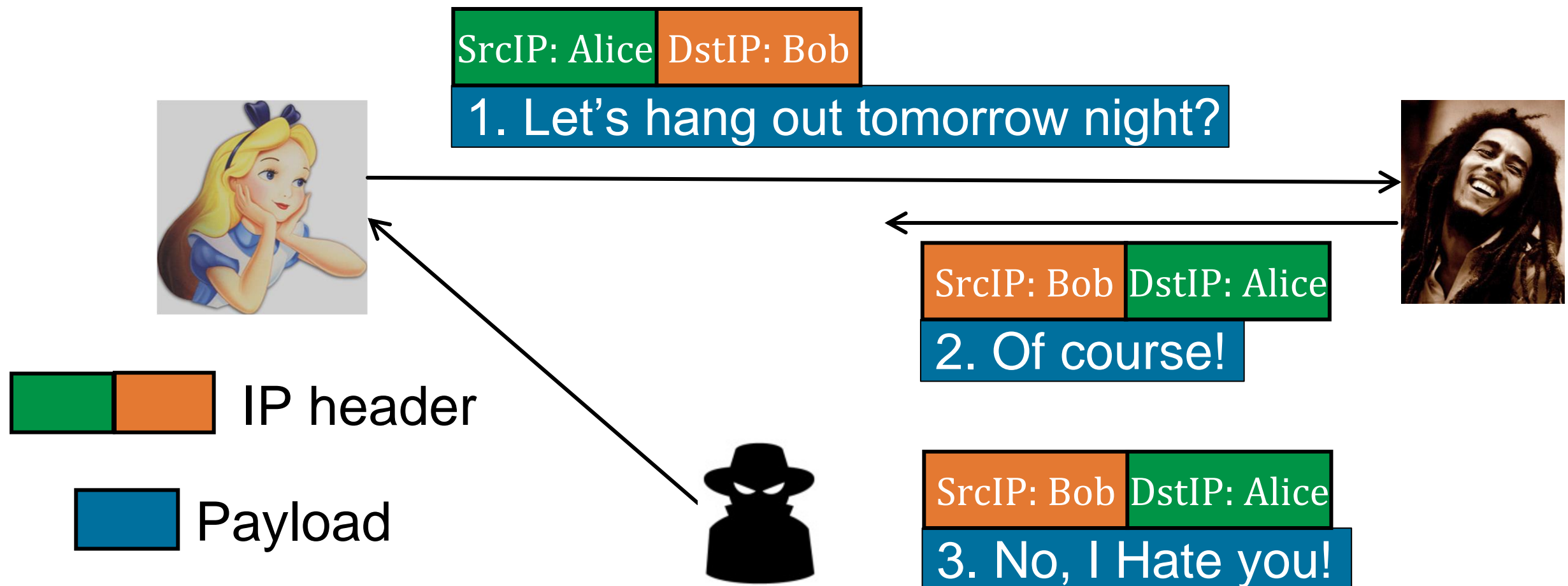
Vers IP version number
 Len Length of IP header (4 octet units)
 TOS Type of Service
 T. Length Length of entire datagram (octets)
 Ident. IP datagram ID (for frag/reassembly)
 Flags Don't/More fragments
 Frag Off Fragment Offset

Field Purpose

TTL Time To Live - Max # of hops
 Protocol Higher level protocol (1=ICMP, 6=TCP, 17=UDP)
 Checksum Checksum for the IP header
 Source IA Originator's Internet Address
 Dest. IA Final Destination Internet Address
 Options Source route, time stamp, etc.
 Data... Higher level protocol data

Problem with IP address (off-path attack)

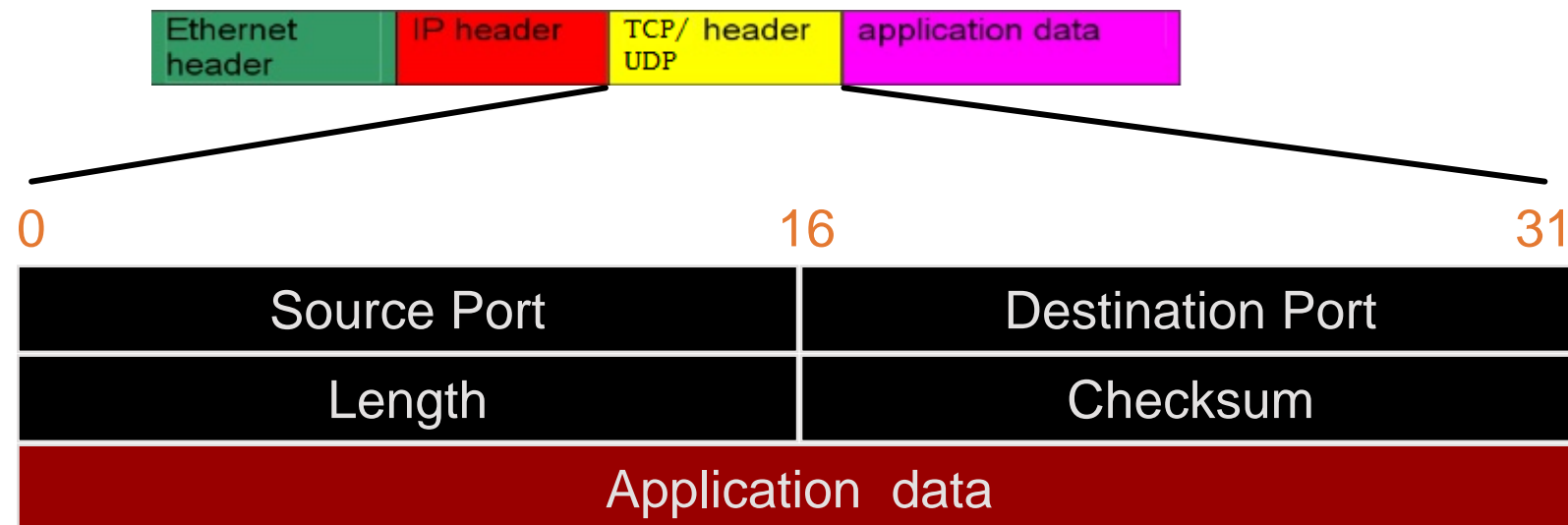
- Source address in a packet can be filled arbitrarily by a host (think of USPS mail)
 - Lack of authentication of packet sources
 - Many vulnerabilities arise because of this



UDP

- Thin layer on top of IP (alternative to TCP)
- Adds packet length + checksum
 - Guard against corrupted packets
- Also **source and destination *ports***
 - Ports are used to associate a packet with a specific application at each end
- Still unreliable:
 - Duplication, loss, out-of-orderness possible

UDP datagram



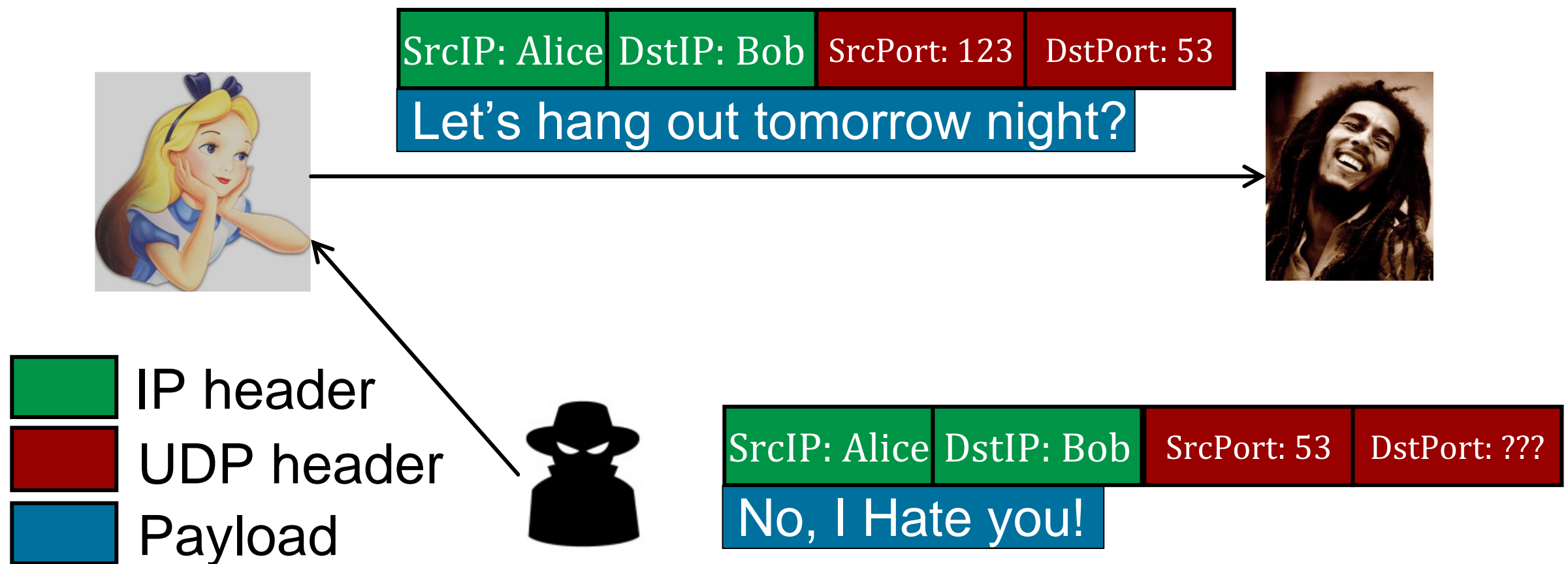
Field	Purpose
Source Port	16-bit port number identifying originating application
Destination Port	16-bit port number identifying destination application
Length	Length of UDP datagram (UDP header + data)
Checksum	Checksum of IP pseudo header, UDP header, and data

Typical applications of UDP

- Where packet loss etc is better handled by the application than the network stack
- Where the overhead of setting up a TCP connection isn't wanted
- DNS (no encryption by default)
- VoIP
- Some games

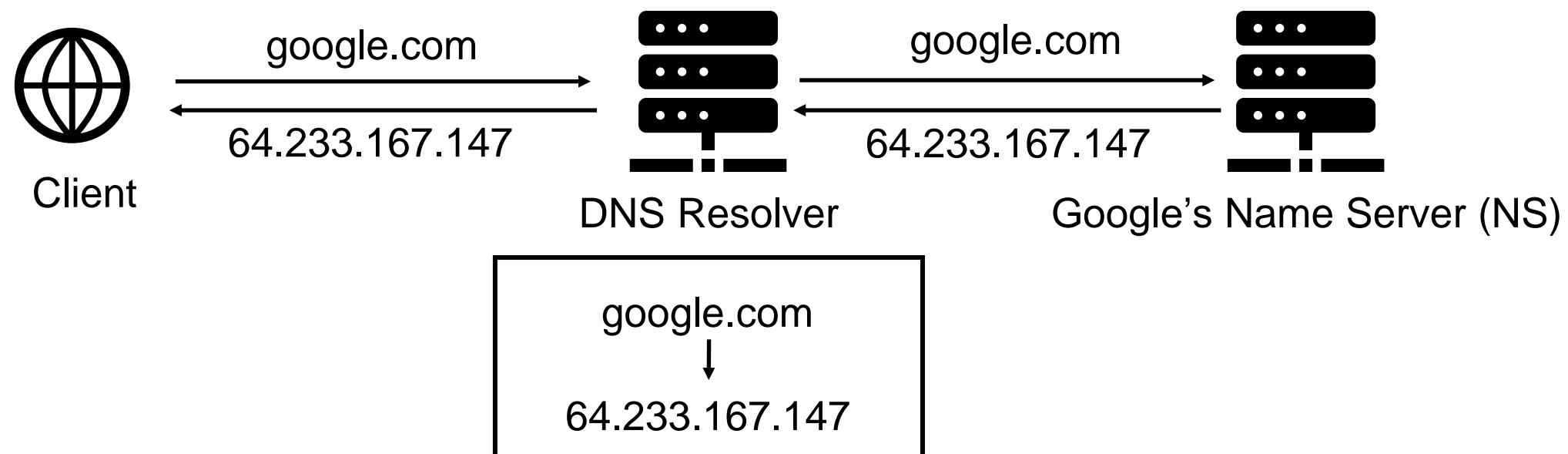
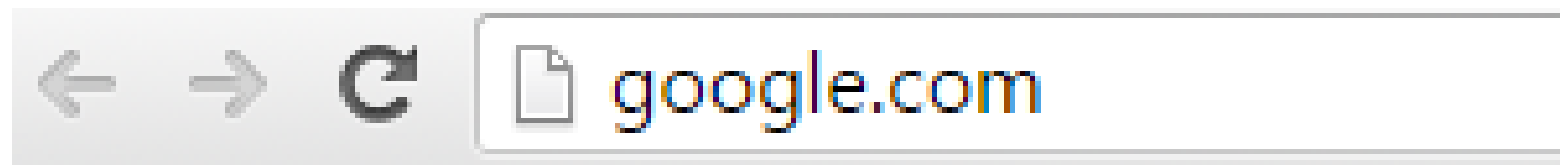
Off-path attack against UDP

- Need to guess the client's port number...



DNS Protocol – Application Layer Protocol

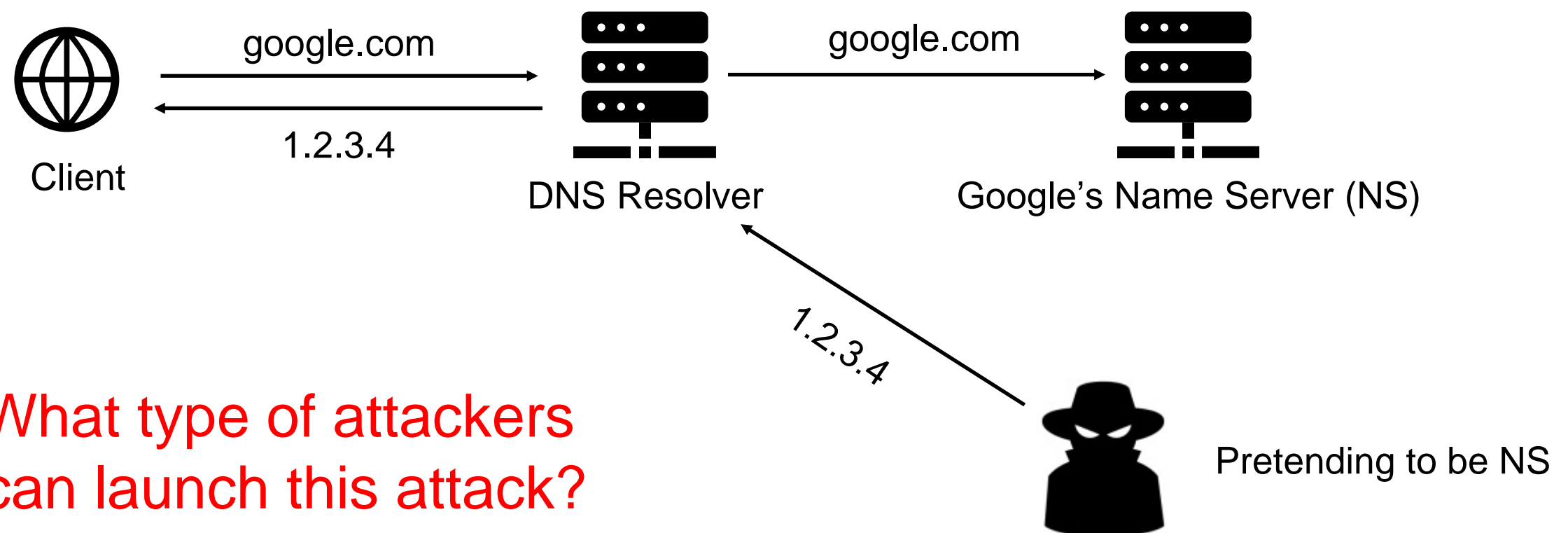
- Domain name -> IP addresses



DNS
UDP
IP
Ethernet

DNS Cache Poisoning Attack

- Attacker can spoof its source IP as name server's IP



What type of attackers
can launch this attack?

Challenge: Need to guess the port number

Our research in 2020 and 2021: infer the port number from off-path

TCP

- Reliable, *full-duplex*, *connection-oriented*, *stream* delivery
 - Interface presented to the application doesn't require data in individual packets
 - Data is guaranteed to arrive, and in the correct order without duplicates
 - Or the connection will be dropped
 - Imposes significant overheads

Applications of TCP

- Most things!
 - HTTP, FTP, SMTP...
- Saves the application a lot of work, so used unless there's a good reason not to
 - QUIC and HTTP 3.0 build on UDP for max performance

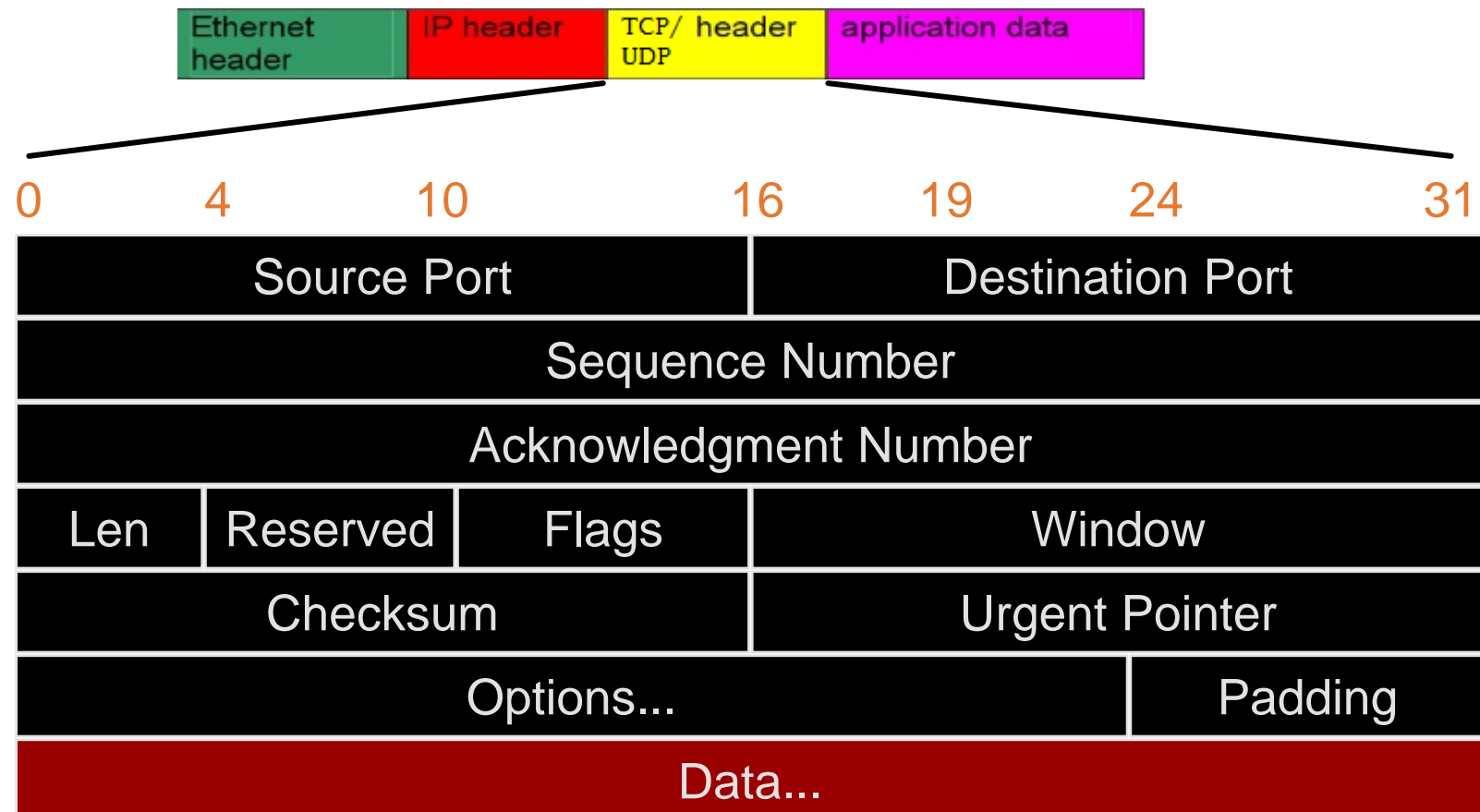
TCP implementation

- Connections are established using a *three-way handshake*
- Data is divided up into packets by the operating system
- Packets are numbered, and received packets are acknowledged
- Connections are explicitly closed
 - (or may abnormally terminate)

TCP Packets

- Source + destination ports
- Sequence number
- Acknowledgement number
- Checksum
- Various options

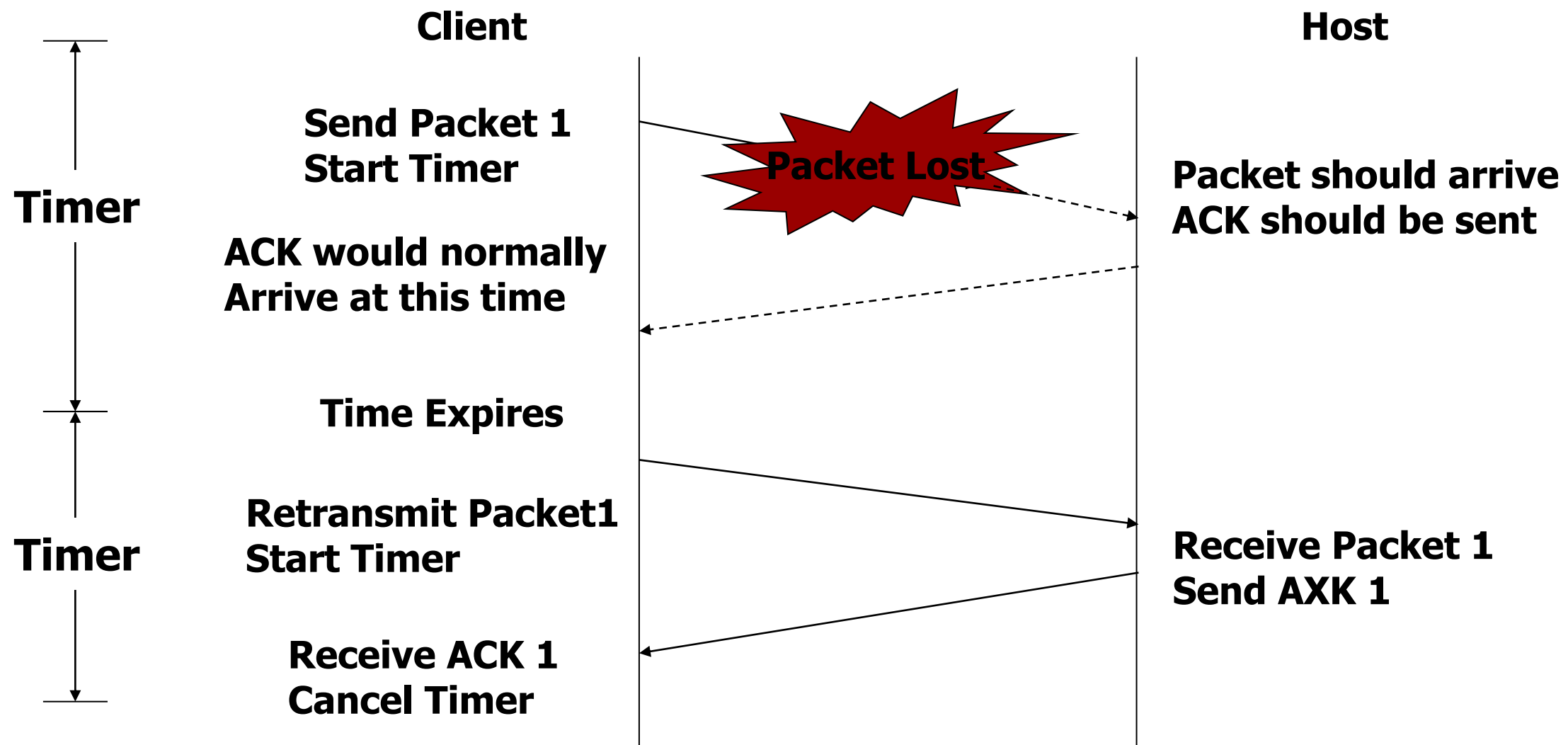
TCP Segment



<u>Field</u>	<u>Purpose</u>
Source Port	Identifies originating application
Destination Port	Identifies destination application
Sequence Number	Sequence number of first octet in the segment
Acknowledgment #	Sequence number of the next expected octet (if ACK flag set)
Len	Length of TCP header in 4 octet units
Flags	TCP flags: SYN, FIN, RST, PSH, ACK, URG
Window	Number of octets from ACK that sender will accept
Checksum	Checksum of IP pseudo-header + TCP header + data
Urgent Pointer	Pointer to end of "urgent data"
Options	Special TCP options such as MSS and Window Scale

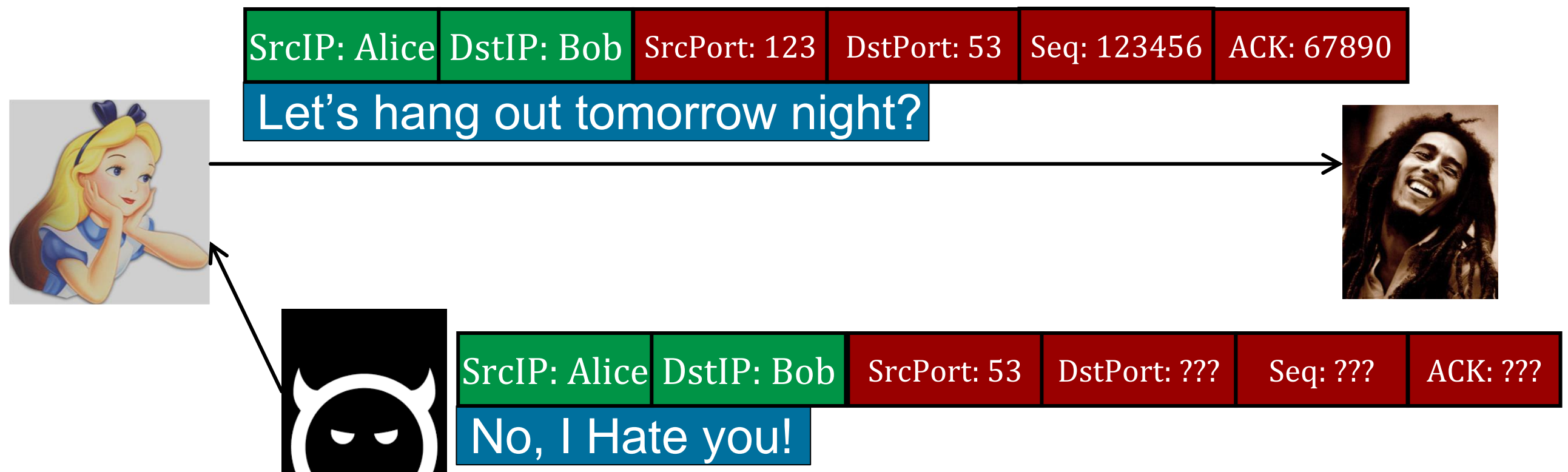
You just need to know port numbers, seq and ack are added




TCP : Data transfer



Off-path attack against TCP

- Need to guess the **port number, sequence number, and acknowledgement number!**






 IP header
 TCP header
 Payload

What about eavesdropping and MITM?

Defenses - encryption

- Without the secret key, an attacker can't read, write, or inject data



 IP header
 TCP header
 Payload

Current state of affairs

- **Traffic almost never encrypted**
 - IP, TCP/UDP, DNS, telnet
- **Traffic encrypted often**
 - Web traffic (HTTPS, QUIC) > 50% encrypted
 - Mileage varies
- **Traffic always encrypted**
 - SSH, Email traffic (SMTP)
- **Encryption sometimes can be broken too!**

Common network security attacks and their countermeasures

- Packet sniffing and spoofing
 - Encryption (SSH, SSL, HTTPS)
- Finding a way into the network
 - Firewalls
- Exploiting software bugs, buffer overflows
 - Intrusion Detection Systems
- Denial of Service
 - Ingress filtering, IDS



Finding a way into the network -- Scanning

Host 192.168.2.1 appears to be up.

MAC Address: 00:04:E2:34:B6:CE (SMC Networks)

Host 192.168.2.79 appears to be up.

MAC Address: 00:11:11:5B:7A:CD (Intel)

Host 192.168.2.82 appears to be up.

MAC Address: 00:10:5A:0D:F6:D7 (3com)

Host 192.168.2.198 appears to be up.

MAC Address: 00:10:DC:55:89:27 (Micro-star International)

Host 192.168.2.199 appears to be up.

MAC Address: 00:C0:4F:36:33:91 (Dell Computer)

Host 192.168.2.200 appears to be up.

MAC Address: 00:0C:41:22:CC:01 (The Linksys Group)

Host 192.168.2.251 appears to be up.

MAC Address: 00:0F:66:75:3D:75 (Cisco-Linksys)

Does That Matter?

- The number of computers an organization has roughly corresponds to the number of people in it
- How large is your competitor?
- (How many computers does Google have in its data centers? They won't say.)

Does That Matter?

- If they identify a service that has a known vulnerability (e.g., buffer overflow), they can launch the corresponding exploit

```
$ nmap -Pn www.cs.ucr.edu
```

```
Starting Nmap 6.40 ( http://nmap.org ) at 2015-11-17 20:03 UTC
```

```
Nmap scan report for www.cs.ucr.edu  
(169.235.30.15)
```

```
Host is up (0.00033s latency).
```

```
rDNS record for 169.235.30.15: thoth.cs.ucr.edu
```

```
Not shown: 996 closed ports
```

```
PORT      STATE SERVICE
```

```
22/tcp    open  ssh
```

```
80/tcp    open  http
```

```
111/tcp   open  rpcbind
```

```
5666/tcp  open  nrpe
```

Firewalls



- Basic problem – many network applications and protocols have security problems that are fixed over time
 - Difficult for users to keep up with changes and keep host secure
 - Solution
 - Administrators limit access to end hosts by using a firewall
 - Firewall is kept up-to-date by administrators

Firewalls



- A firewall is like a castle with a drawbridge
 - Only one point of access into the network
 - This can be good or bad
- Can be hardware or software
 - Ex. Some routers come with firewall functionality
 - ipfw, ipchains, pf on Unix systems, Windows XP and Mac OS X have built in firewalls

Firewalls



- Used to filter packets based on a combination of features
 - These are called packet filtering firewalls
 - There are other types too, but they will not be discussed
 - Ex. Drop packets with destination port of 23 (Telnet)
 - Can use any combination of IP/UDP/TCP header information
- But why don't we just turn Telnet off?

Firewalls

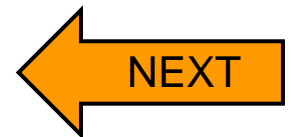


- Here is what a computer with a default Windows install looks like:

- 135/tcp open loc-srv
- 139/tcp open netbios-ssn
- 445/tcp open microsoft-ds
- 1025/tcp open NFS-or-IIS
- 3389/tcp open ms-term-serv
- 5000/tcp open UPnP

Common network security attacks and their countermeasures

- Packet sniffing and spoofing
 - Encryption (SSH, SSL, HTTPS)
- Finding a way into the network
 - Firewalls
- Exploiting software bugs, buffer overflows
 - Intrusion Detection Systems
- Denial of Service
 - Ingress filtering, IDS



Intrusion Detection



- Used to monitor for “suspicious activity” on a network
 - Can protect against known software exploits, like buffer overflows
- Open Source IDS:
 - Snort, www.snort.org
 - Bro
- Monitor payload of packets
 - Modern firewalls also do that (blurred definition)

Intrusion Detection



- Uses “intrusion signatures” vs. “anomaly detection”
 - Well known patterns of behavior
 - Ping sweeps, port scanning, web server indexing, OS fingerprinting, DoS attempts, etc.
- Example
 - IRIX vulnerability in `webdist.cgi`
 - Can make a rule to drop packets containing the line
 - `"/cgi-bin/webdist.cgi?distloc=?;cat%20/etc/passwd"`
- However, IDS is only useful against certain forms of attacks
 - What if traffic is encrypted?
 - What if an attacker can morph into a different payload?

Common network security attacks and their countermeasures

- Packet sniffing and spoofing
 - Encryption (SSH, SSL, HTTPS)
- Finding a way into the network
 - Firewalls
- Exploiting software bugs, buffer overflows
 - Intrusion Detection Systems
- Denial of Service
 - Ingress filtering, IDS

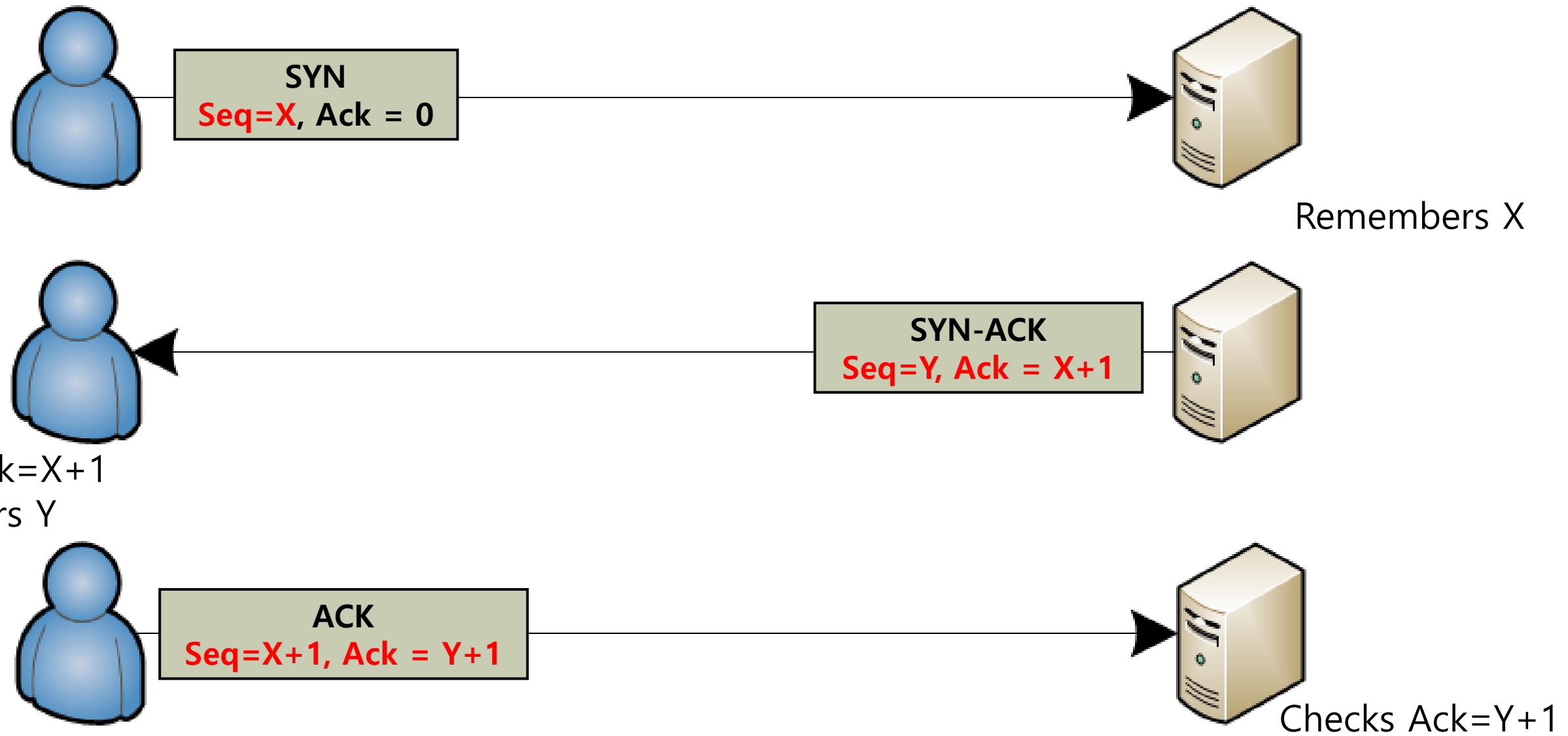


Denial of Service

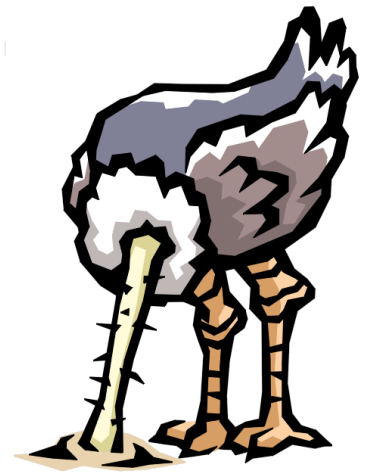


- Purpose: Make a network service unusable, usually by overloading the **server or network**
- General strategies of DoS attacks
 - Attacker: small resource -> amplifying impact (asymmetric)
 - Attacker: large resource -> bruteforce

TCP Three-way handshake



Denial of Service

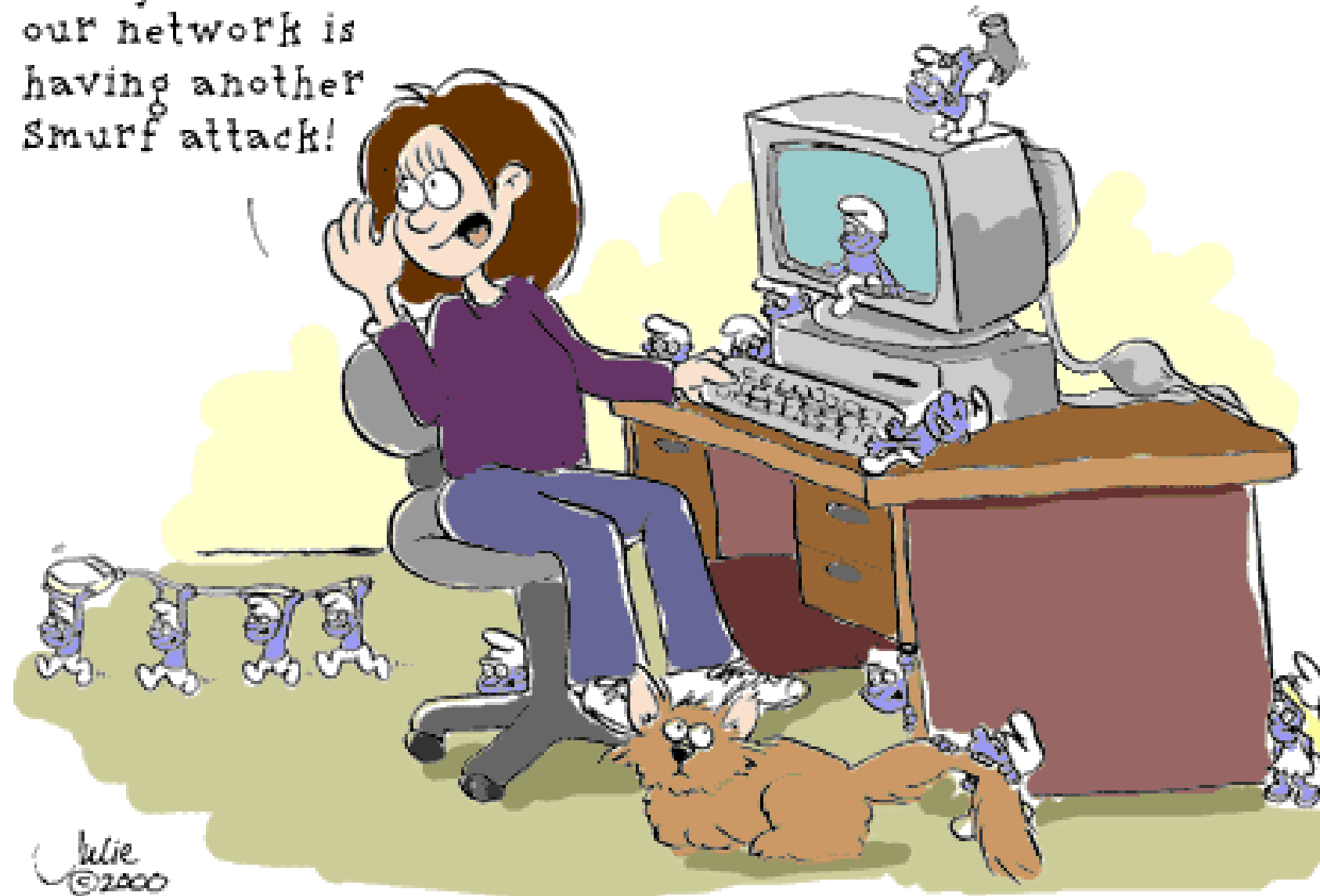


- SYN flooding attack
 - Send SYN packets with spoofed/bogus source address (IP spoofing)
 - Why?
 - Server responds with SYN ACK and keeps state about TCP half-open connection
 - Eventually, server memory is exhausted with this state

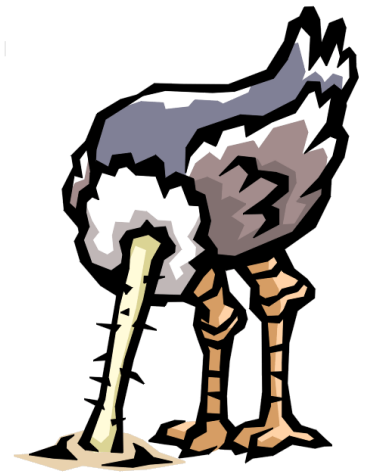
Denial of Service



Honey! I think
our network is
having another
Smurf attack!

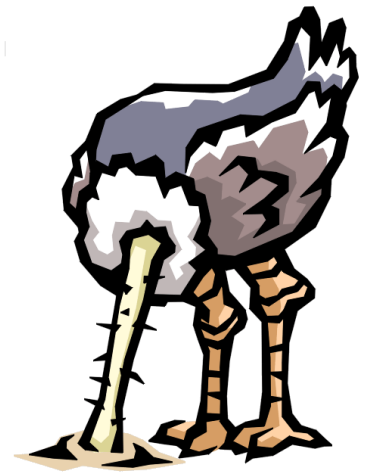


Denial of Service

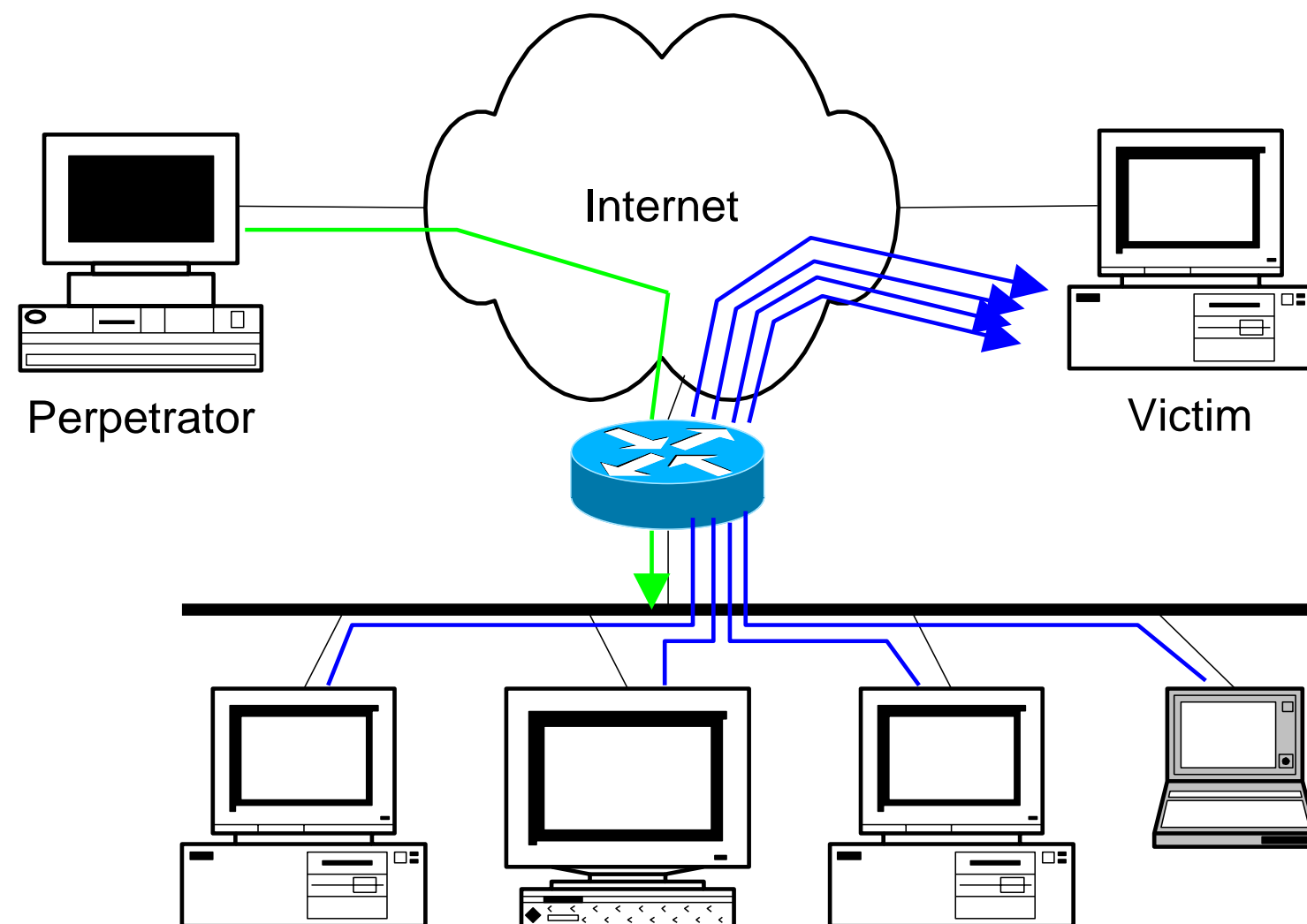


- SMURF
 - Source IP address of a broadcast ping is forged
 - Large number of machines respond back to victim, overloading it

Denial of Service

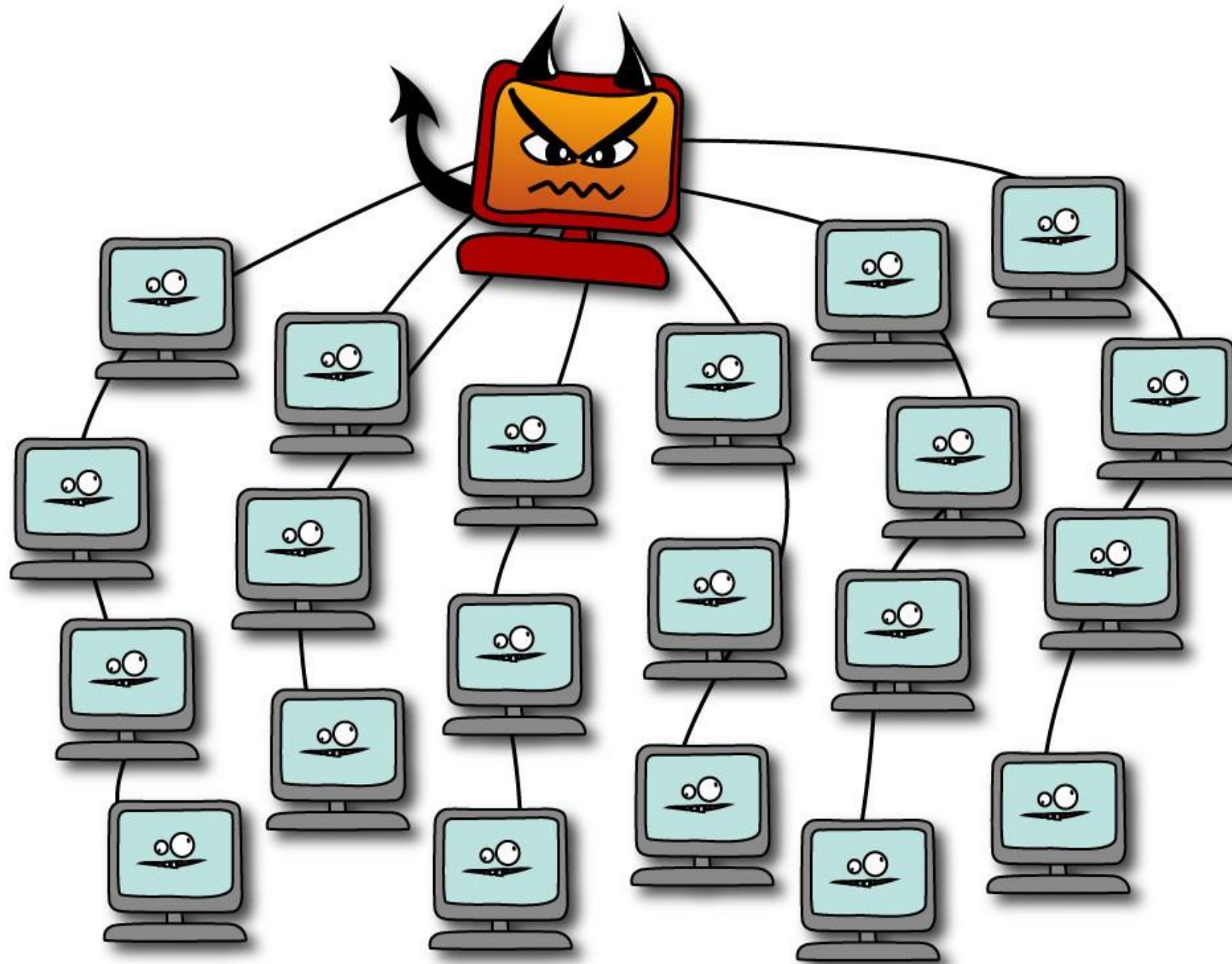


- ICMP echo (spoofed source address of victim)
Sent to IP broadcast address
- ICMP echo reply



Denial of Service

- Distributed attacks, e.g., botnet



Questions

