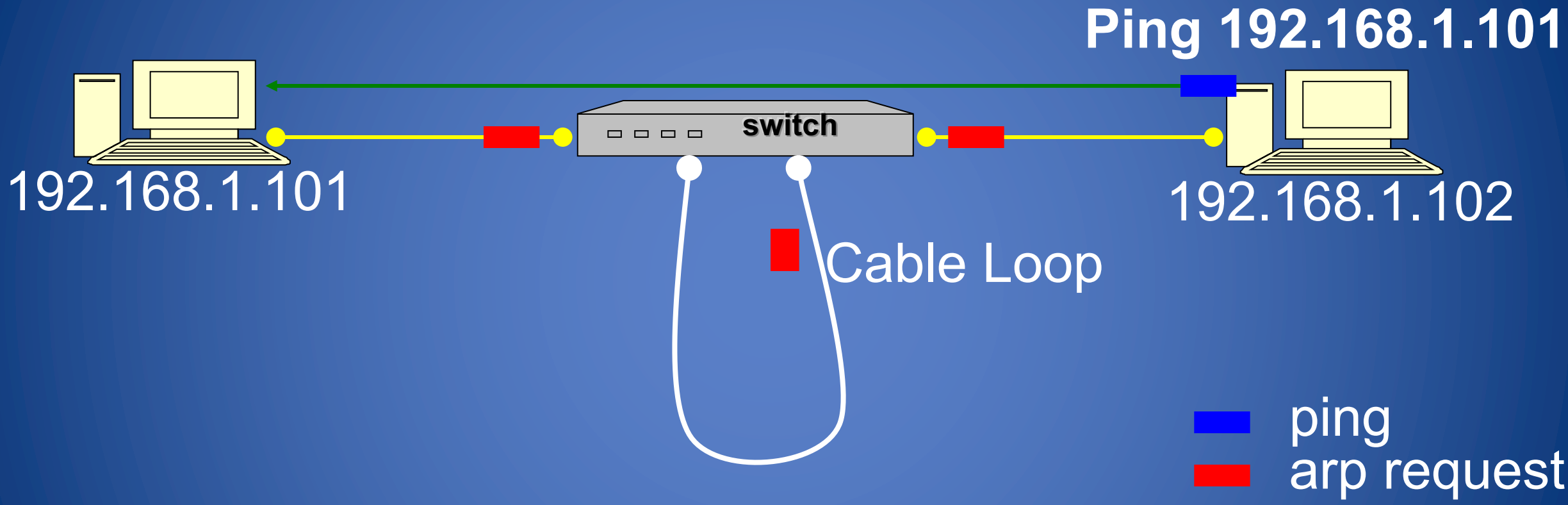
# Networks III
# DoS, Routing, Transport Layer

## CS 1660: Introduction to Computer Systems Security

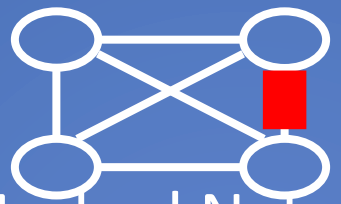
# Link Layer DoS

# network DoS using ARP

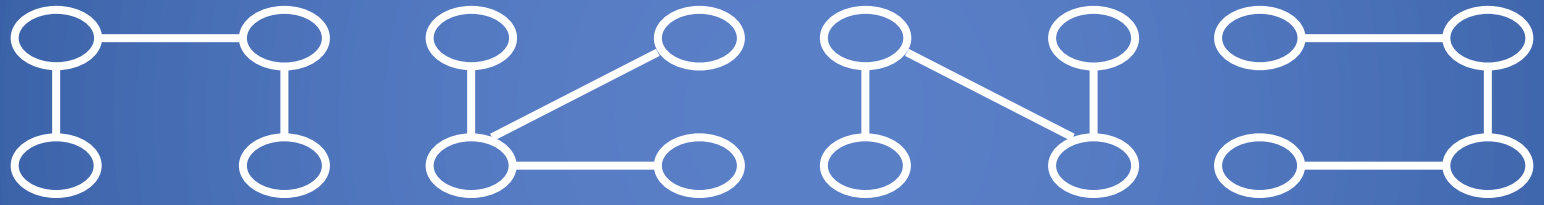



How can it be solved?

# Spanning Tree Protocol (ISO 802.1D)



A Meshed Network



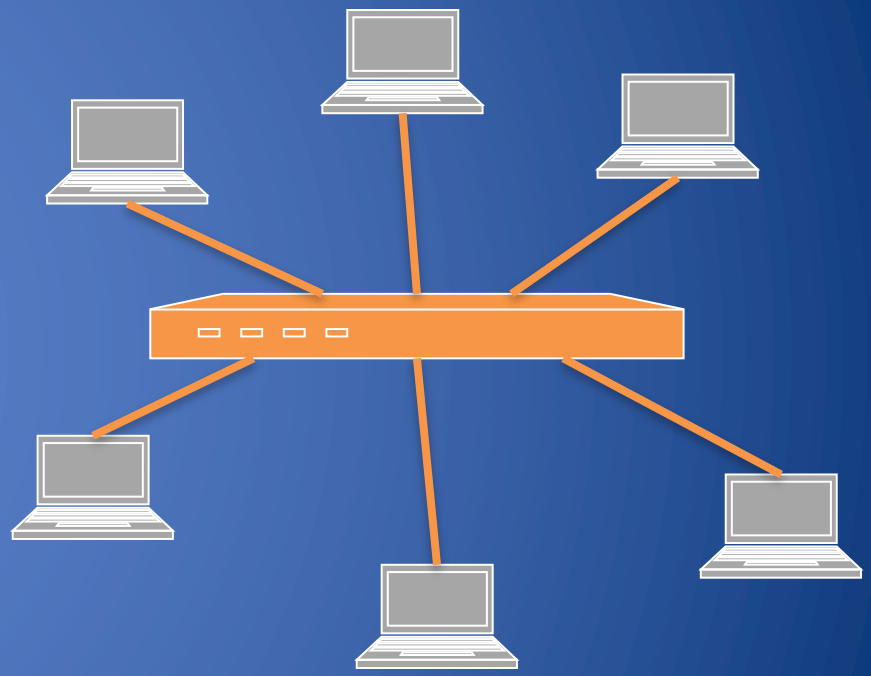
Four spanning trees of the Meshed Network

- Suppose you have a Meshed Network with bidirectional links that make loops/cycles…
- …then a spanning tree of the Meshed Network is the same network and no loops/cycles

# Another attack on switches

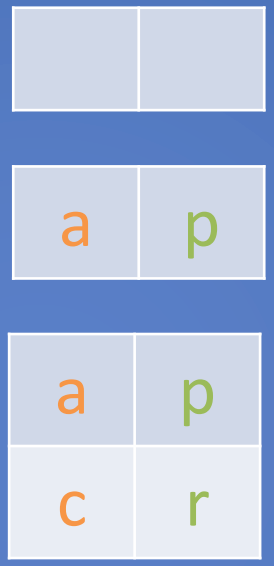
# Switching

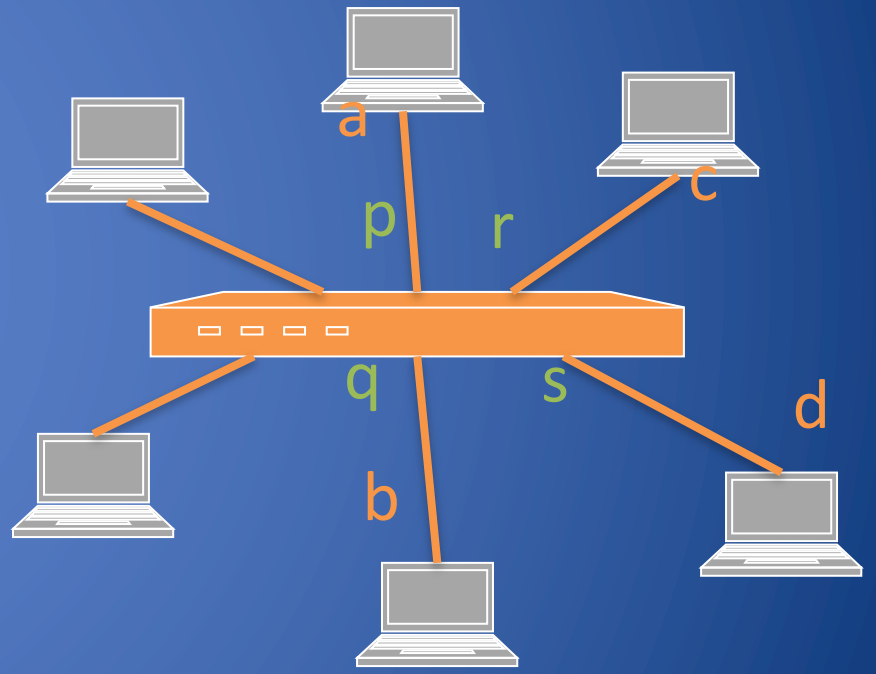
- A switch connects devices on a local area network (LAN)
- Has multiple interfaces, or ports
- Operates on link-layer frames
- As devices connect, learns MAC addresses of some or all the devices on the network



# Extra: Example

- Table initially empty
- Frame (a, b) broadcast;
  - entry (a, p) added to table
- Frame (c, a) forwarded on p
  - entry (c, r) added to table
- Frame (a, c) forwarded on r
  - table unchanged
- Frame (a, d) broadcast
  - table unchanged



# Frame Processing

- Switch receives on port p frame with source s and destination d

  e = get(d)

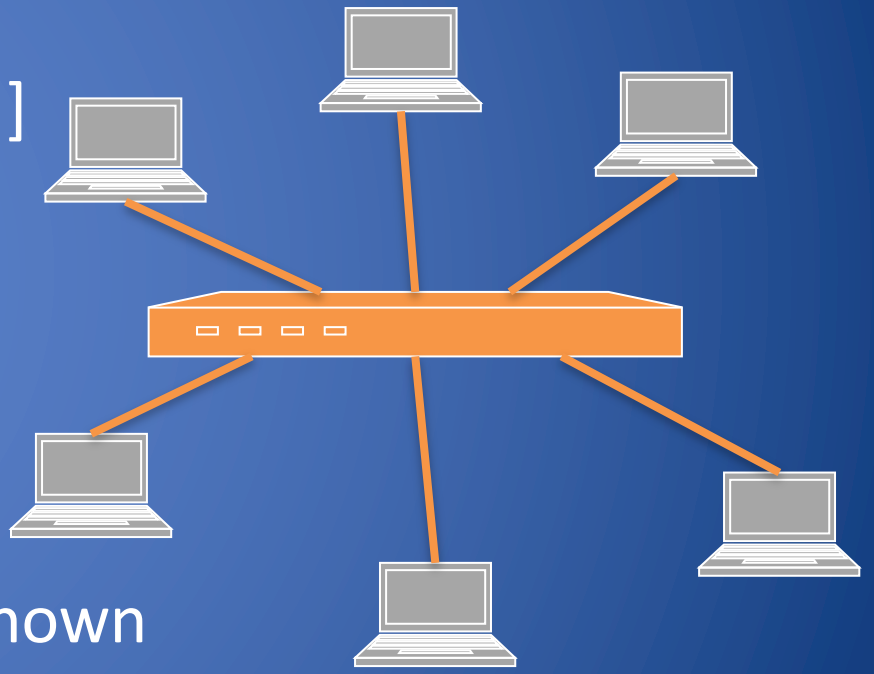
  **if** e == null [ device with address d not known ]

    broadcast frame on all ports but p

  **else** [ device with address d known ]

    forward frame on e.port

  put(s, p) [ adds or updates table entry ]



For a network with a single switch, a frame with known destination address is directly delivered only to the recipient

# Attack on a learning switch

- Idea:  flood the switch with many packets from different source MAC addresses

- If MAC table is full, switch uses to broadcast mode for other packets not in table

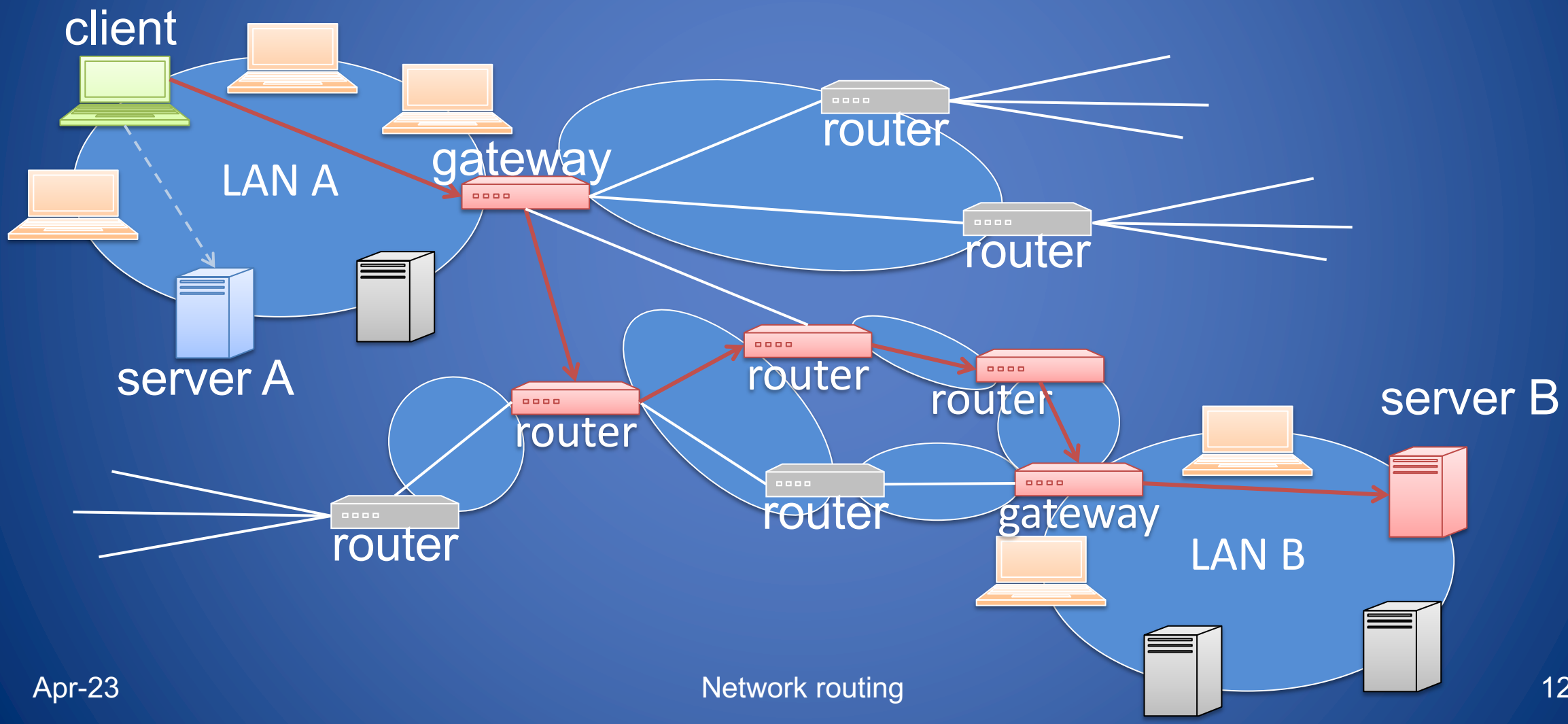
# Routing
# How does internet actually work?

# A big incident

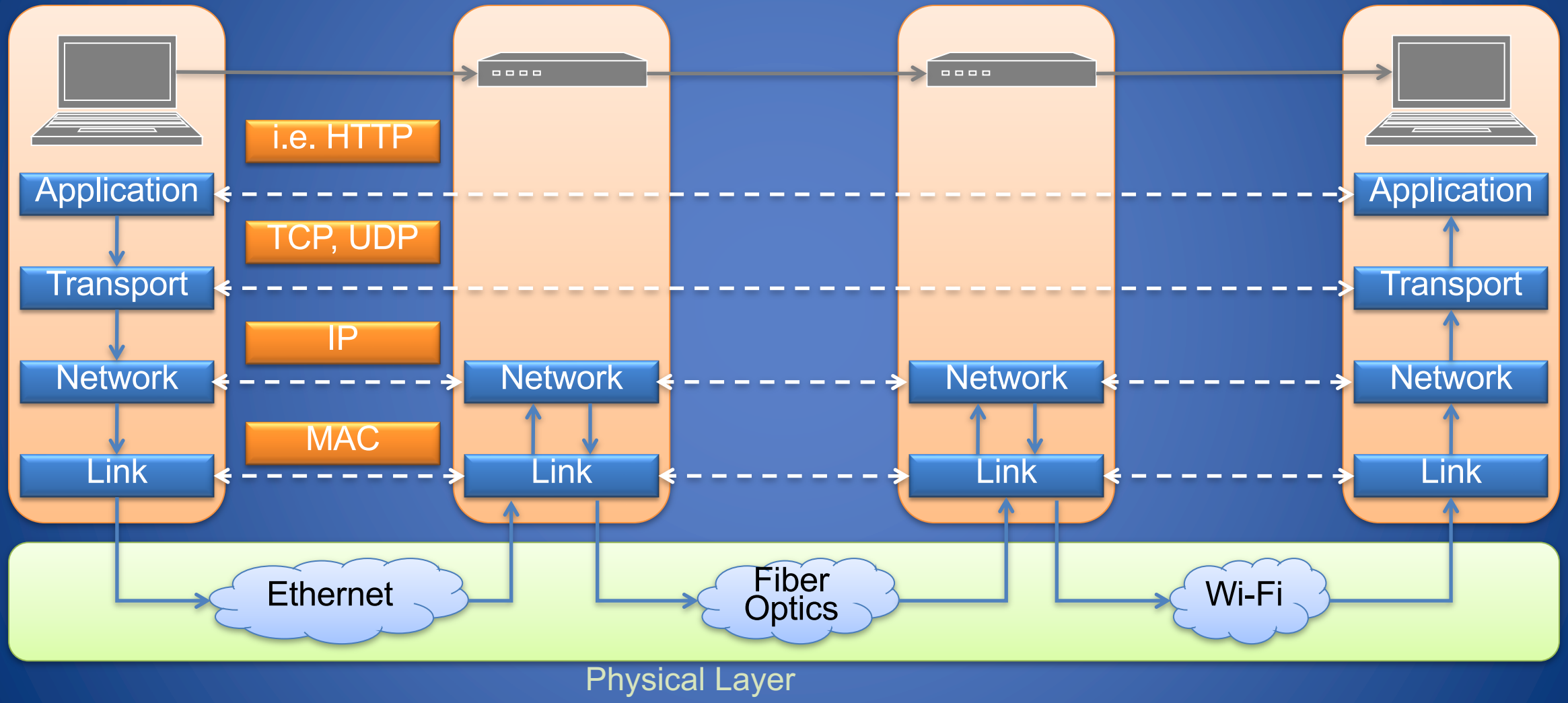
- February 2008 Pakistan Telecom (PT) would like to block Youtube access from Pakistan
  - PT falsely informed that through this company there was the most directed way to reach Youtube
- Soon over 2/3 of the Internet was not able to reach Youtube for a couple of hours
- A Routing problem…

# Why Routing?

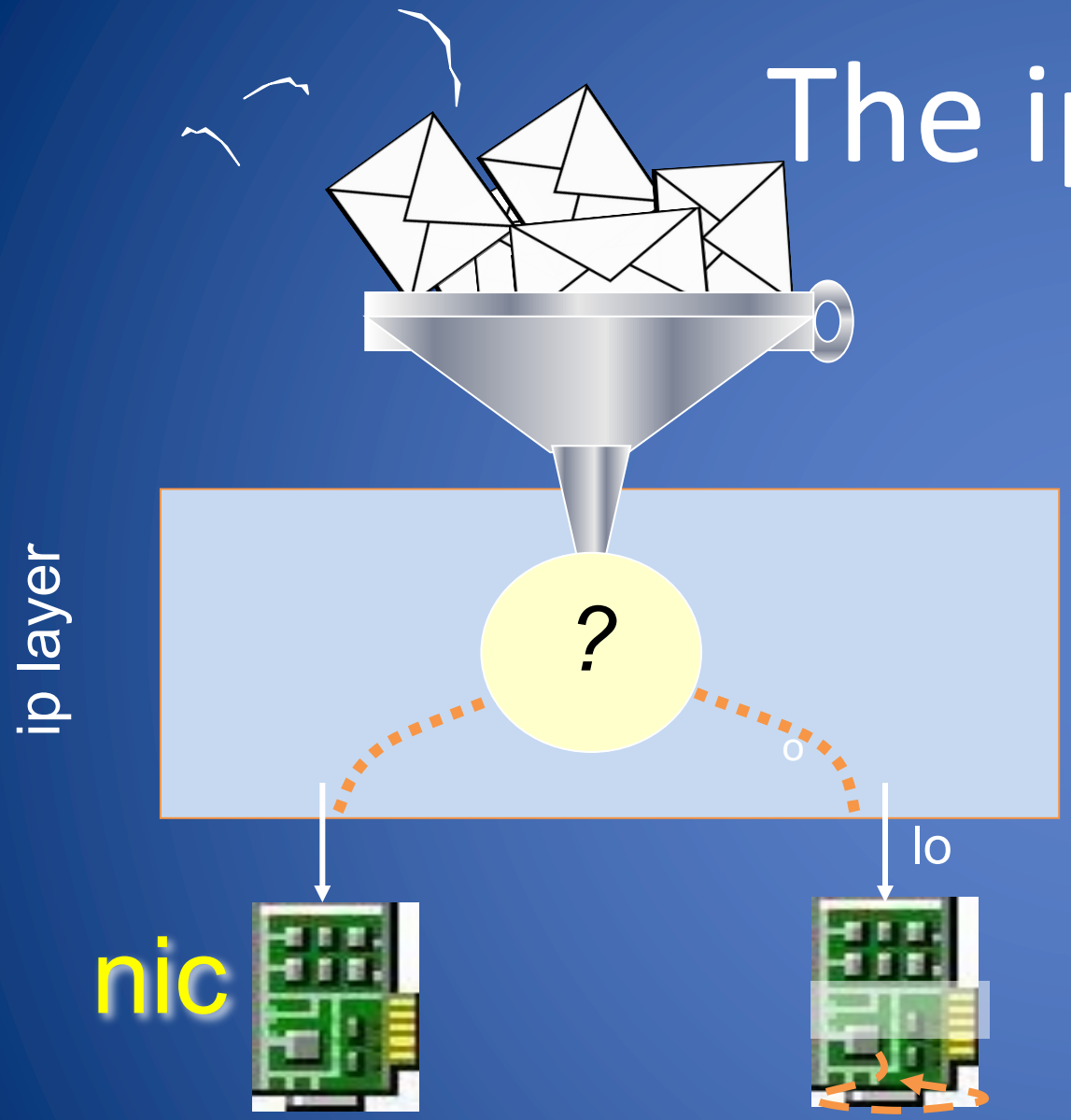
- Reaching a host within a network is a routing problem



# Internet Layers

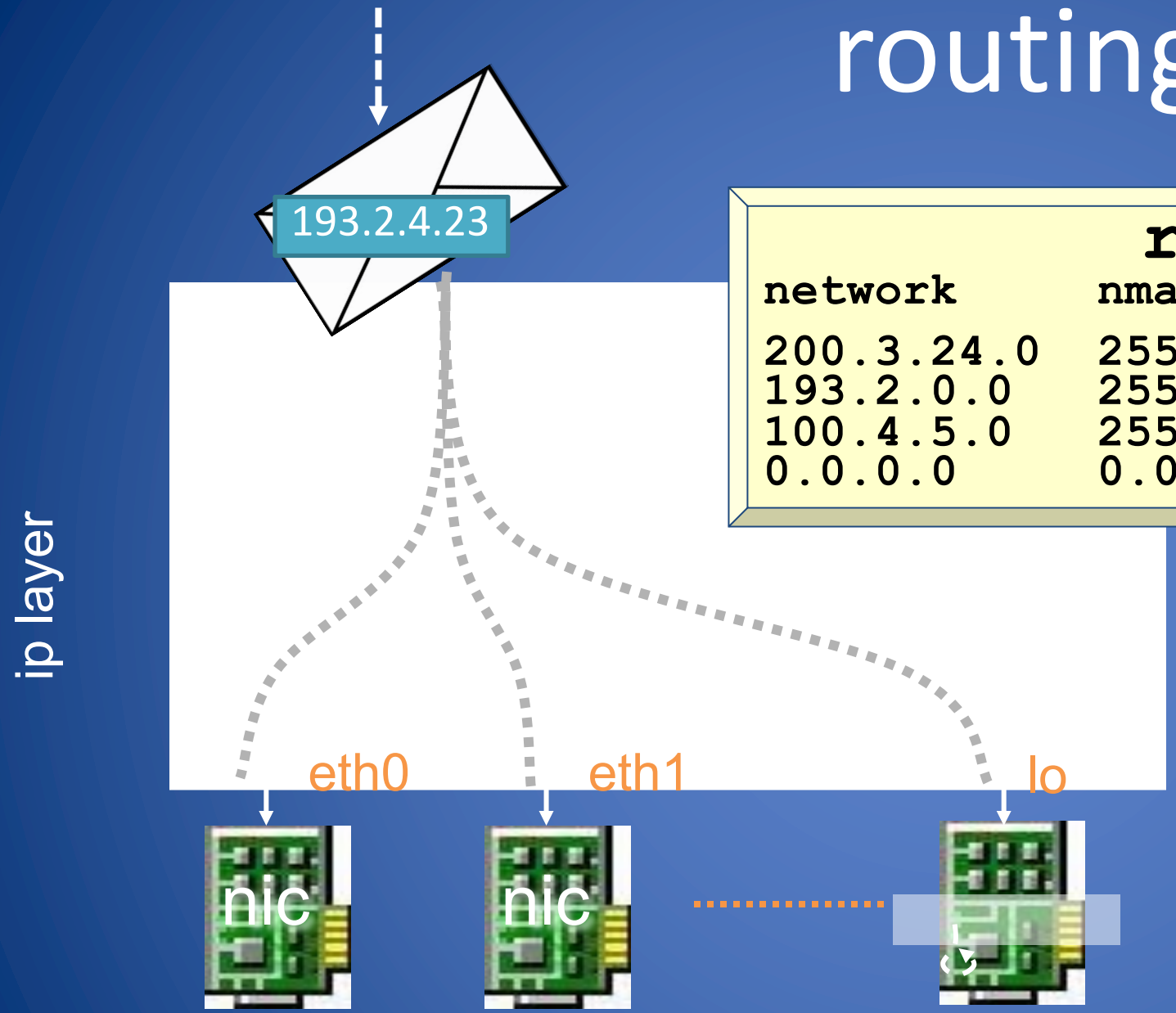


# The ip layer




- the ip layer decides which interface an outgoing packet has to be forwarded to
  - regular hosts have at least two interfaces, nic and loopback

# routing table




**routing table**

| network | nmask | nexthop | int |
|---|---|---|---|
| 200.3.24.0 | 255.255.255.0 | 12.0.0.4 | eth1 |
| 193.2.0.0 | 255.255.248.0 | 11.0.0.2 | eth0 |
| 100.4.5.0 | 255.240.0.0 | 11.0.0.3 | eth0 |
| 0.0.0.0 | 0.0.0.0 | 11.0.0.2 | eth0 |

# routing table usage

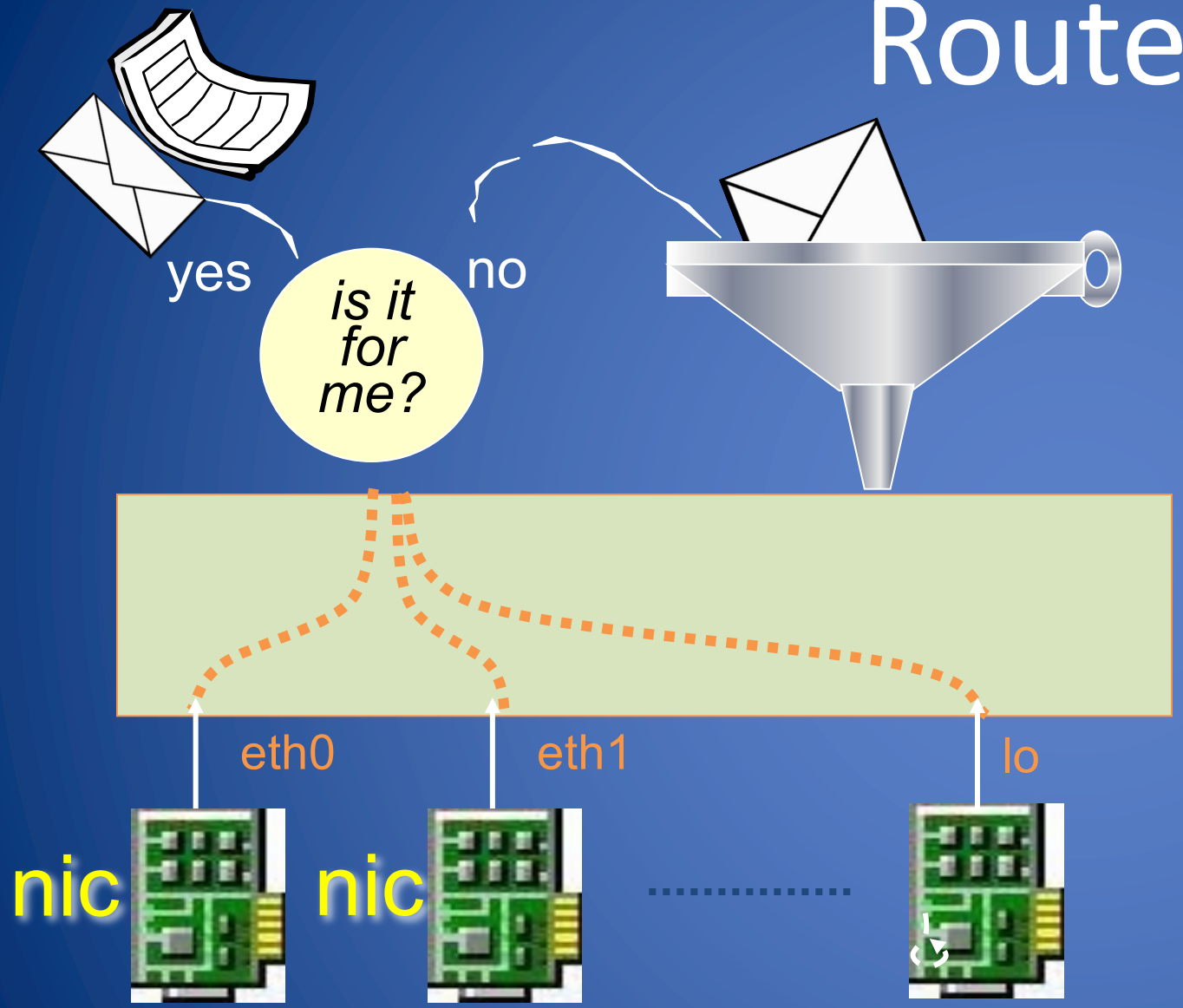

1100 0001.0000 0010.0000 0100.0001 0111

**routing table**

| network | nmask | nexthop | int |
|---|---|---|---|
| 200.3.24.0 | 255.255.255.0 | 12.0.0.4 | eth1 |
| 193.2.0.0 | 255.255.248.0 | 11.0.0.2 | eth0 |
| 100.16.0.0 | 255.240.0.0 | 11.0.0.3 | eth0 |
| 0.0.0.0 | 0.0.0.0 | 11.0.0.2 | eth0 |

| network | nmask |
|---|---|
| 1100 1000.0000 0011.0001 1000.0000 0000 | 1111 1111.1111 1111.1111 1111.0000 0000 |
| 1100 0001.0000 0010.0000 0000.0000 0000 | 1111 1111.1111 1111.1111 1000.0000 0000 |
| 0110 0100.0001 0000.0000 0000.0000 0000 | 1111 1111.1111 0000.0000 0000.0000 0000 |
| 0000 0000.0000 0000.0000 0000.0000 0000 | 0000 0000.0000 0000.0000 0000.0000 0000 |

# Routers




- a router:
  - has more than one network interface card
  - feeds incoming ip packets (that are not for the router itself) back in the routing process
    - this operation is called *relaying* or *forwarding*
  - also called: *gateway, intermediate-system*

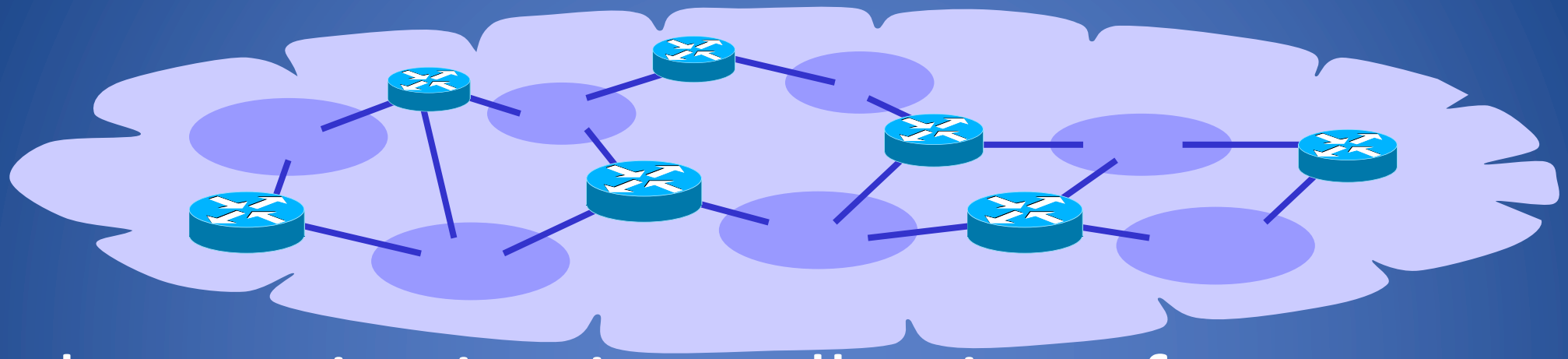
# how to update the routing tables?

- Which are the main features that we need?
  1. Global reachability
  2. Dynamic & Automatic update
  3. Fast convergence time
- Different Routing protocols are available
  - Static and manual routing table update is possible but usually not practical

# Routing protocols

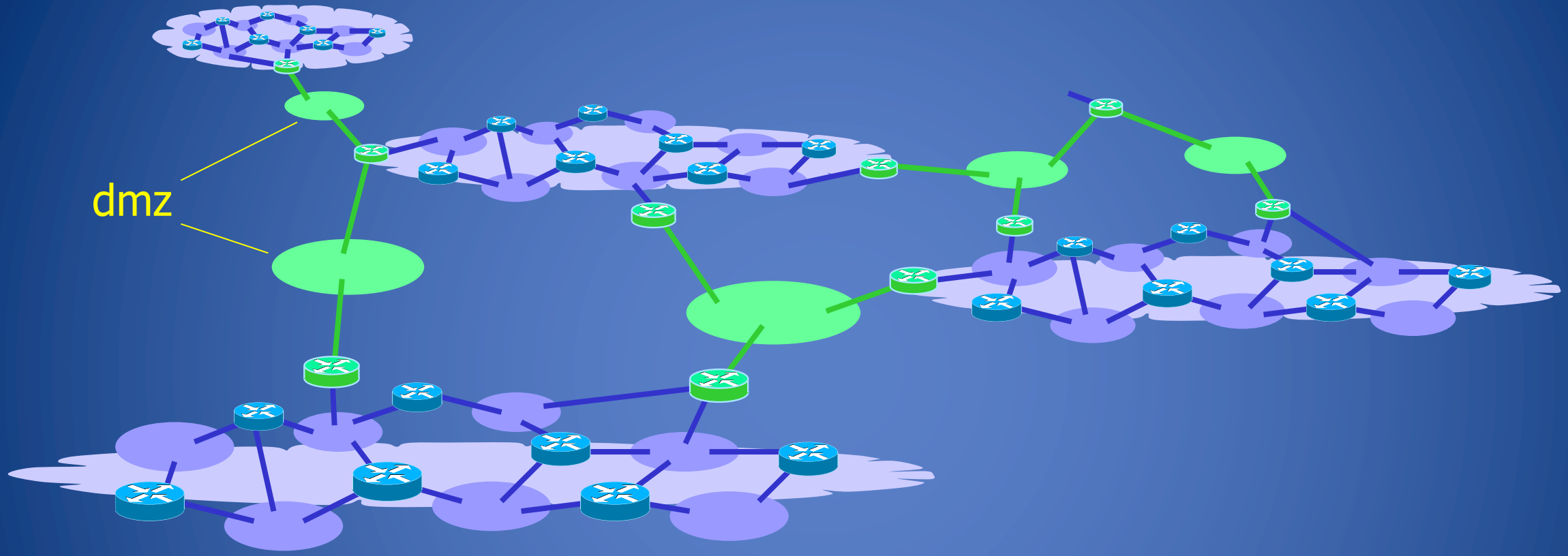
- They fall into two main cathegories:
  - link-state routing protocols
    - approach: talk about your neighbors to everyone
    - each router reconstructs the whole network graph and computes a shortest path tree to all destinations
    - examples: IS-IS, OSPF
  - distance-vector routing protocols
    - approach: talk about everyone with your neighbors
    - update your routing information based on what you hear
    - examples: RIP

# Why interdomain routing?



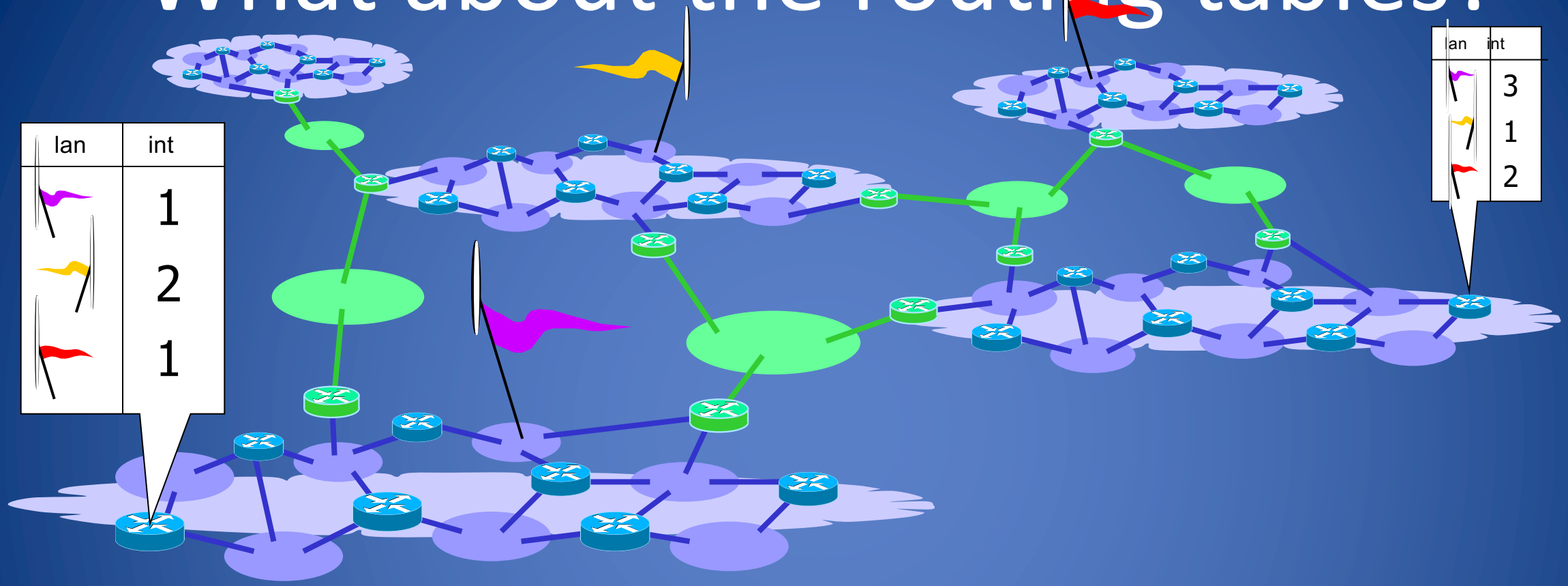
- Each organization is a collection of routers and lan under a single administration
- A routing algorithm may be chosen to automatically update the routing tables

# Why interdomain routing?




- when several organizations join to form the internet they have to set up links between them
  - the added lan are called “demarcation zones”

# What about the routing tables?




- in order to have global connectivity:
  - each router must have a routing entry (possibly the default one) that matches the destination address of the packet
  - this should be true for packets to be delivered locally as well as for packets to be delivered to remote lans

# Border Gateway Protocol (BGP)

- The routing protocol that makes the Internet work
  - A path vector protocol (similar to a distance vector)
- Used by:
  - customers connected to an Internet Service Provider (ISP) or several ISPs
  - transit providers
  - ISPs that exchange traffic at an Internet eXchange Point (IXP) or Neutral Access Point (NAP)
  - customers with very large networks

# Autonomous System

- autonomous systems (ASes) are the cornerstones of BGP
  - used to uniquely identify networks with a common routing policy
  - usually under single ownership, trust and administrative control
- each AS is identified by an *autonomous system number* (asn): 32 bit integer
- two ranges
  - 0-65535 (original 16-bit range)
  - 65536-4294967295 (32-bit range - RFC4893)

# Autonomous System Number

- you may ask an asn to:
  - global asn - to your *regional internet registry* (rir): ripe, arin, apnic, etc.
  - private asn - to your upstream isp
- see also:

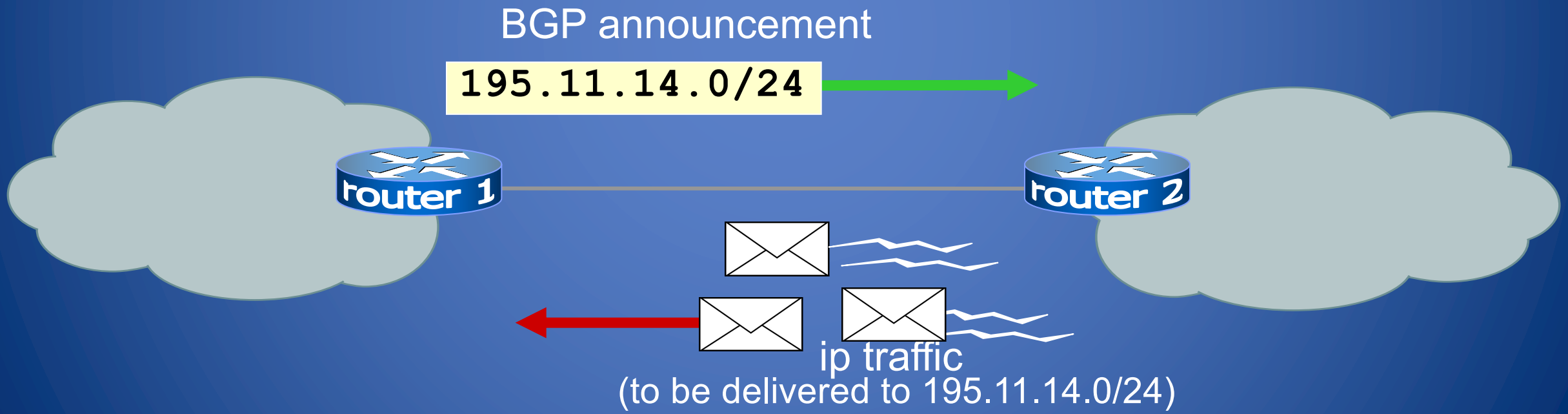
  www.iana.org/assignments/as-numbers

# BGP peering

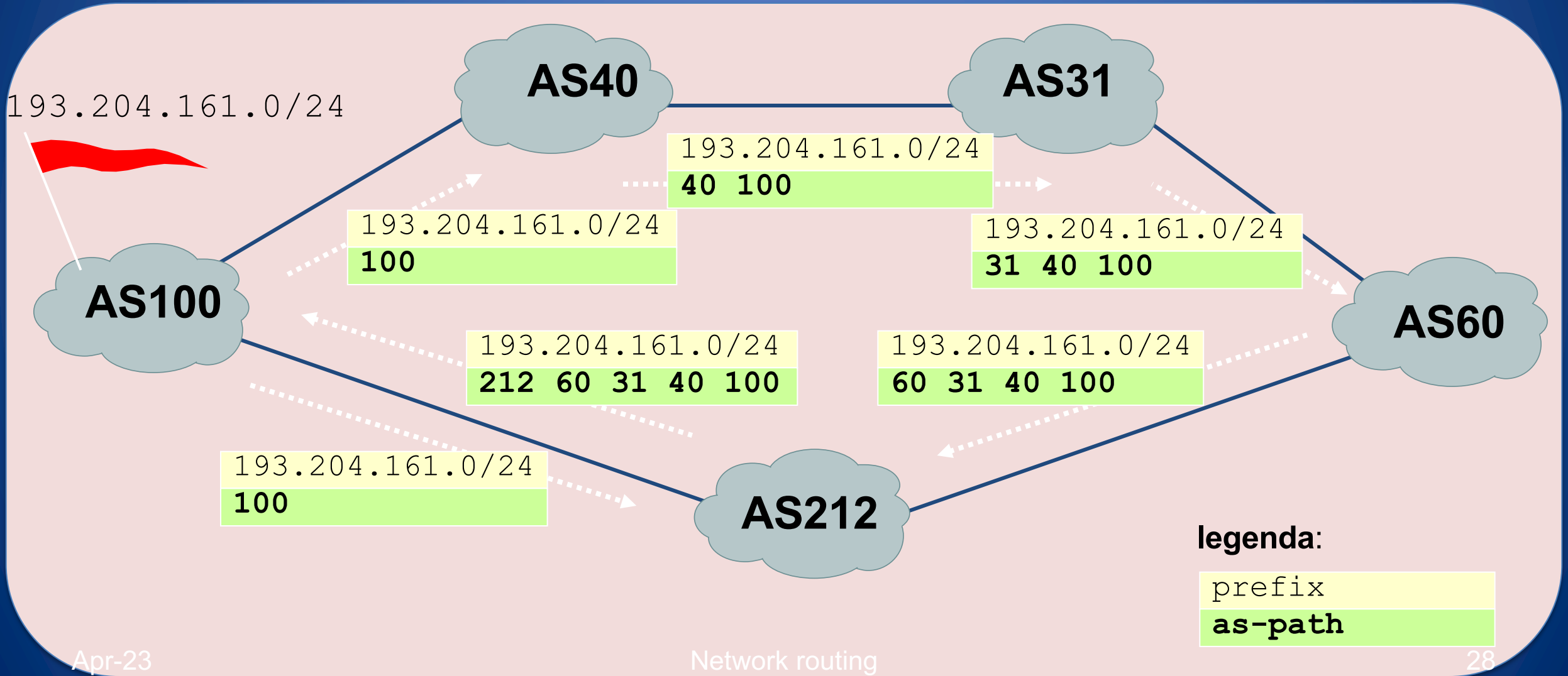
- BGP allows routers to exchange information only if a *peering* session is up
- a BGP peering is the tcp connection (port 179) over which routing information will be exchanged

# Announcements and traffic flows

- BGP allows a router to offer connectivity to another router
- “offering connectivity” means “promising the delivery to a specific destination”



# attributes: AS-path



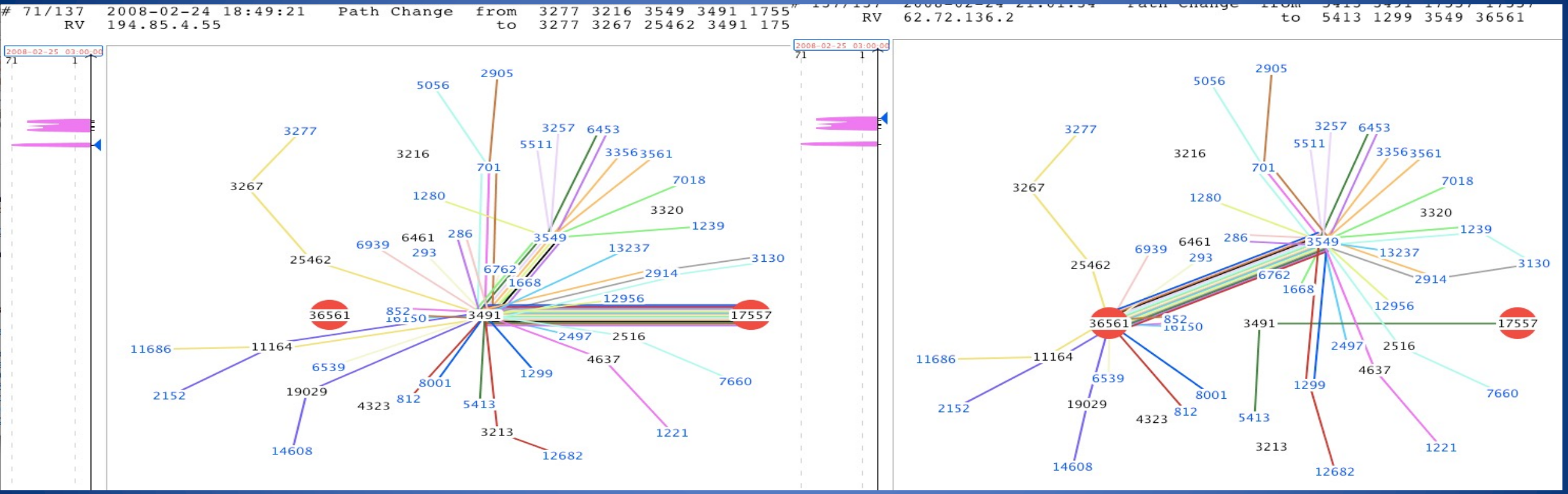
# Looking Glass Server (Demo)

- Provides backbone routing and network efficiency information
  - BGP, Traceroute, and Ping
    - tools that are possible to use with the same transparency that users on ISP network receive directly
- Demo: Hurricane Electric
  - http://bgp.he.net - http://lg.he.net/

# BGP Vulnerabilities

- In the original version BGP has no security mechanisms:
  - No encryption: Eavesdropping
  - No timestamp: Replaying
  - No signature: Hijacking
  - Selective dropping
- Possible attacks:
  - Injecting false information into the global routing database
  - Reroute traffic to perform a Man-in-the-Middle (MITM) attack
  - Trying to create a Denial of Service (DoS) like a black hole in the network

# YouTube Internet Hijacking In Pakistan



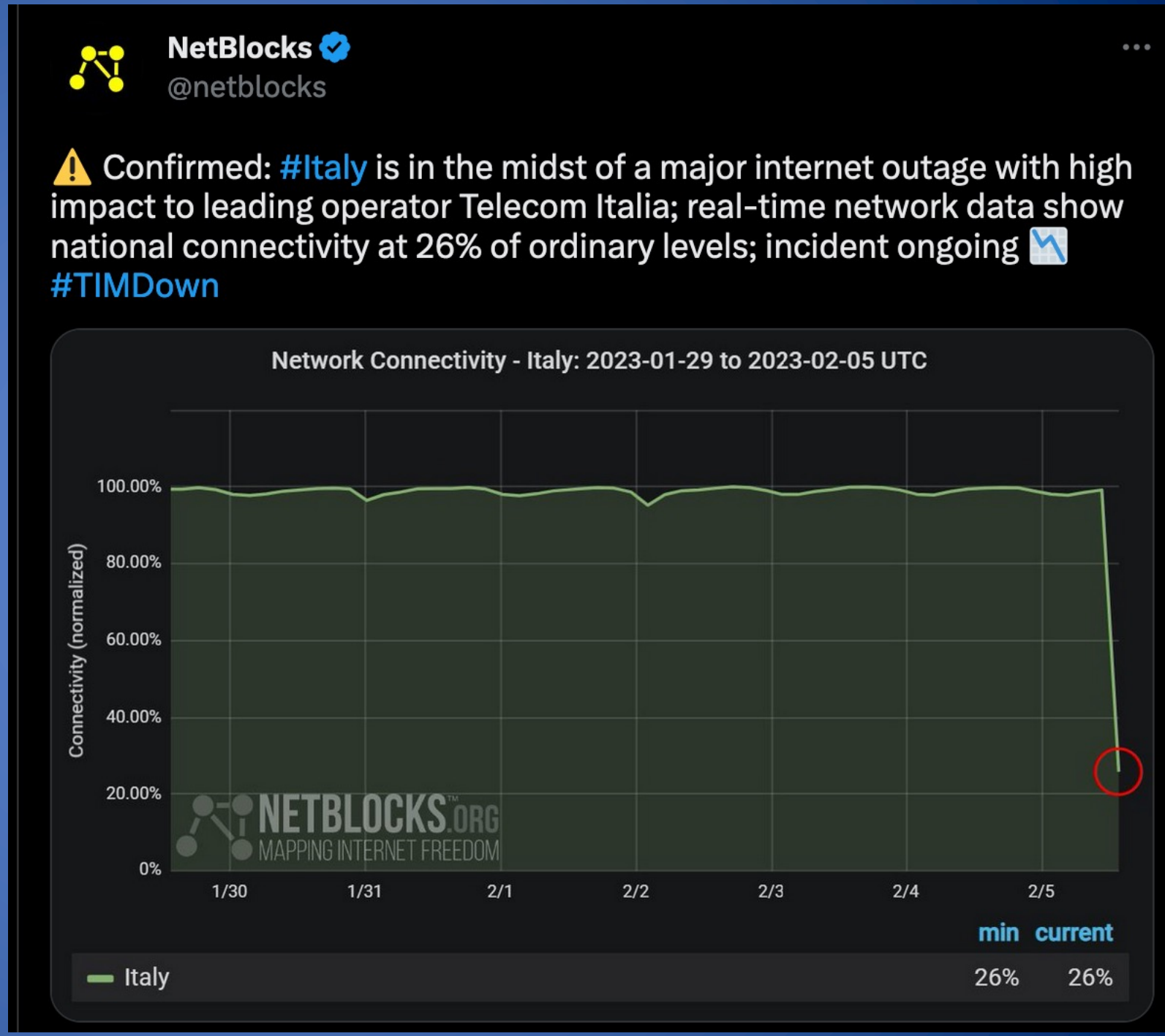

**AS 17557 Pakistan, AS 36561 Youtube**

*[Ripe description using bgplay tool developed at Roma Tre University: https://www.youtube.com/watch?v=IzLPKuAOe50]*

# TIMDown

Stopped the communication for 6 hours on 2/5/23

Probably a human error due to a bad DDOS configuration




Apr-23

# Back to local networks:

# How do you get an IP address?

# Obtaining Host IP Addresses - DHCP

- Networks are free to assign addresses within block to hosts
- Tedious and error-prone: e.g., laptop going from CIT to library to coffee shop
- Idea:  client asks network for IP on connection

# Obtaining Host IP Addresses - DHCP

- Networks are free to assign addresses within block to hosts
- Tedious and error-prone: e.g., laptop going from CIT to library to coffee shop
- Idea:  client asks network for IP on connection

=> But how?  How to send packets with no IP address?

# Broadcast traffic

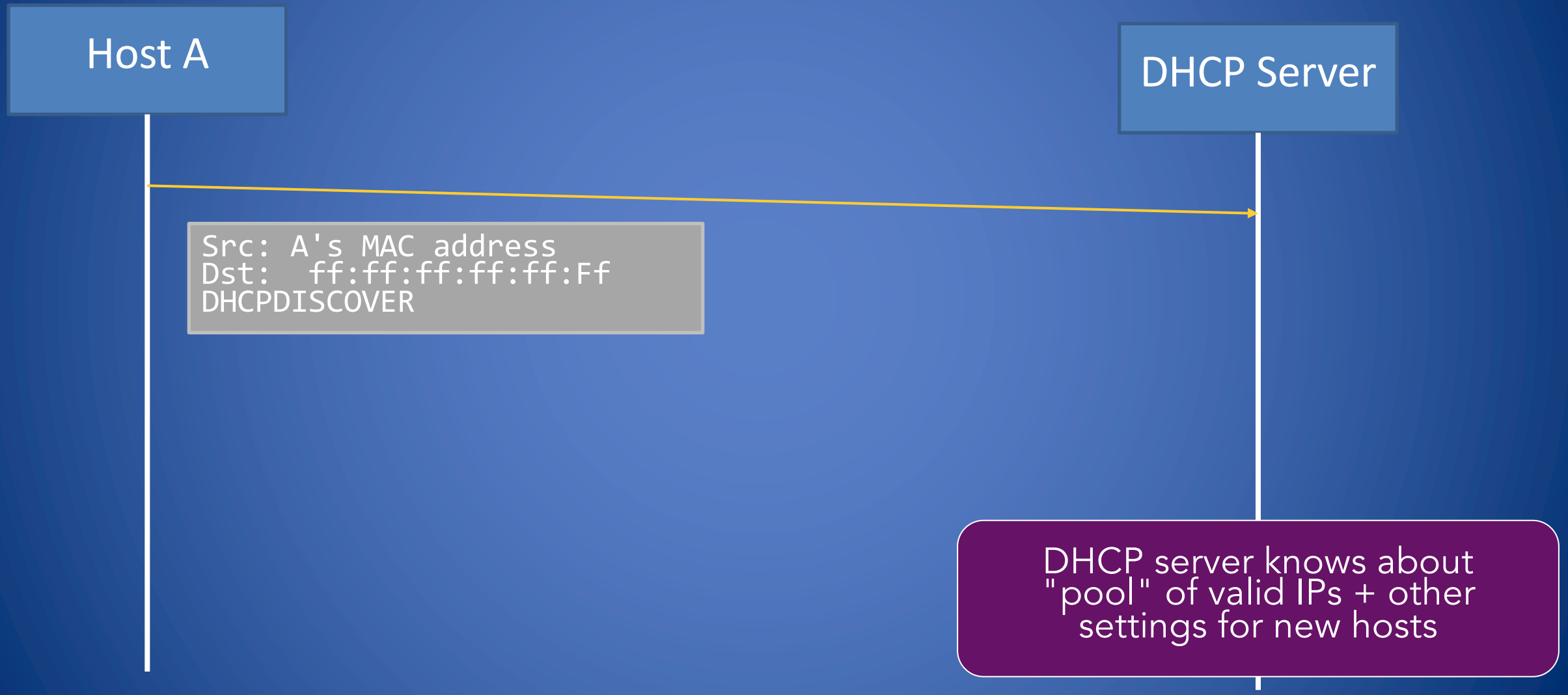
Special MAC address:  ff:ff:ff:ff:ff:ff

- Forwarded to all hosts on network!
- Used for link-layer protocols, particularly for finding IP addresses (DHCP, ARP)

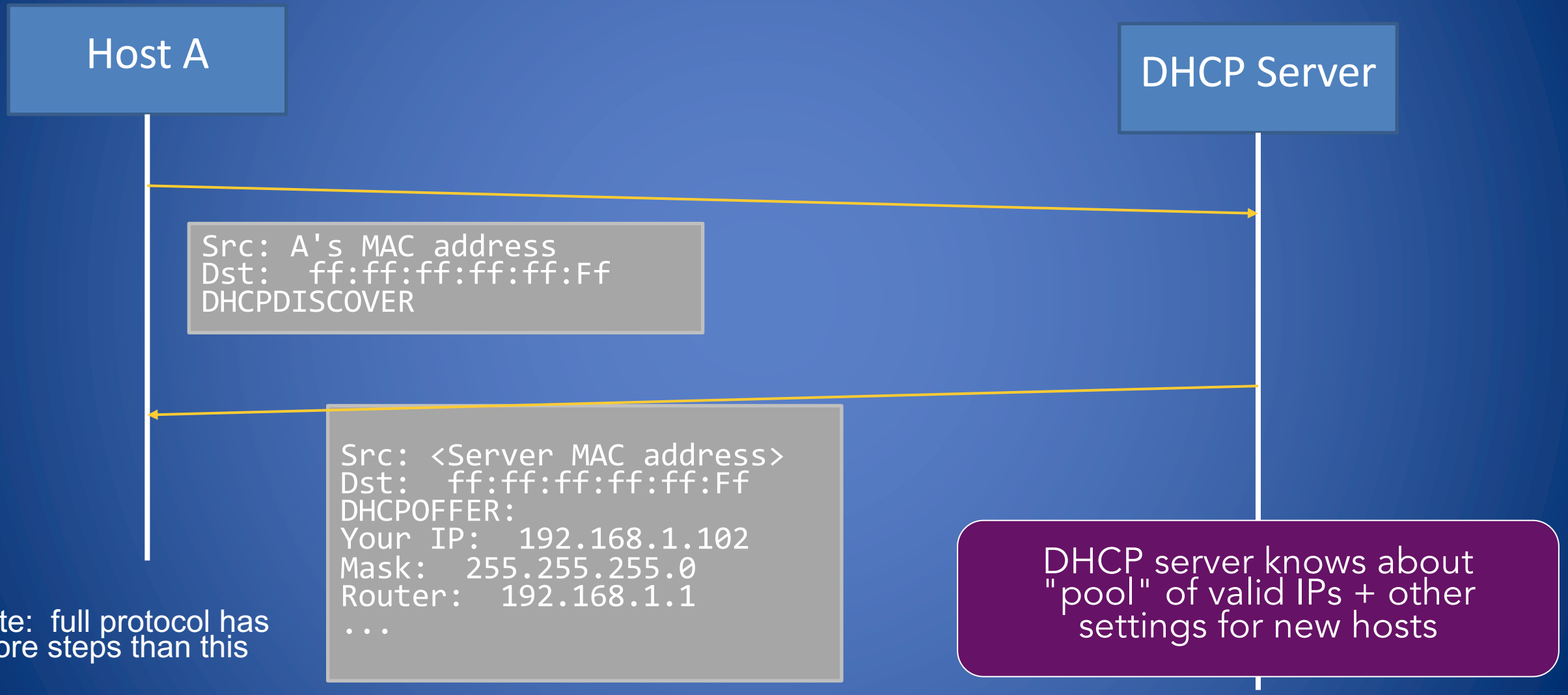
Each IP subnet also has a broadcast address, usually last IP (eg. 192.168.1.255)

# Start of DHCP

# Start of DHCP



# Start of DHCP




Note: full protocol has more steps than this

# Problems with DHCP?

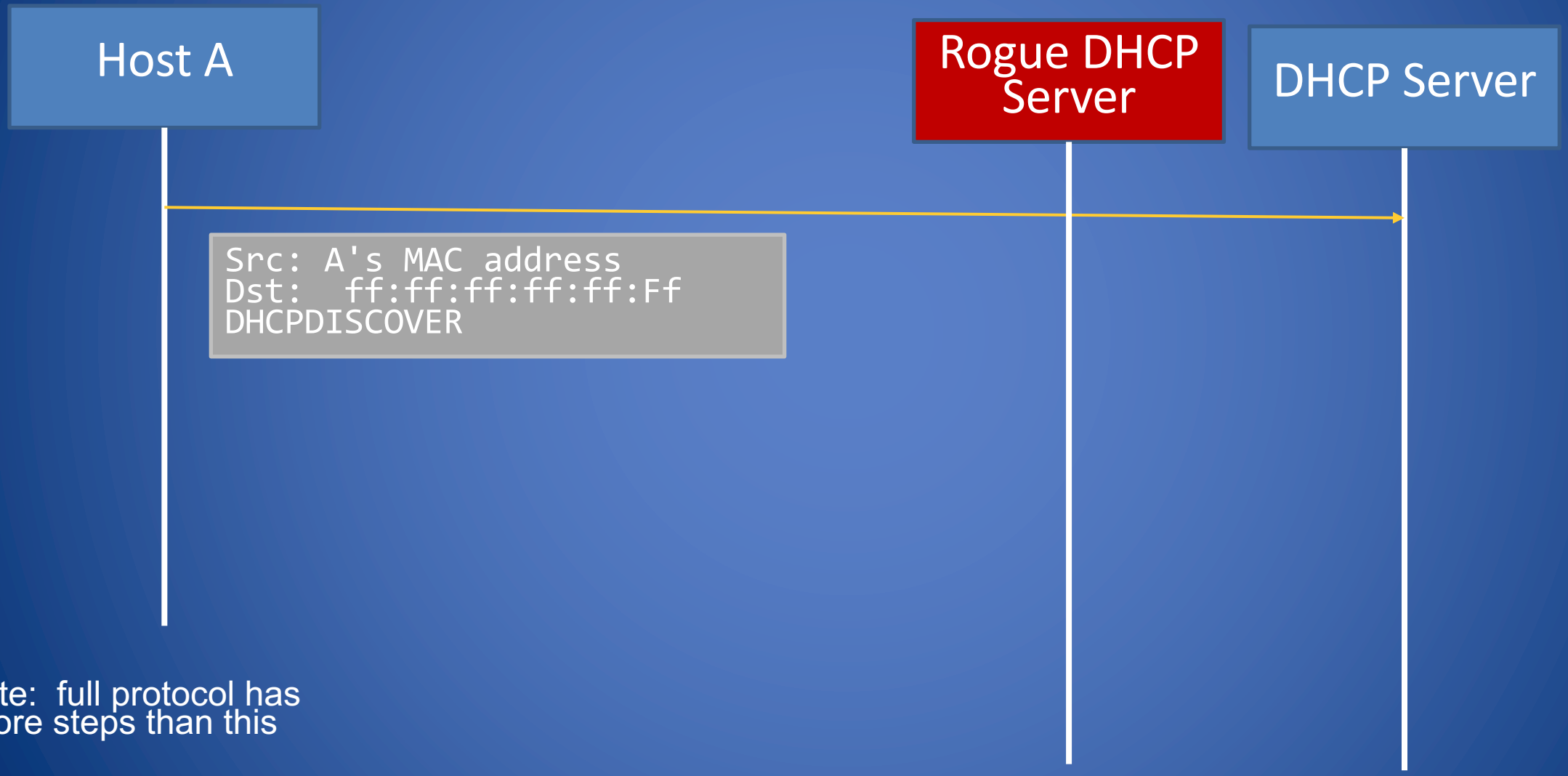
- What happens if a random host decides to be a DHCP server?

# Problems with DHCP?

- What happens if a random host decides to be a DHCP server?

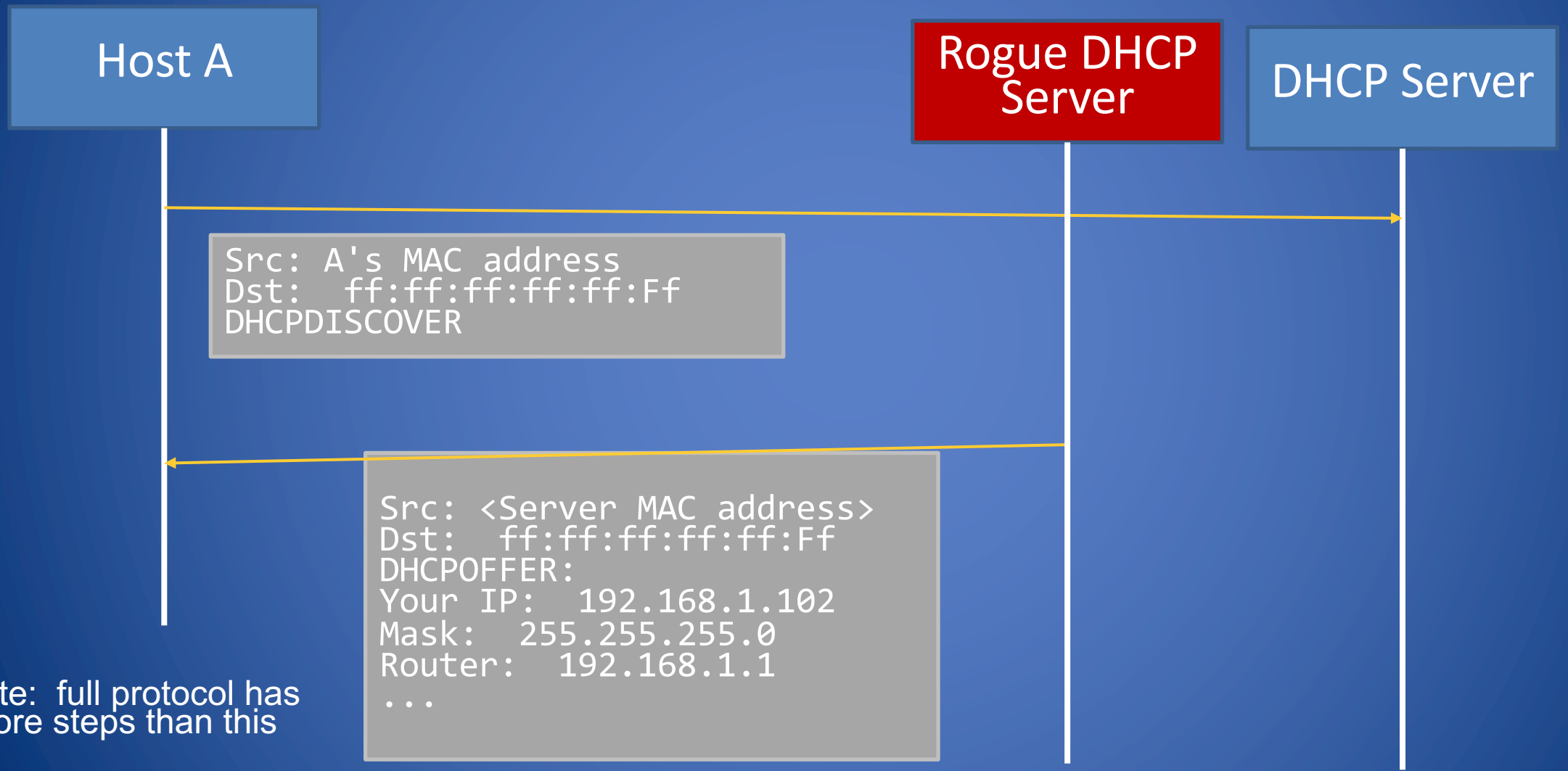
⇒Race condition! If an attacker can make an offer more quickly than the server, can assign a host's IP settings
  - Would be detected by the real DHCP server, though (why?)

# DHCP Spoofing




Note: full protocol has more steps than this

# DHCP Spoofing




Note: full protocol has more steps than this

# How to defend?

# How to defend?

Initial DHCP messages are broadcast, so real server will see the rogue server's response

=> Can detect the attack!

# How to defend?

Initial DHCP messages are broadcast, so real server will see the rogue server's response

=> Can detect the attack!

Why use broadcast? Allows multiple, redundant DHCP servers without extra coordination

# Transport Layer (Ports, TCP, UDP)

# The Transport Layer

Network layer:  moving data between hosts

Transport layer:  Abstraction for getting data data to different *applications* on a host

# The Transport Layer

Network layer:  moving data between hosts

Transport layer:  Abstraction for getting data data to different *applications* on a host

- Multiplexing multiple connections at same IP with port numbers

# The Transport Layer

Network layer:  moving data between hosts

Transport layer:  Abstraction for getting data data to different *applications* on a host

- Multiplexing multiple connections at same IP with port numbers
- Series of packets => stream of data/messages
- May provide:  reliable data delivery

# The Transport Layer

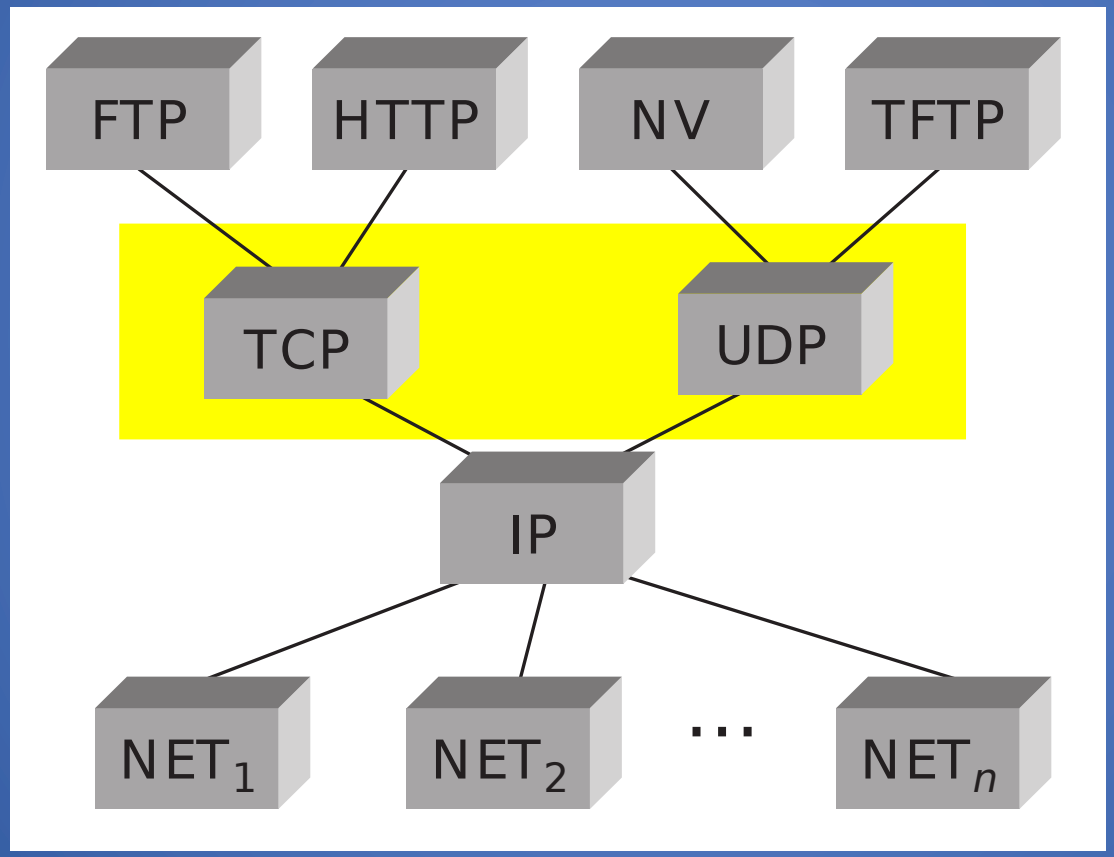
Network layer:  moving data between hosts

Transport layer:  Abstraction for getting data data to different *applications* on a host

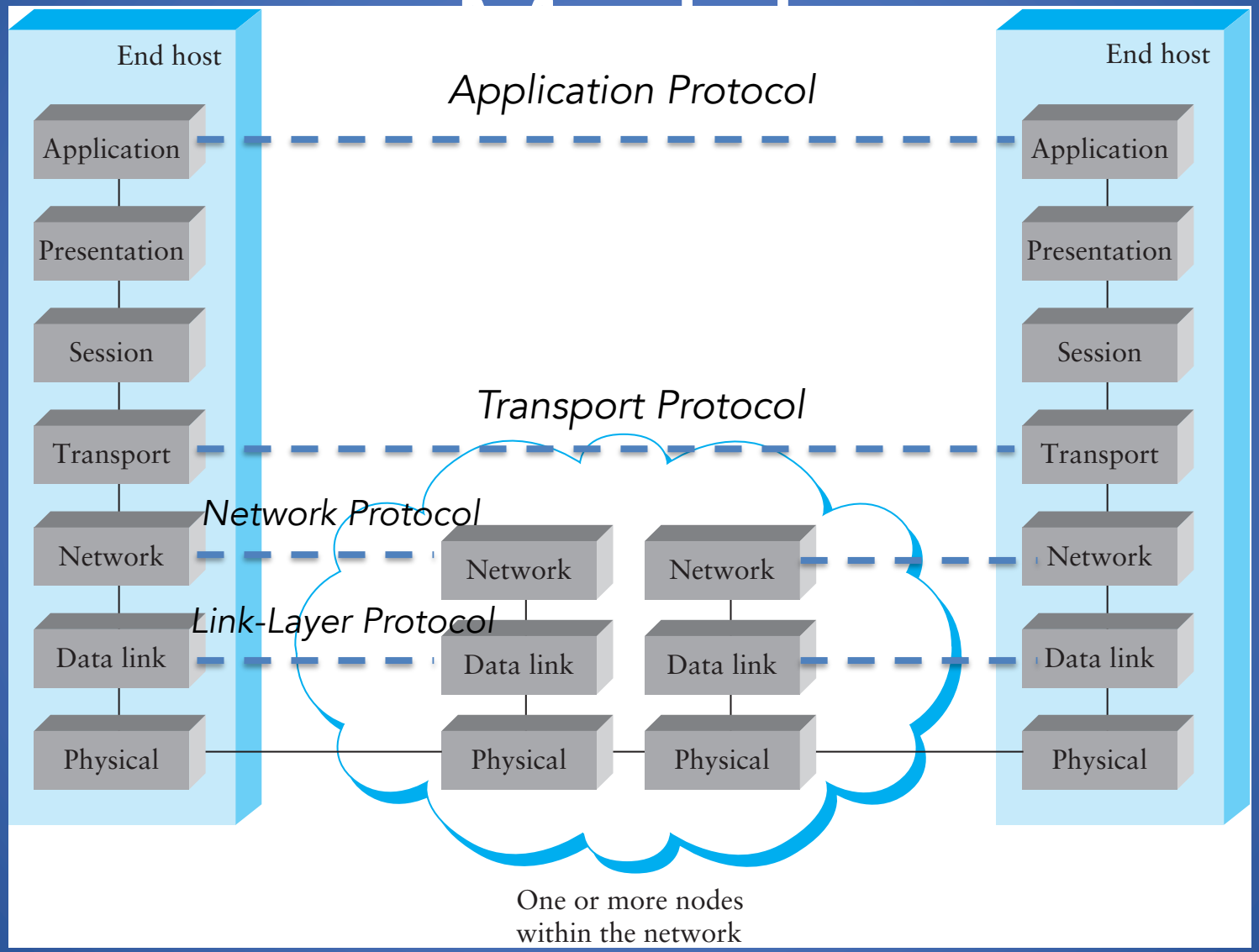
- Multiplexing multiple connections at same IP with port numbers
- Series of packets => stream of data/messages
- May provide:  reliable data delivery

Two key protocols:  TCP, UDP

# Transport Layer



# From earlier: OSI



# What’s a port number?

- 16-bit unsigned number, 0-65535
- Ports define a communication *endpoint*, usually a process/service on a host
- OS keeps track of which ports map to which applications

# What's a port number?

- 16-bit unsigned number, 0-65535
- Ports define a communication *endpoint*, usually a process/service on a host
- OS keeps track of which ports map to which applications

Port numbering

- port < 1024: "Well known port numbers"
- port >= 20000: "ephemeral ports", for general app. use

# Some common ports

| Port | Service |
|---|---|
| 20, 21 | File Transfer Protocol (FTP) |
| 22 | Secure Shell (SSH) |
| 23 | Telnet (pre-SSH remote login) |
| 25 | SMTP (Email) |
| 53 | Domain Name System (DNS) |
| 67, 68 | DHCP |
| 80 | HTTP (Web traffic) |
| 443 | HTTPS (Secure HTTP over TLS) |

# How ports work

Two modes:

- Applications "listen on" or "bind to" a port to wait for new connections

- Hosts make connections to a particular IP and port

# How ports work

Two modes:

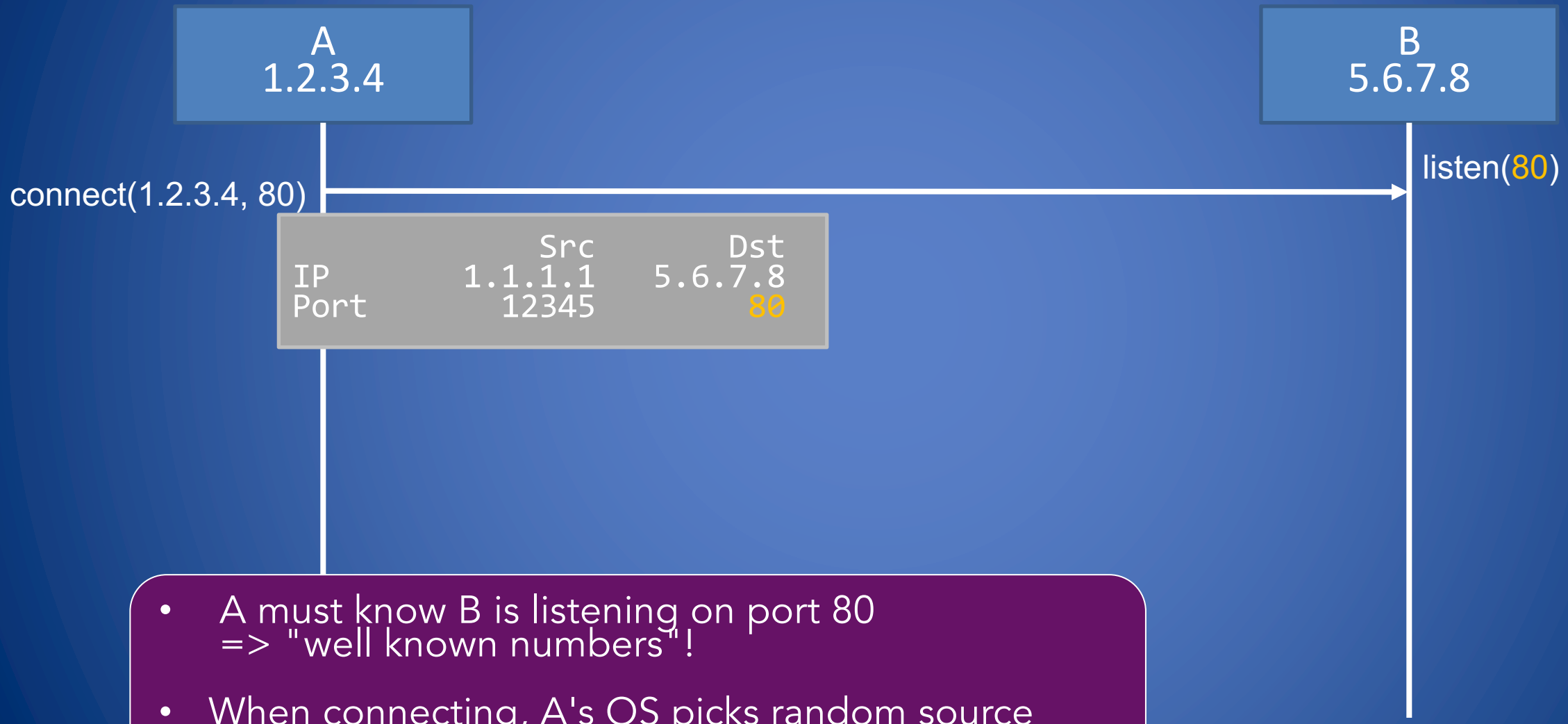
- Applications "listen on" or "bind to" a port to wait for new connections

  => Example: webserver listens on port 80

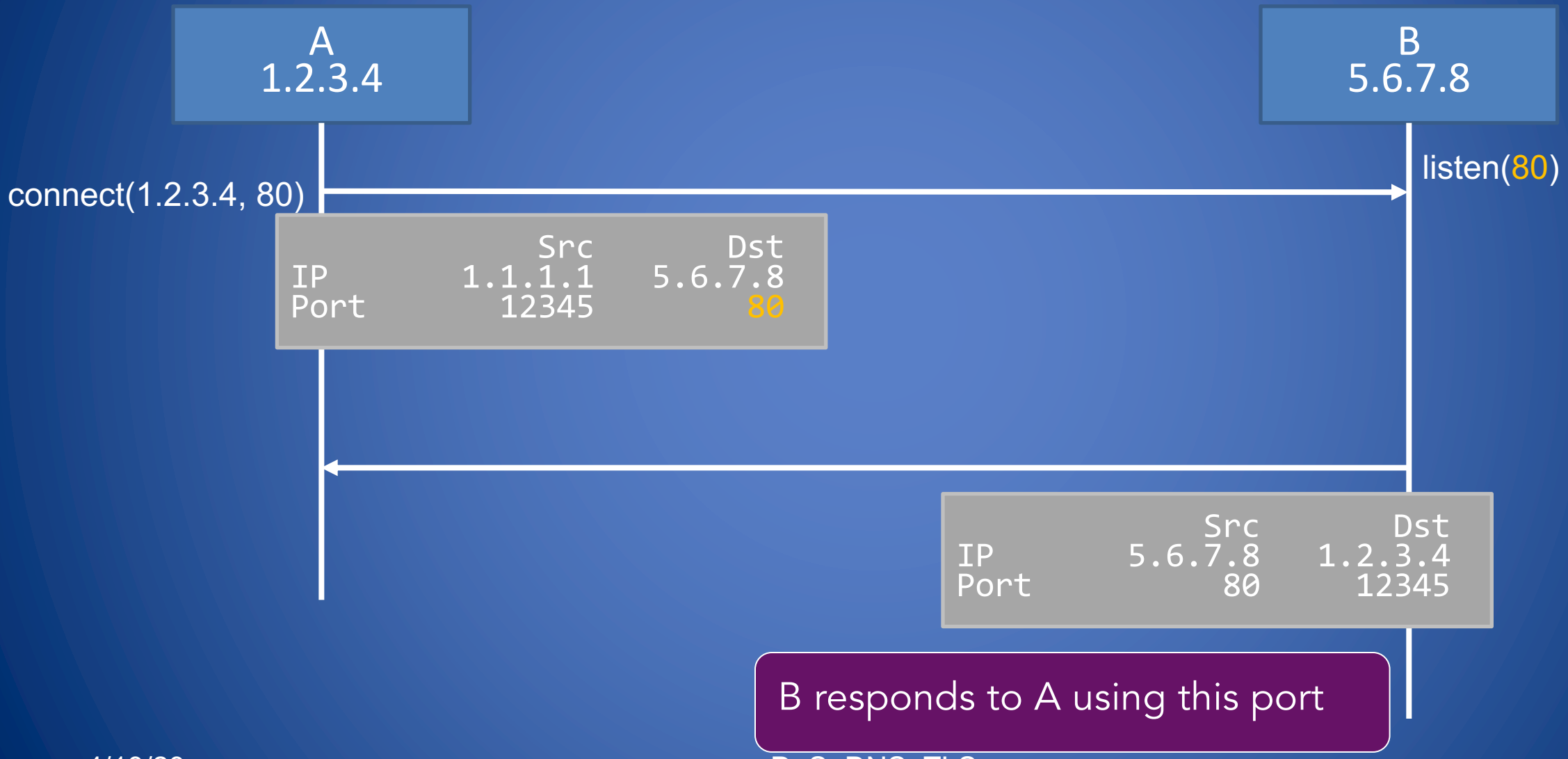
- Hosts make connections to a particular IP and port

  => Example: client connects to <webserver IP>, port 80

  (eg. 1.2.3.4:80)




- A must know B is listening on port 80
  => "well known numbers"!
- When connecting, A's OS picks random source port (eg. 12345), used for its side of connection


A
1.2.3.4
B
5.6.7.8
listen(80)
connect(1.2.3.4, 80)
Src
Dst
IP
1.1.1.1
5.6.7.8
Port
12345
80
Src
Dst
IP
5.6.7.8
1.2.3.4
Port
80
12345
B responds to A using this port

# Sockets

OS keeps track of which application uses which port

Two types:

- Listening ports
- Connections between two hosts (src/dst port)

Socket: OS abstraction for a network connection, like a file descriptor

Table maps: port => socket

Want to know more? Take CS1680!

# Netstat

```
deemer@vesta ~/Development % netstat -an
Active Internet connections (including servers)
Proto Recv-Q Send-Q  Local Address          Foreign Address        (state)
tcp4       0      0  10.3.146.161.51094     104.16.248.249.443     ESTABLISHED
tcp4       0      0  10.3.146.161.51076     172.66.43.67.443       ESTABLISHED
tcp6       0      0  2620:6e:6000:900.51074 2606:4700:3108::.443   ESTABLISHED
tcp4       0      0  10.3.146.161.51065     35.82.230.35.443       ESTABLISHED
tcp4       0      0  10.3.146.161.51055     162.159.136.234.443    ESTABLISHED
tcp4       0      0  10.3.146.161.51038     17.57.147.5.5223       ESTABLISHED
tcp6       0      0  *.22                   *.*                    LISTEN
tcp4       0      0  *.51036                *.*                    LISTEN
tcp4       0      0  127.0.0.1.9999         *.*                    LISTEN
```

`netstat -an`: Show all connections
`netstat -lnp`: Show listening ports + applications using them (as root)

# Why do we care?

Ports define what services are exposed  to the network

- Open port:  can send data to application (reconnaissance, attacks, ...)

- OS and network hardware can monitor port numbers
  - Make decisions on how to filter/monitor traffic

# Demo:  netcat

# Port scanning

What can we learn if we just start connecting to well-known ports?

- Can discover things about the network
- Can learn about vulnerabilities

# Large-scale port scanning

- Can reveal lots of open/insecure systems!
- Examples:
  - shodan.io
  - VNC roulette
  - Open webcam viewers..
  - …
- Also:  penetration testing/vulnerability scanning (more on this later)

# Disclaimer

- Network scanning is easy to detect
- Unless you are the owner of the network, it's seen as malicious activity
- If you scan the whole Internet, the whole Internet will get mad at you (unless done very politely)
- Do NOT try this on the Brown network.  I warned you.

# Scanning I have done

- Scanned IPv4 space for ROS (Robot Operating System)
- Found ~200 “things” using ROS (some robots, some other stuff)

# Transport Layer Protocols

# Transport Layer

- The transport layer supports one or more of the following features
  A. Reliable data transfer (resending of dropped packets)
  B. In-order delivery of segments of file or media stream
  C. Congestion control (request longer/shorter segments)
  D. Ability to distinguish multiple applications on same host via ports (16-bit numbers)
- The main transport layer protocols are
  - UDP (supports B, D)
  - TCP (supports A, B, C, D)

# User Datagram Protocol (UDP)

- Stateless, unreliable transport-layer protocol
- Can distinguish multiple concurrent applications on a single host
- No delivery guarantees or acknowledgments
  - Efficient
  - Suitable for audio/video streaming and voice calls
  - Unsuitable for file transmission and text messaging

# Transmission Control Protocol (TCP)

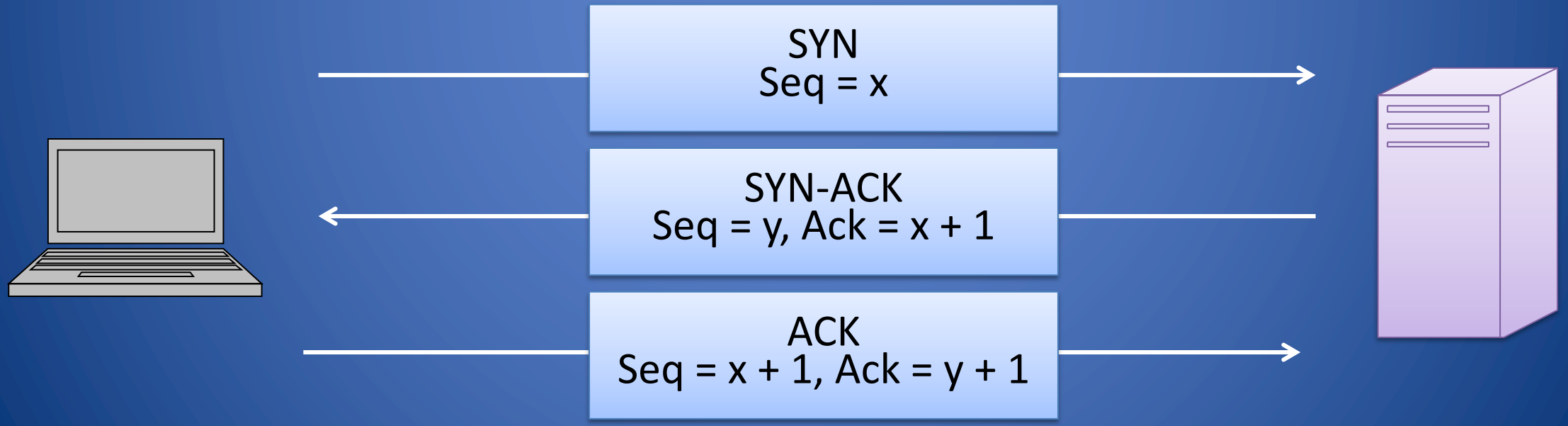
- Stateful protocol for reliable data transfer, in-order delivery of messages and ability to distinguish multiple applications on same host
  - HTTP and SSH are built on top of TCP
- TCP packages a data stream it into segments transported by IP
  - Order maintained by marking each packet with sequence number
  - Every time TCP receives a packet, it sends out an ACK to indicate successful receipt of the packet
- TCP generally checks data transmitted by comparing a checksum of the data with a checksum encoded in the packet

# TCP Packet Format

| Bit Offset | 0-3 | 4-7 | 8-15 | 16-18 | 19-31 |
|---|---|---|---|---|---|
| 0 | **Source Port** | | | **Destination Port** | |
| 32 | **Sequence Number** | | | | |
| 64 | **Acknowledgment Number** | | | | |
| 96 | Offset | Reserved | Flags | **Window Size** | |
| 128 | Checksum | | | Urgent Pointer | |
| 160 | Options | | | | |
| >= 160 | Payload | | | | |

# Establishing TCP Connections

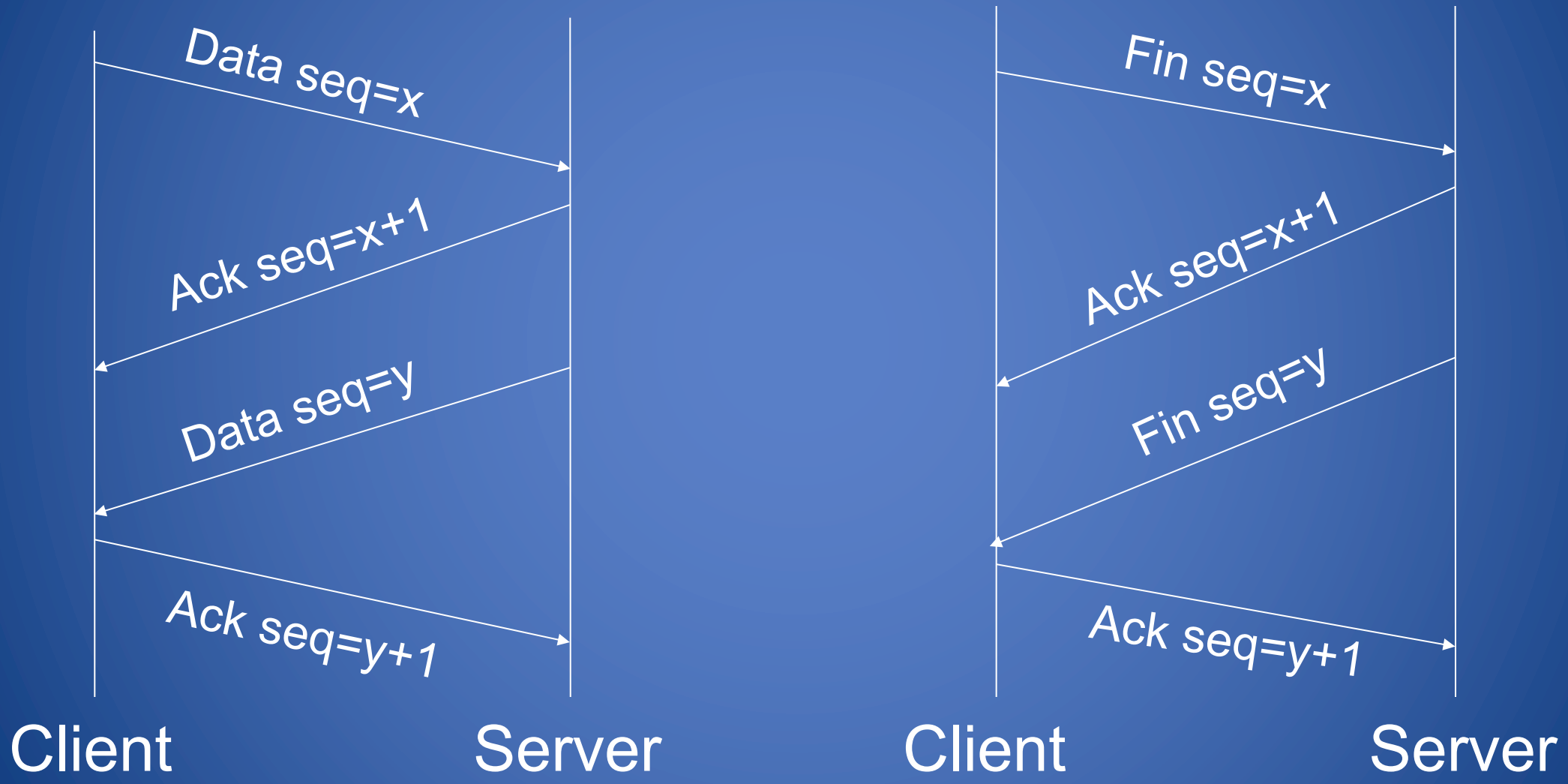
- TCP connections are established through a three-way handshake
- The server generally is a passive listener, waiting for a connection request
- The client requests a connection by sending out a SYN packet
- The server responds by sending a SYN/ACK packet, acknowledging the connection
- The client responds by sending an ACK to the server, thus establishing connection



# TCP Data Transfer

- The three way handshake initializes sequence numbers for the request and response data streams
- The TCP header includes a 16 bit checksum of the payload and parts of the header, including source and destination
- Acknowledgment or lack thereof is used by TCP to keep track of network congestion and control flow
- TCP connections are cleanly terminated with a 4-way handshake
  - The client which wishes to terminate the connection sends a FIN message to the other client
  - The other client responds by sending an ACK
  - The other client sends a FIN
  - The original client now sends an ACK, and the connection is terminated

# TCP Data Transfer and Teardown



# Clicker Question (2)

Eve is once again up to no good. She decides to modify the payload of a TCP packet that Alice sends to Bob by randomly flipping a bit. Would Bob be able to detect this?

A. Yes, since most likely the checksum will not match
B. Yes, since the packet will be totally corrupted
C. No, since there are no security features in TCP
D. No, since it is computationally infeasible

# Clicker Question (2) - Answer

Eve is once again up to no good. She decides to modify the payload of a TCP packet that Alice sends to Bob by randomly flipping a bit. Would Bob be able to detect this?

**A. Yes, since most likely the checksum will not match**

B. Yes, since the packet will be totally corrupted

C. No, since there are no security features in TCP

D. No, since it is computationally infeasible

# Attacks at the transport layer?

# Denial of Service (DoS) Attack

- Cyberattacker disrupts the availability of a service
- Usually targets specific computer, device, or IP address
  - Attacker overwhelms server with network packets
  - Or, attacker takes advantage of known vulnerability in server to cause it to crash

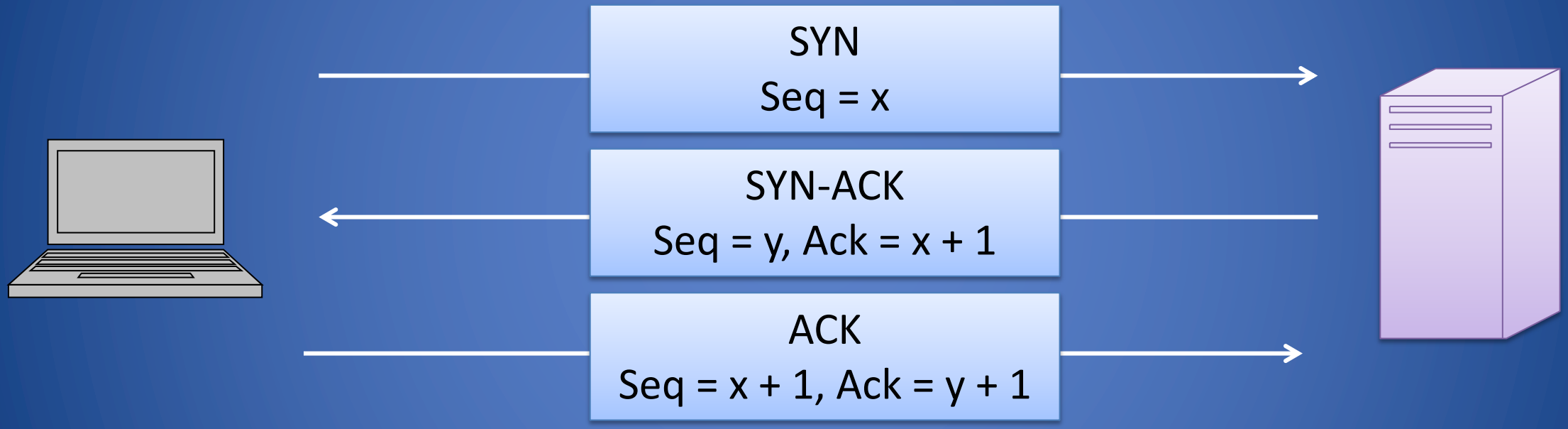
- Distributed denial of service (DDoS) attack: DoS with many sources of malicious traffic
  - Requires a botnet, i.e., coordinated network of multiple machines to send traffic at once
  - Dyn attack (2016): utilized Mirai IoT botnet
  - GitHub attack (2018): exploited memcaching vulnerability to amplify spoofed requests
  - Hacktivist with several userd

https://www.nbcnews.com/tech/security/hacktivists-new-veteran-target-russia-one-cybers-oldest-tools-rcna20652

# Denial of Service

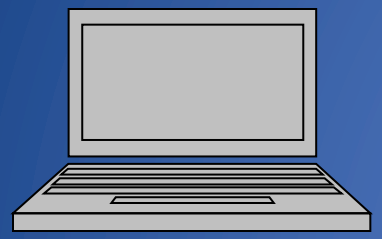
- How do you DoS a target's Internet access?
  - Send many, many, many tiny packets—enough to fill up a bottleneck in the target's router's ability to process packets
  - Send very large packets—enough to fill up a bottleneck in the target's network / Internet connection
- No isolation between Internet traffic (*note OS sec similarities*)
- Needs a good amount of attacker resources
  - At least as much bandwidth as the bottlenecks above
  - Might be hard to get this (without being blocked by a firewall)
  - …but just use a botnet (see Mirai; might be hard to get this too…)

# Recall: TCP Establishment Handshake




What happens on the server?

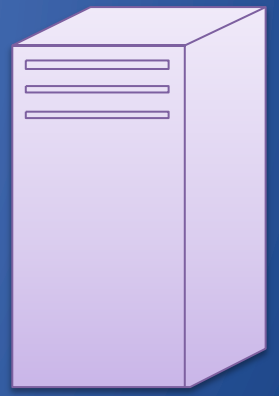
# Recall: TCP Establishment Handshake



SYN
Seq = x

Server creates **state** associated with connection (client IP, port, counters…)

SYN-ACK
Seq = y, Ack = x + 1



What happens if client doesn't send this ACK?

ACK
Seq = x + 1, Ack = y + 1

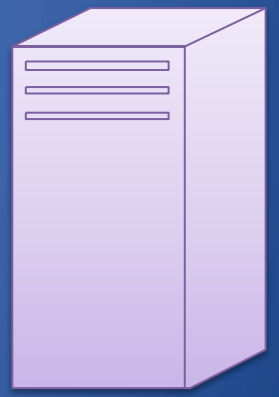
# Recall: TCP Establishment Handshake

SYN
Seq = x

SYN-ACK
Seq = y, Ack = x + 1

ACK
Seq = x + 1, Ack = y + 1

Server creates **state** associated with connection (client IP, port, counters…)



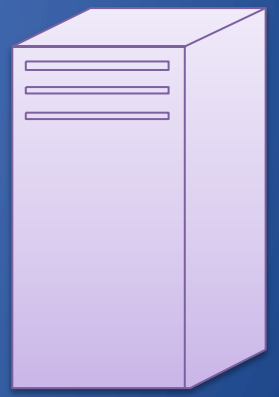
What happens if adversary doesn't send this ACK?

# Recall: TCP Establishment Handshake

SYN
Seq = x

Server creates **state** associated with connection (client IP, port, counters…)

SYN-ACK
Seq = y, Ack = x + 1



What happens if adversary doesn't send this ACK?

ACK
Seq = x + 1, Ack = y + 1

A single SYN from adversary forces the server to spend some memory…

# Recall: TCP Establishment Handshake


What happens if adversary doesn't send this ACK?

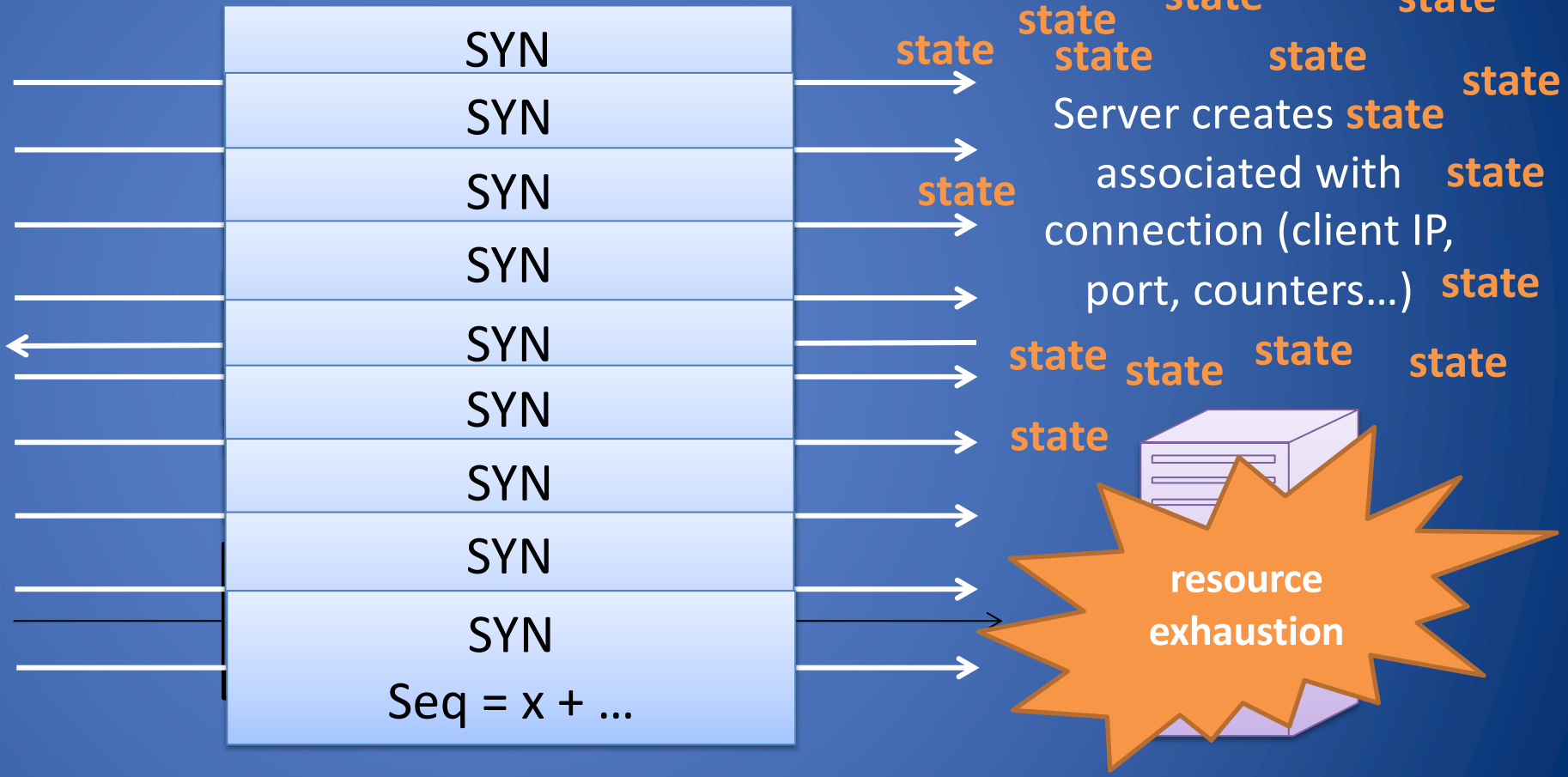
A single SYN from adversary forces the server to spend some memory…

# IP Address Spoofing

- IP packets can be created with a forged source address field
  - Similar to sending a package with a fake return address
- Can be used to mask traffic source or to impersonate a specific system
- Can be used for good!
  - Developers may have to simulate large volume of user traffic to test web service performance

- Use in DDoS attacks:
  - Randomly generated source addresses makes mitigation and forensic analysis difficult
  - Reflection/amplification attack: Malicious packet may list its source address to be the address of another targeted device, triggering a flood of traffic to it from another network service
    - Example: DNS reflection

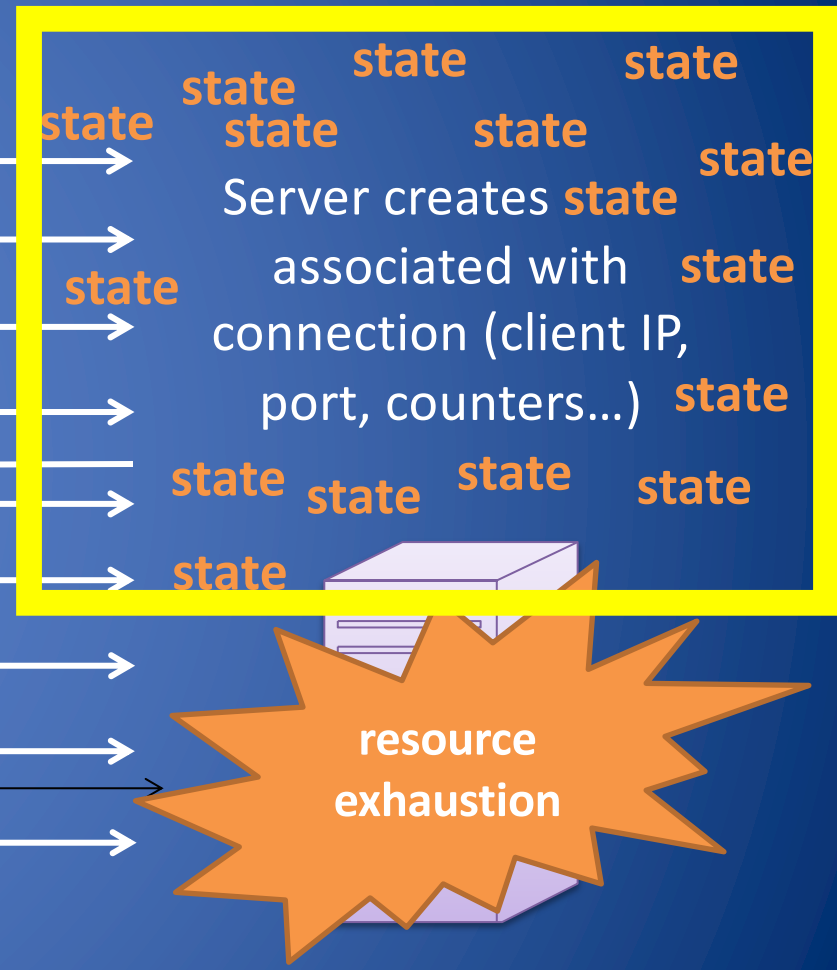
# Recall: TCP Establishment Handshake




What happens if adversary doesn't send this ACK?

A single SYN from adversary forces the server to spend some memory…

# Recall: TCP Establishment Handshake


What happens if adversary doesn't send this ACK?

A single SYN from adversary forces the server to spend some memory…

# SYN Flooding

- Attacker targets server memory rather than network capacity
- Every (unique) SYN forces the server to spend memory
  - Server can't necessarily clear up the memory (at least, not right away)
- What happens when the server runs out of memory?
  - Refuse new connection?
    - Legitimate new users can't access service
  - Evict old connections?
    - Legitimate old users get kicked out

# What We Have Learned

- IP address space allocation
- ARP protocol
- ARP poisoning attack
- Transport layer protocols
  - TCP for reliable transmission
  - UDP when packet loss/corruption is tolerated
- Lack of built-in security for link, network, and transport layer protocols
  - Security enhanced protocols have been developed for these layers
  - Alternate solution is to provide security at application layer