Network Security Intro: A Historical Perspective

CS 591 Guest Lecture

Charles V. Wright

cvwright@cs.pdx.edu

Roadmap for today's talk

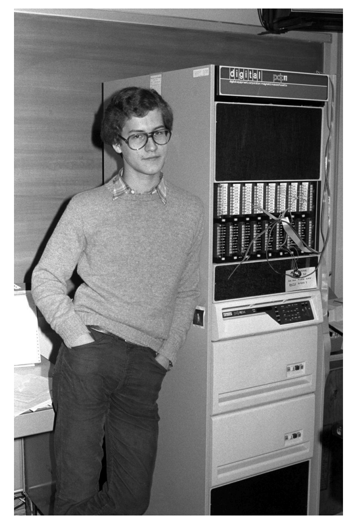
- A brief history lesson
  - Graybeards and green screens
- Basic network protocols
  - TCP/IP and friends.. The series of tubes
- What could possibly go wrong?
  - Quite a lot, actually…

Graybeards and green screens

A BRIEF HISTORY LESSON

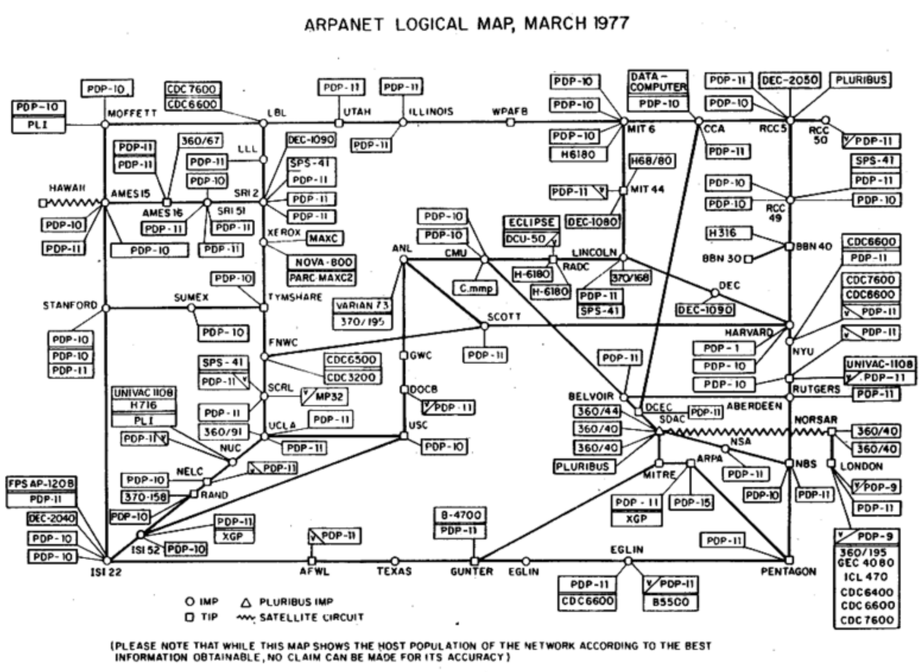
In the beginning…

- Computers were big
- Computers were expensive
- Computers were slow



In the beginning…

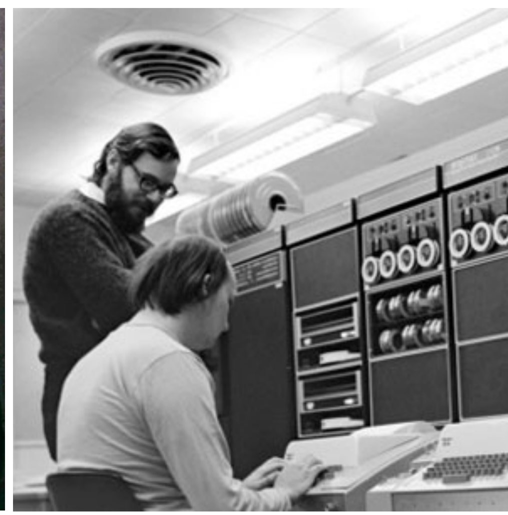
- Networks were small
- Networks were expensive
- Networks were slow



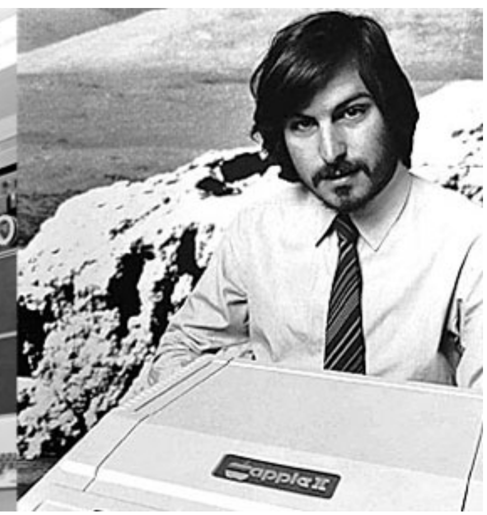
In the beginning…

- Not many organizations had computers
- Not many people got to use the computers
- Users had to be geographically near to their computers

Microsoft in 1979



Ken Thompson and Dennis Ritchie at AT&T Bell Labs

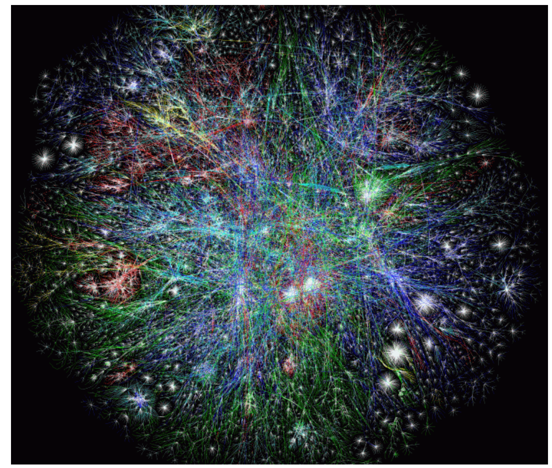


Steve Jobs

As a result…

- Extra computation or communication for security was very, very costly
- The community of computer users and admins was relatively small
  - If somebody misbehaved, it was easier to **find them** and **make them stop**
- Overall, the risks were low and protection was expensive

Today

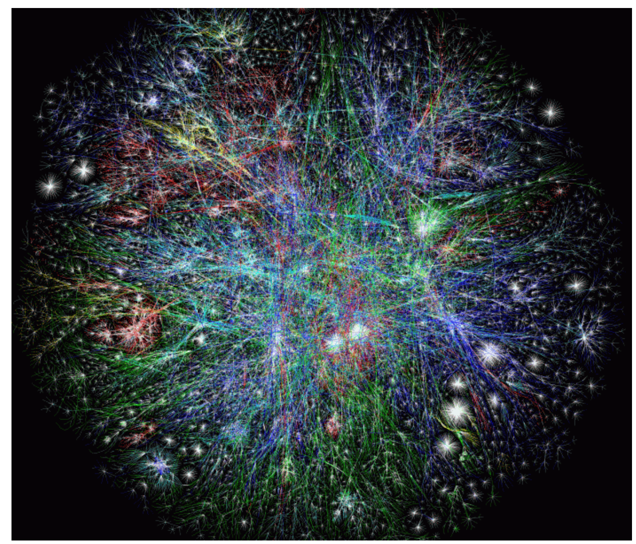


- Computers are cheap
- Computers are fast
- Networks are fast
- The Internet is huge
  - Lots of extra compute power and bandwidth
  - Lots of devices that aren't well-maintained
  - Lots of people who aren't so nice
  - Lots of people beyond your jurisdiction

Tell me if you've heard this story before…

- Design decisions made for this system

- Don't work so well for this one



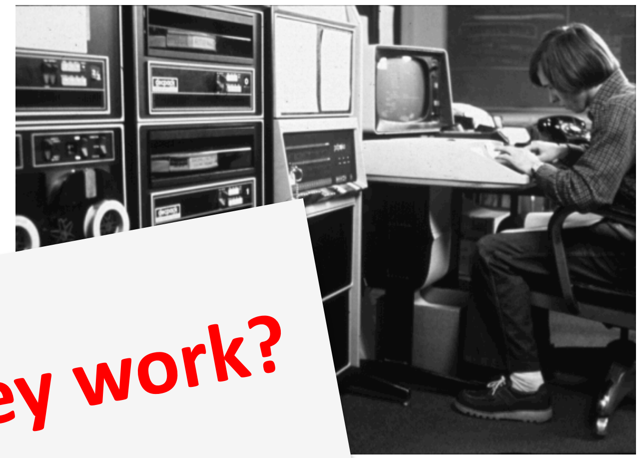
Tell me if you've heard this story before…

- Design decisions made for this system

- Don't w… for this

But WHY don't they work?

And HOW do they fail?



Goals for today's talk (1)

- Understand network security problems by digging deeper into the underlying technology
  - What is it?
  - How does it work?
  - What are its assumptions? (explicit AND implicit)
  - What happens if those assumptions are violated?
  - How could the assumptions be violated?

Goals for today’s talk (2)

- Understand why security has been such a problem for networks and the Internet

Goals for today's talk (3)

- Understand **what NOT to do** when writing a protocol or designing a networked system

The right perspective…

- The people who designed these old protocols weren't lazy or incompetent
  - In fact, they were probably smarter than all of us
- But in most cases, it was their first experience designing such large and complex systems
  - And they didn't have security as a priority
  - Sound like anyone you know?
- We're not here to laugh at their mistakes, but to learn from them
- **Don't let this happen to you!**

TCP/IP and friends… Or, the series of tubes…

BASIC NETWORK PROTOCOLS

Basic Network Protocols

- How do computer networks work?
- How does a packet make its way across the network?
- What's TCP/IP? How is that related to Ethernet?
- Acronym soup: FTP, DNS, HTTP, SMTP, BGP

Caveats

- This is going to be a whirlwind tour of computer networks and internet protocols
  - Not intended to be complete or thorough
  - Just enough to see where things went wrong
- If you're interested in learning more about networks
  - Wikipedia is a pretty good starting point
  - For more detailed info, check out some books
    - *TCP/IP Illustrated*, by Richard Stevens
    - *Computer Networks* by Andrew Tanenbaum
    - *Computer Networking: A Top-Down Approach* by Kurose and Ross
  - Better yet, take CS 494 / 594

What the heck is all this stuff?

Network Protocol “Stacks”

Application Layer: HTTP, SMTP, …
- Send messages to people, send files, …

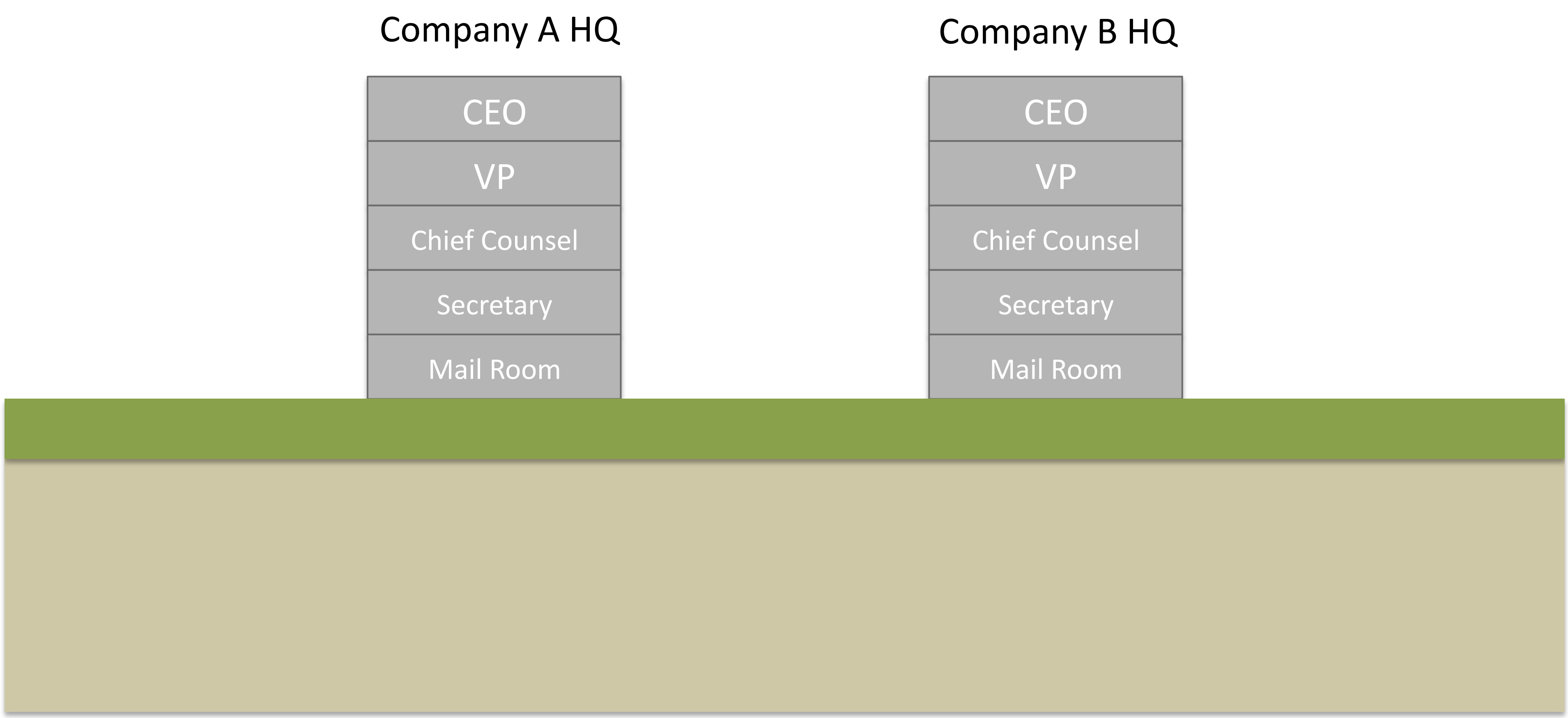
Transport Layer: TCP
- Stream-oriented I/O

Network Layer: IP
- Sends messages across multiple hops

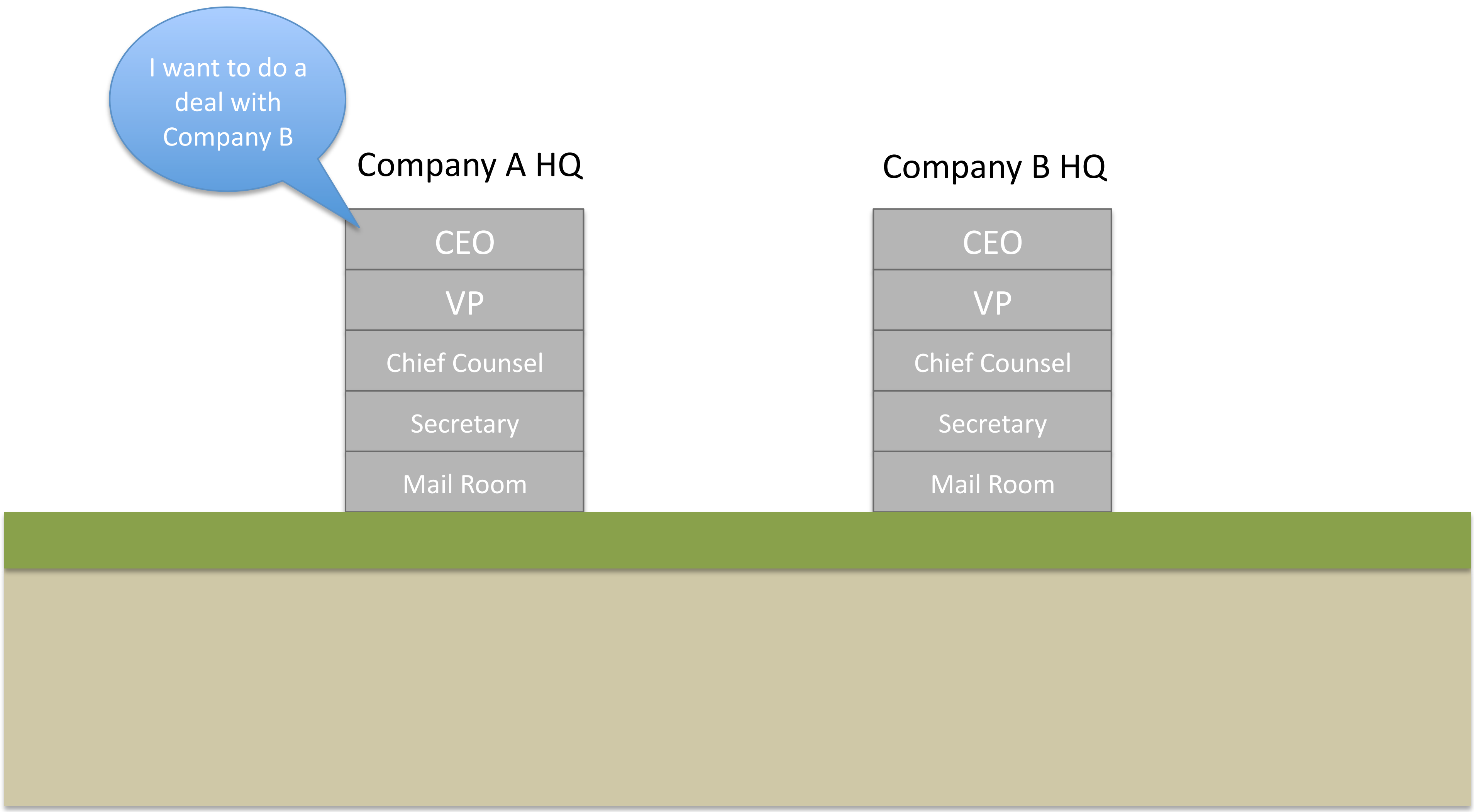
Data/Link Layer: Ethernet, Wifi, GSM…
- Sends messages over a single hop

Physical Layer: Copper, Fiber, Wireless…
- Transmits bits through space

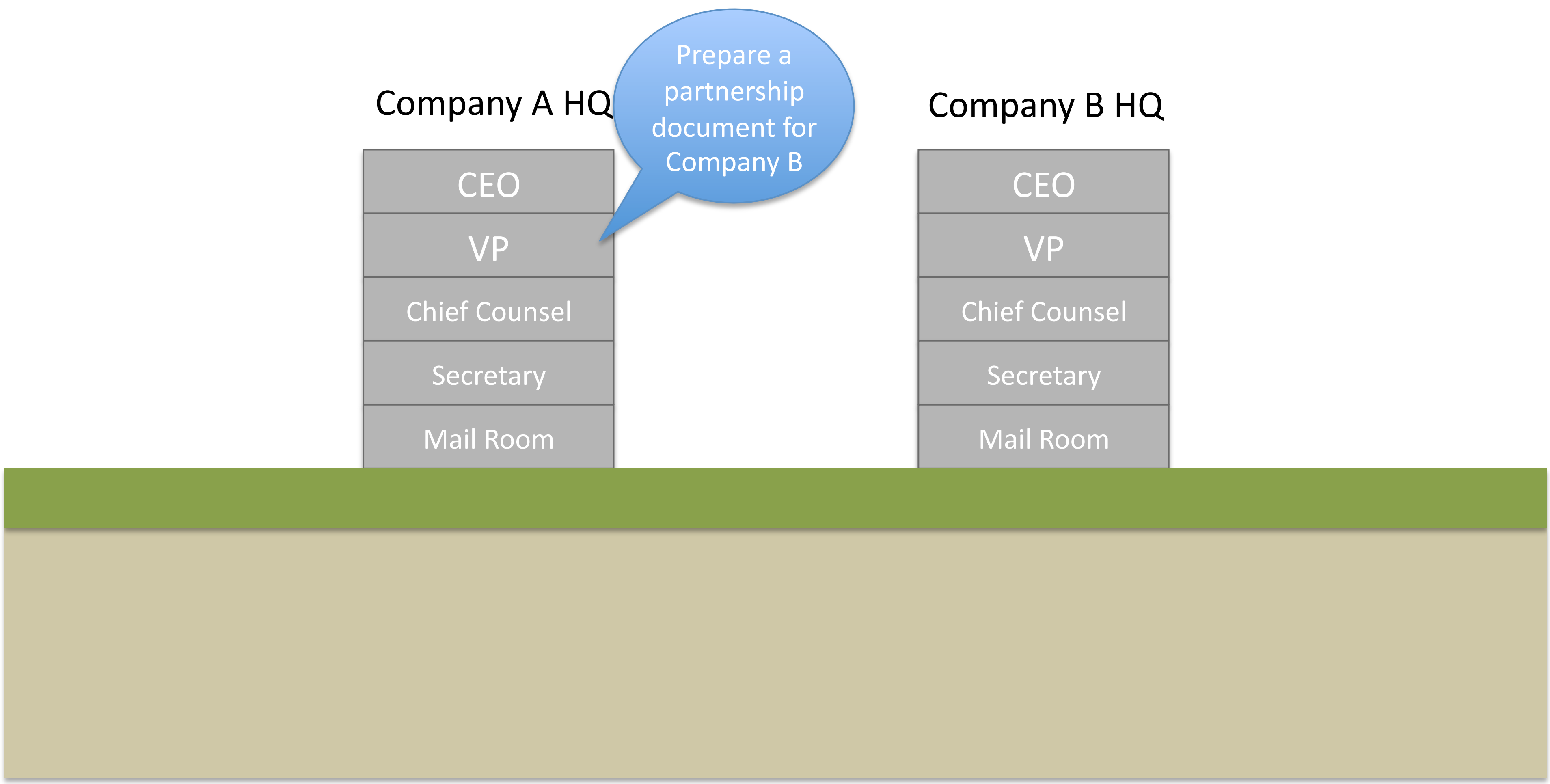
Corporate HQ Metaphor



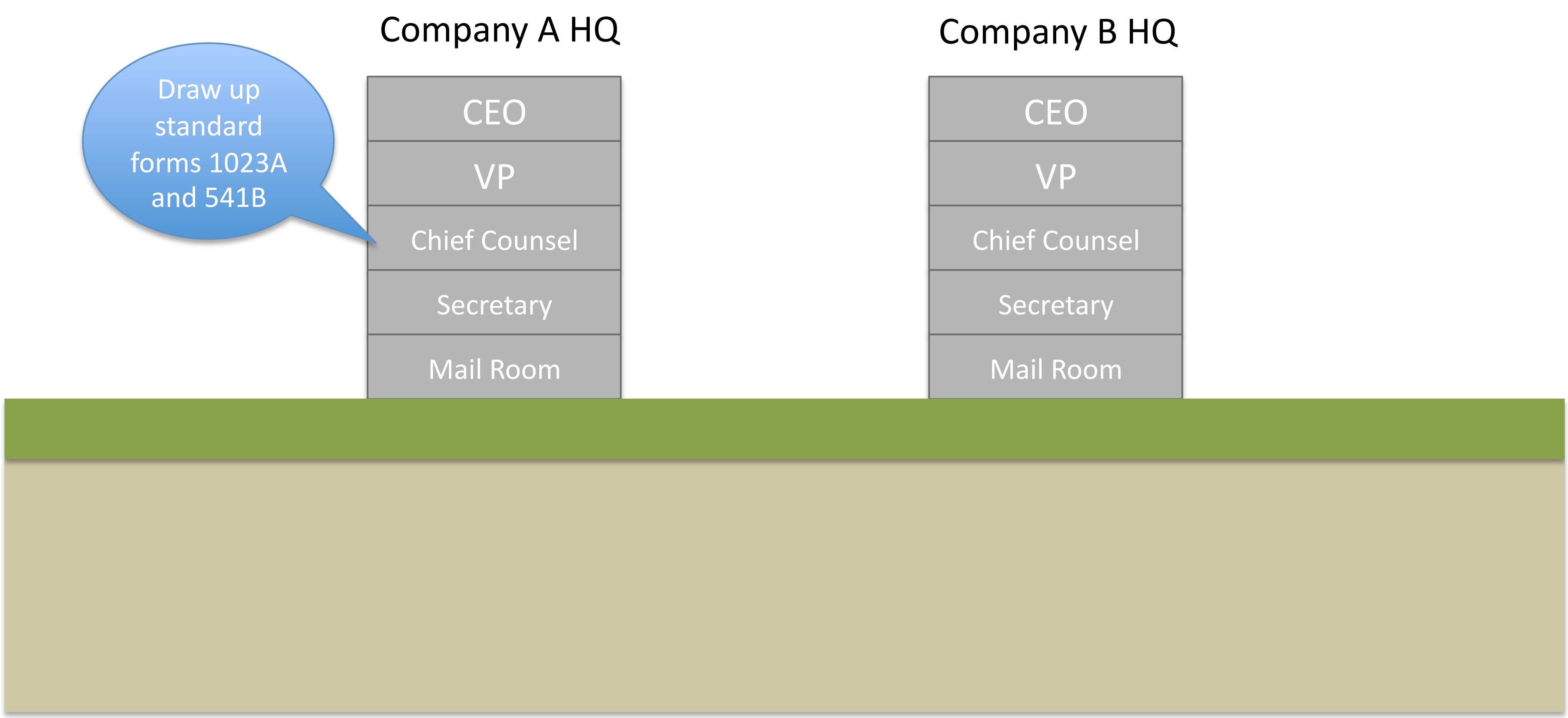
Corporate HQ Metaphor



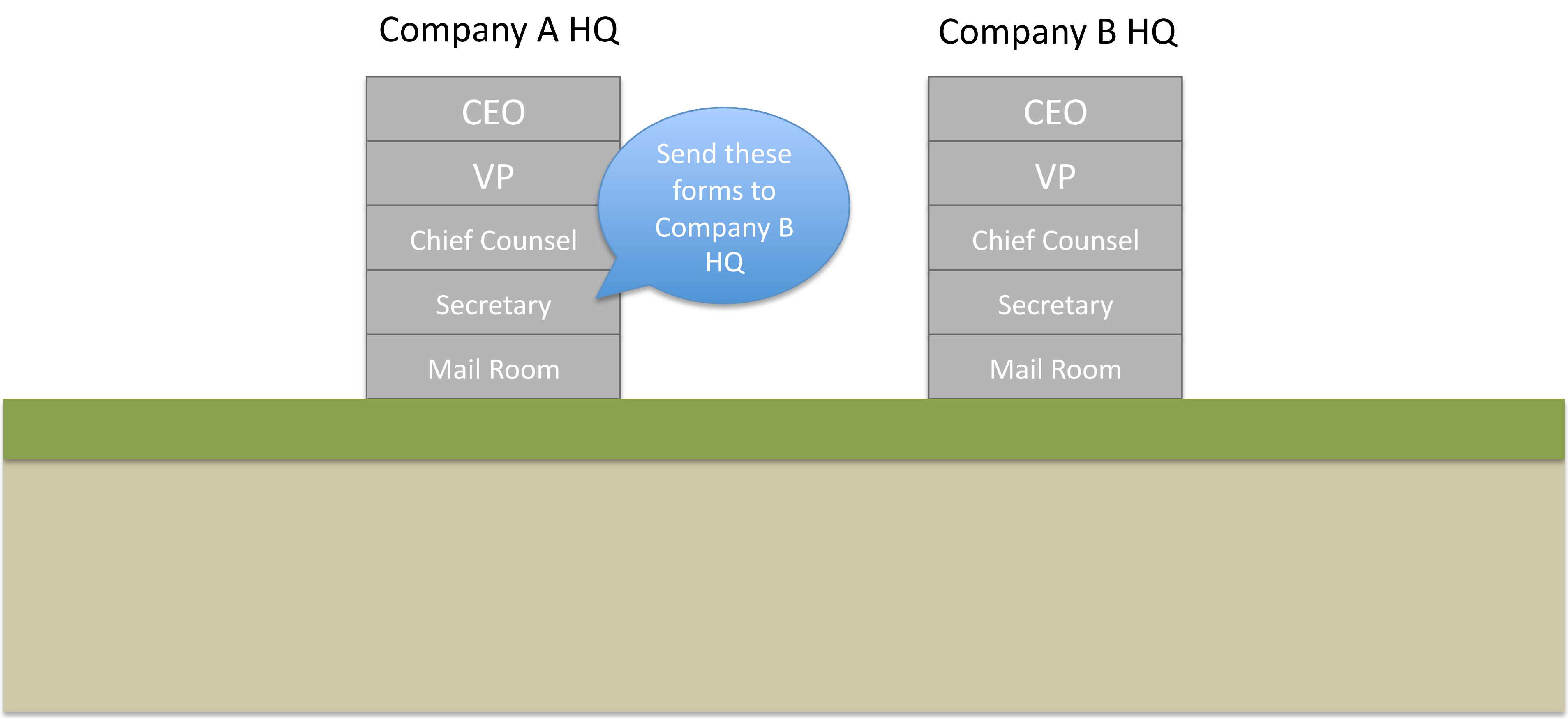
Corporate HQ Metaphor



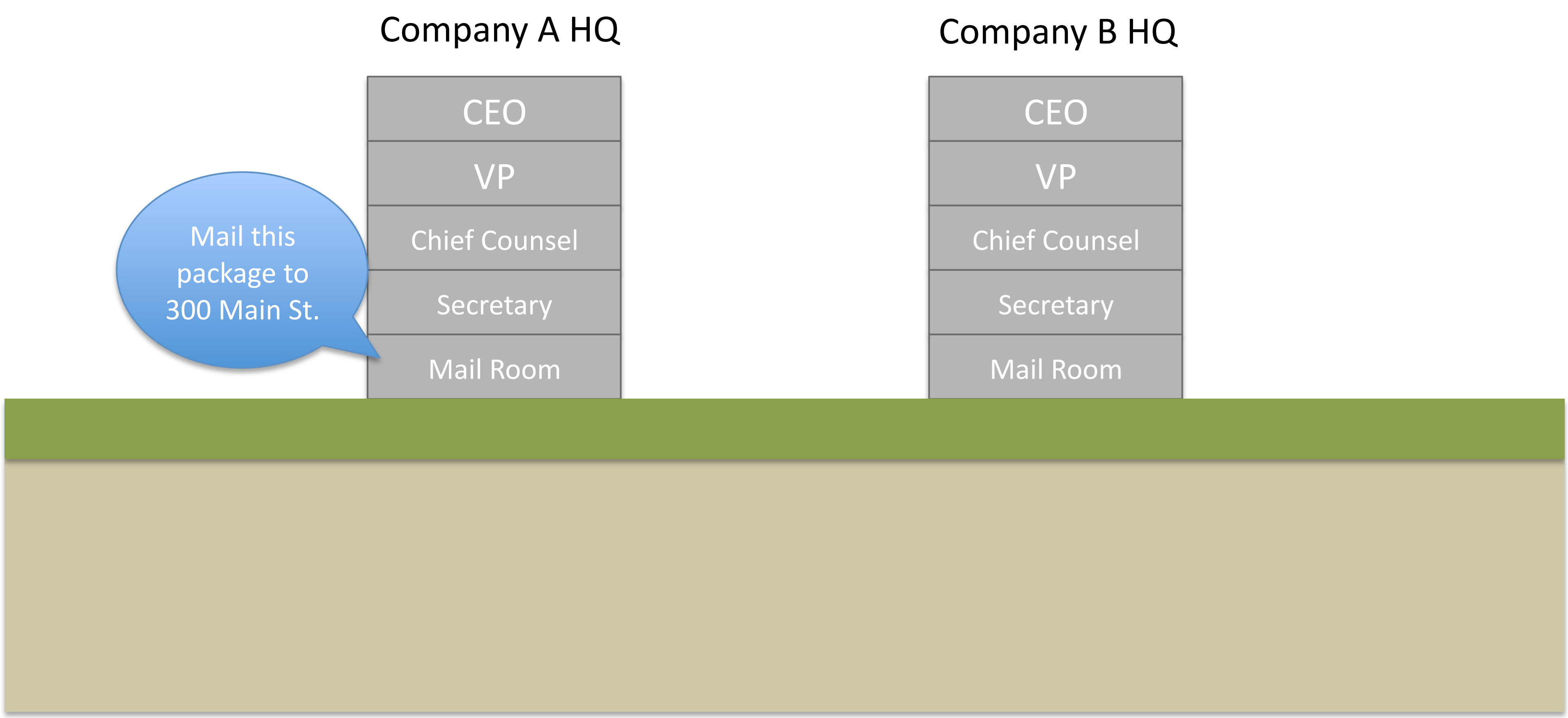
Corporate HQ Metaphor



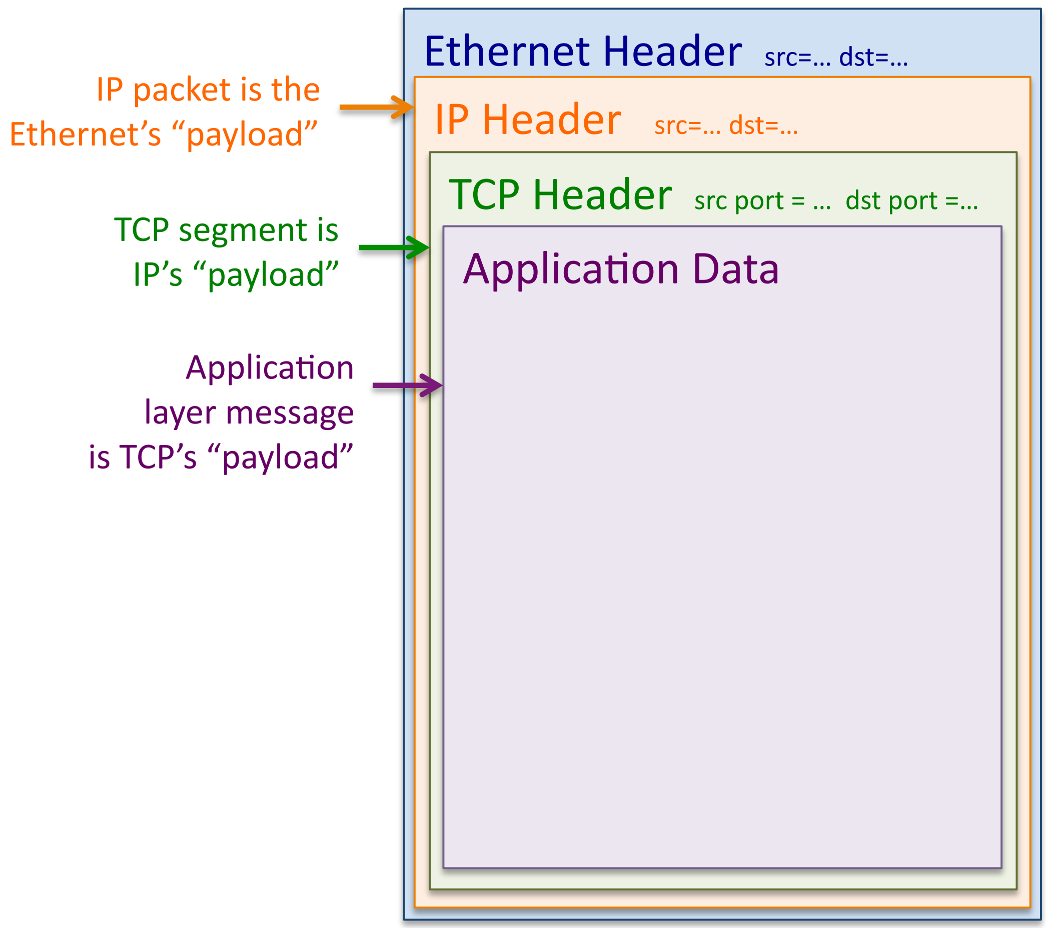
Corporate HQ Metaphor



Corporate HQ Metaphor



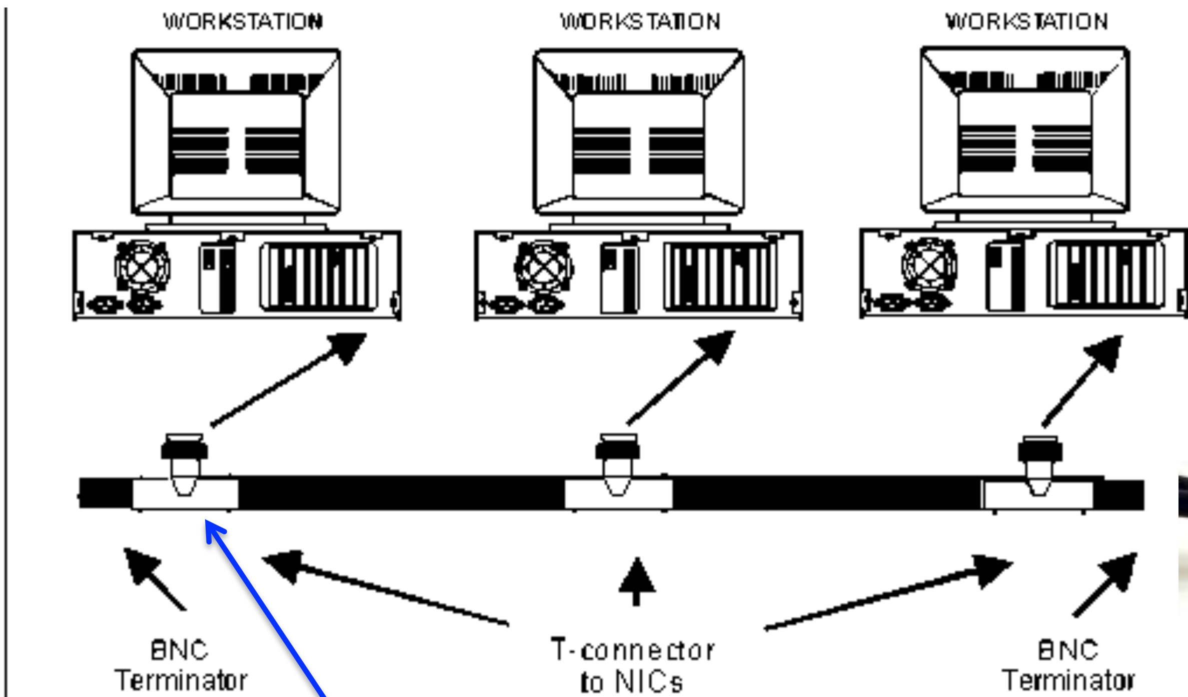
Message Encapsulation



Ethernet

- LAN – **Local** area network
  - Not for wide-area use
  - Assumes a common physical layer for all nodes
- Shared broadcast medium
  - Remind you of anything?
- Addresses are 48 bits (6 bytes)
  - Typically written as 6 pairs of hex characters
  - Ex: aa:bb:cc:00:11:22

Ethernet History: 10Base2

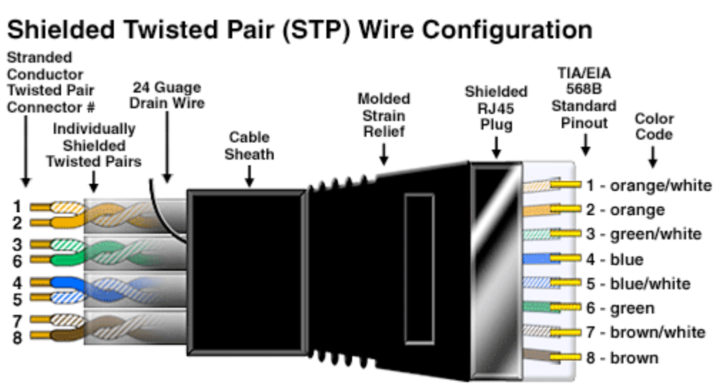


Coaxial wiring
-- Just like cable TV

Shared broadcast medium (literally, a copper wire)
-- Every computer sees every message!

Modern Ethernet

- 10BaseT
  - Like 10Base2, but over twisted-pair copper wire
- 100BaseTX, 1000BaseT
  - Faster versions – 100 mbps, 1 gbps
  - Often backwards compatible with 10BaseT
  - Still very similar at the link layer



IP: Internet Protocol

- Connectionless, “best effort” service
  - Kind of like the post office, or SMS text messaging
  - Each packet may or may not be delivered all the way to its destination
  - Packets may not arrive in the same order they were sent

Internet Protocol Addressing

- What's an IP address? What's a netmask?
  - Ex: 192.168.0.1 / 255.255.255.0
- In IPv4, addresses are 32-bit unsigned integers
  - Can be written in "dotted quad" notation as 4 numbers between 0 and 255 (inclusive)
  - High-order bits denote the network address
  - Low-order bits are the host address within that network

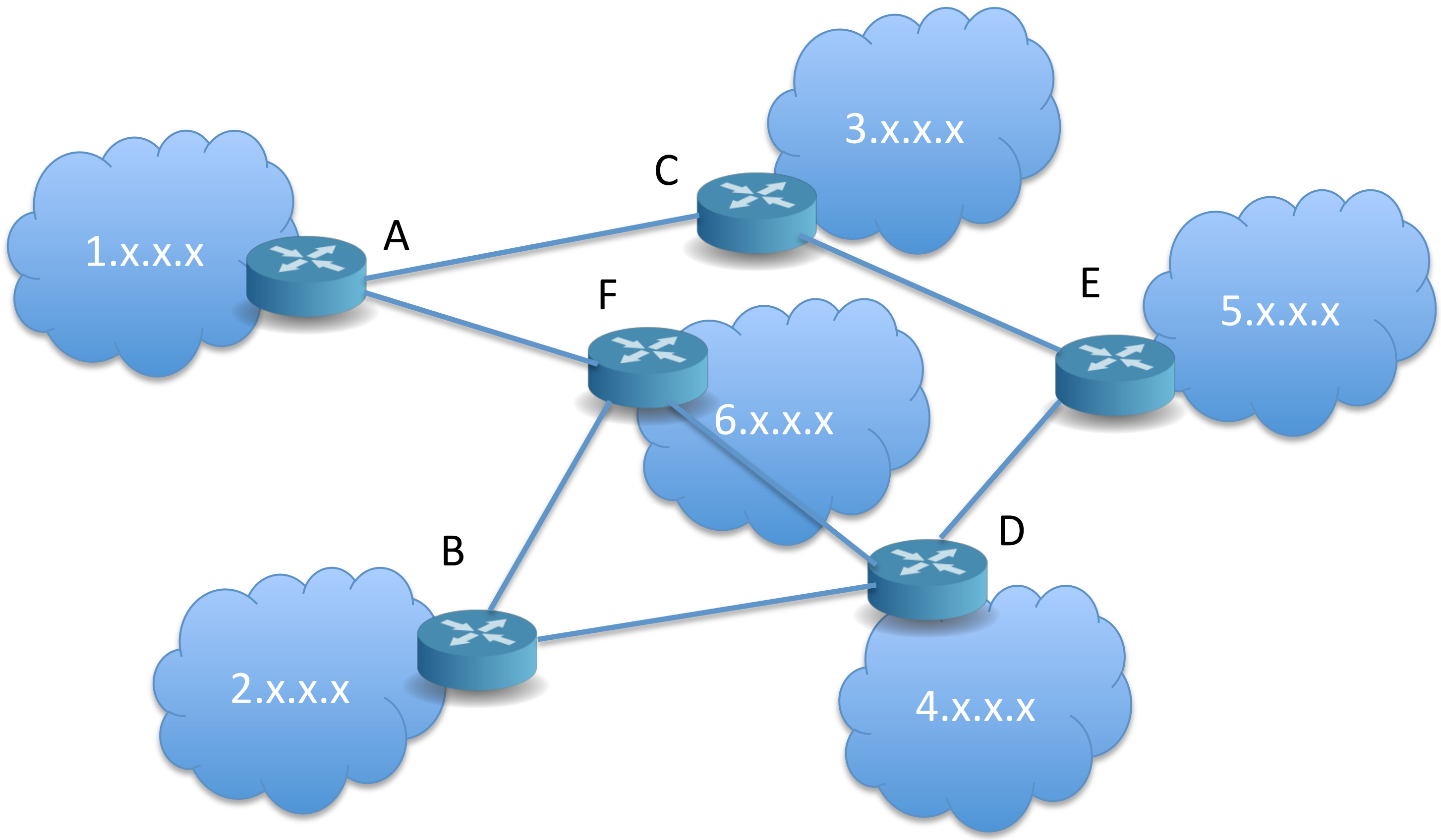
Internet Protocol Addressing

- The netmask tells which bits of the IP address are for the network and which are for the host
  - Example above: netmask = 255.255.255.0
    - Network address = 192.168.0.*
    - Host address = 1
  - Hosts with the same network address are on the same LAN
- Alternative notation: prefix length
  - Ex 192.168.0.1 / 24

IP Routing

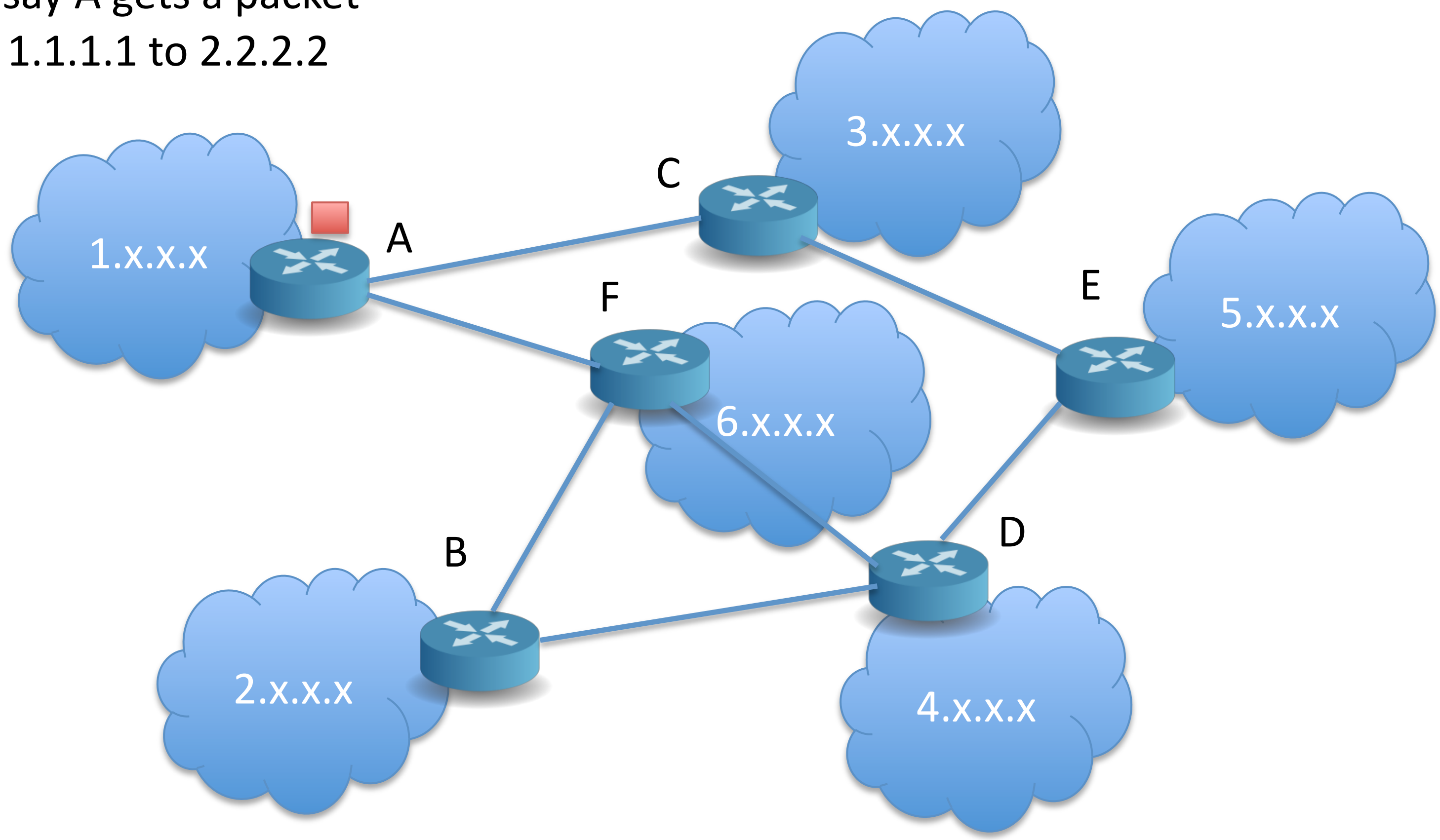
- Routers operate on individual IP packets
  - Examine each packet in isolation
  - Decide quickly who should get it next
  - Forward it on to the next "hop"
- The *routing table* stores next hop info for various destinations
  - Created either statically (by hand) or by execution of a routing protocol with other routers

IP Routing



IP Routing

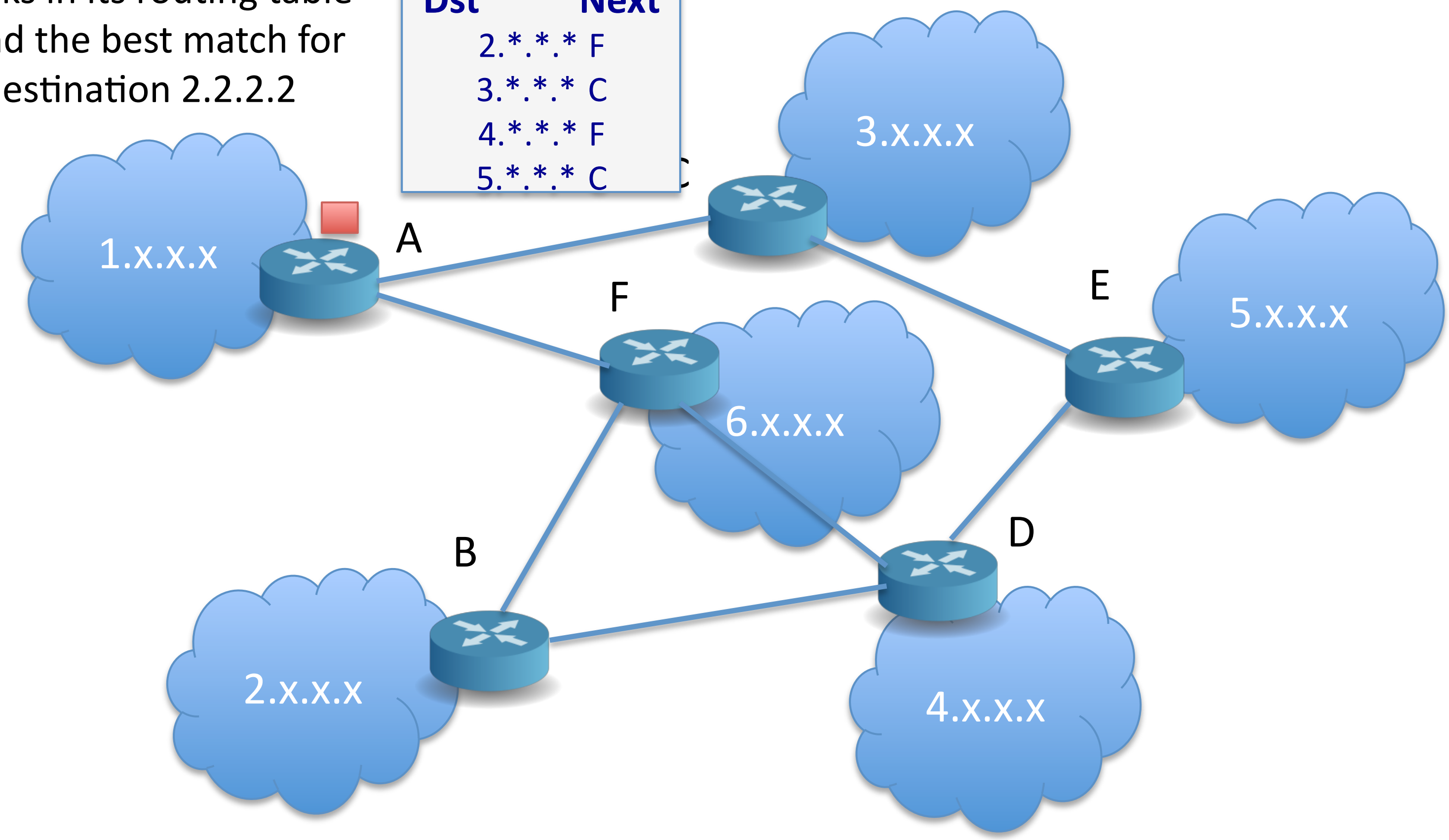
Let's say A gets a packet
from 1.1.1.1 to 2.2.2.2



IP Routing

A looks in its routing table to find the best match for the destination 2.2.2.2

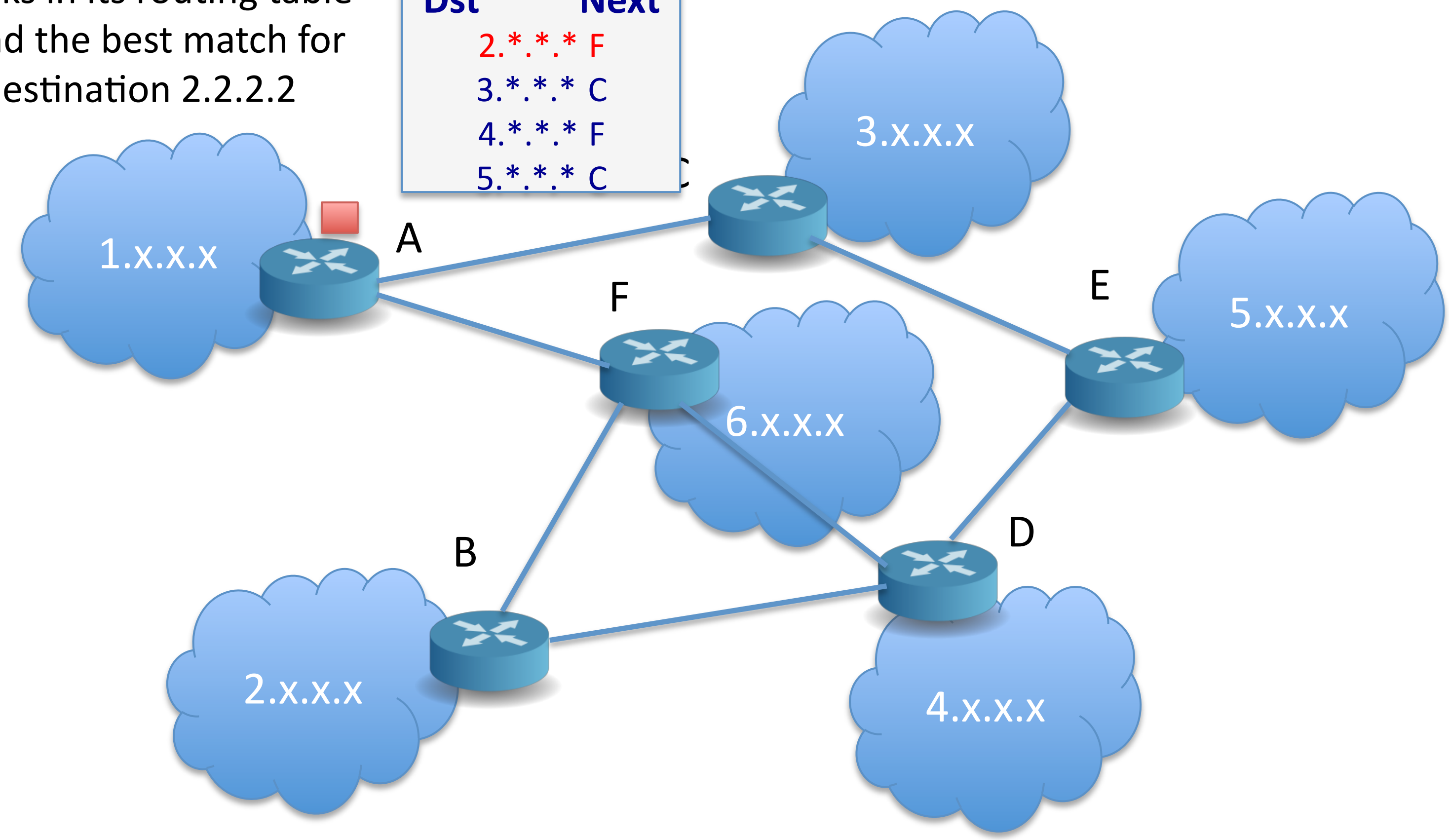
Dst	Next
2.*.*.*	F
3.*.*.*	C
4.*.*.*	F
5.*.*.*	C



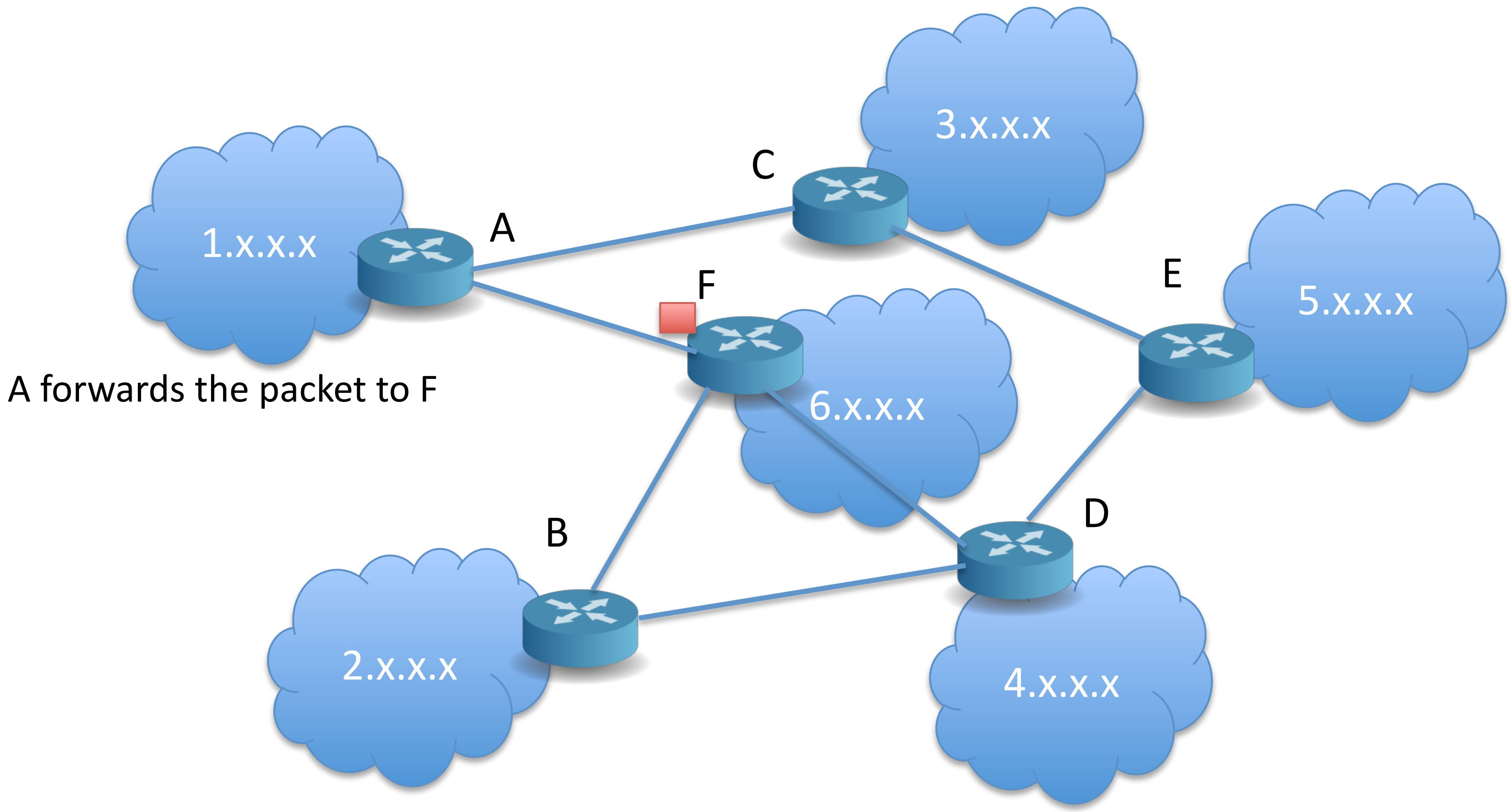
IP Routing

A looks in its routing table to find the best match for the destination 2.2.2.2

Dst	Next
2.*.*.*	F
3.*.*.*	C
4.*.*.*	F
5.*.*.*	C

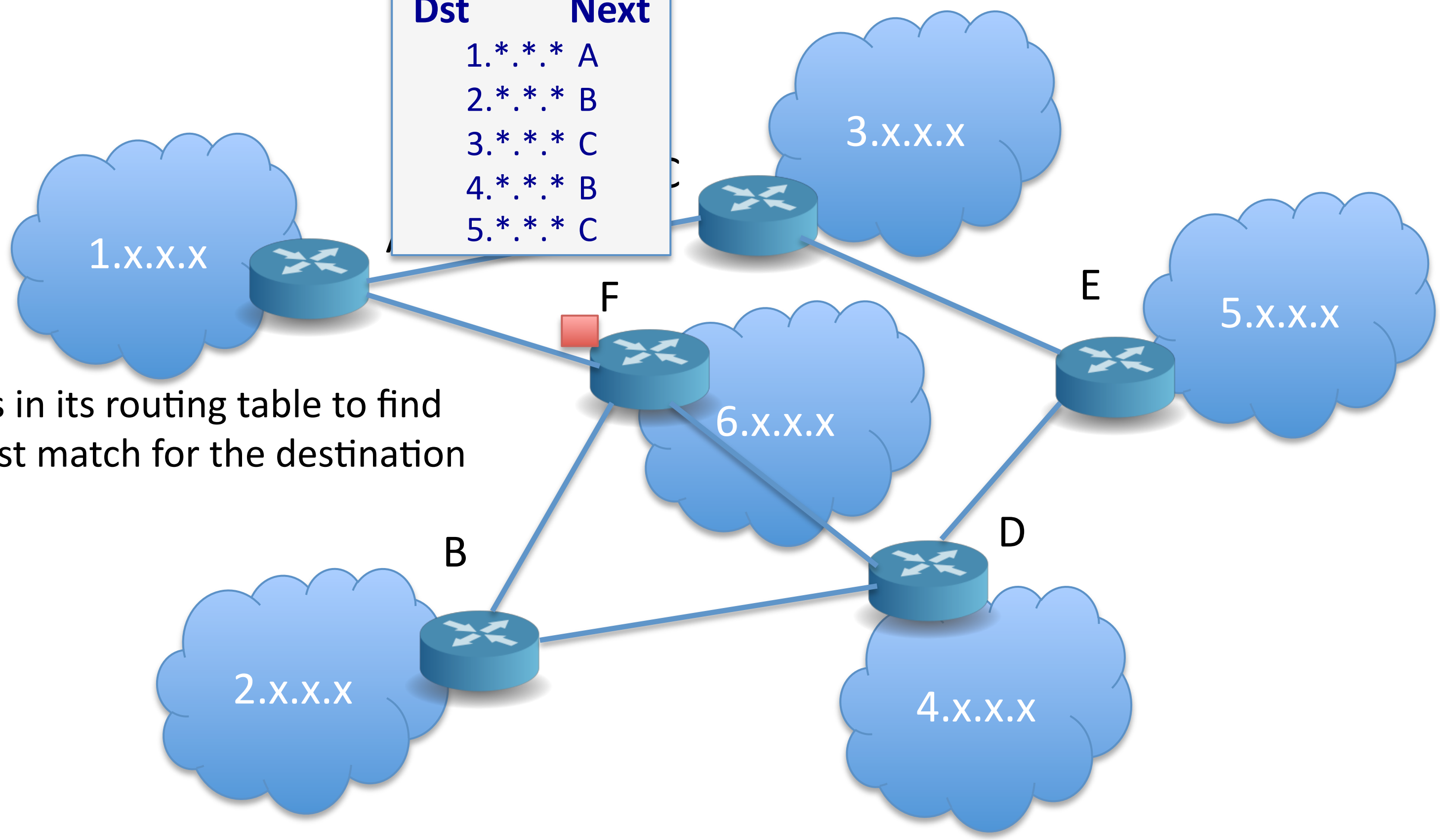


IP Routing


1.x.x.x
2.x.x.x
3.x.x.x
4.x.x.x
5.x.x.x
6.x.x.x
A
B
C
D
E
F
A forwards the packet to F

IP Routing

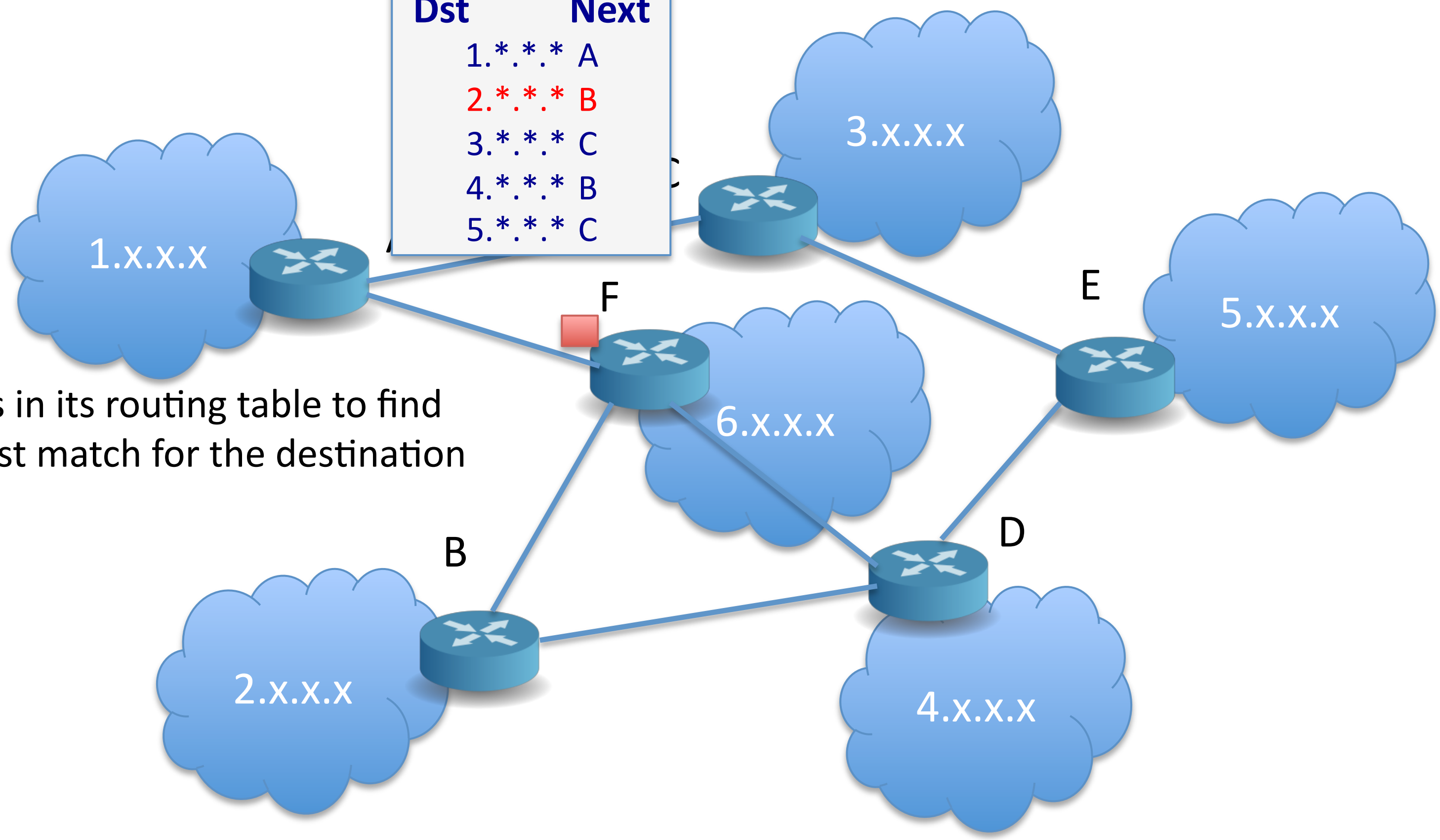
Dst	Next
1.*.*.*	A
2.*.*.*	B
3.*.*.*	C
4.*.*.*	B
5.*.*.*	C



F looks in its routing table to find the best match for the destination

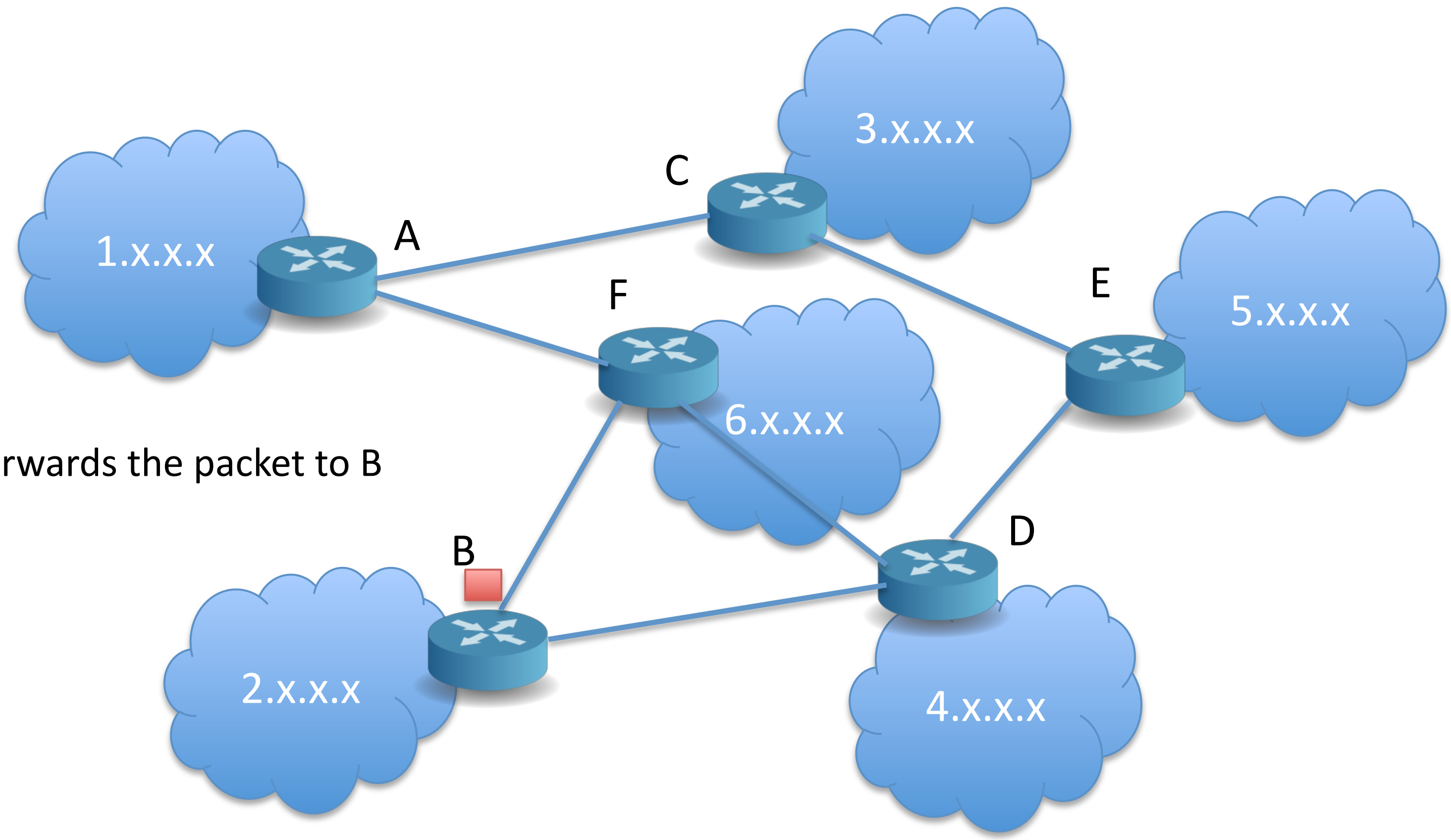
IP Routing

Dst	Next
1.*.*.*	A
2.*.*.*	B
3.*.*.*	C
4.*.*.*	B
5.*.*.*	C



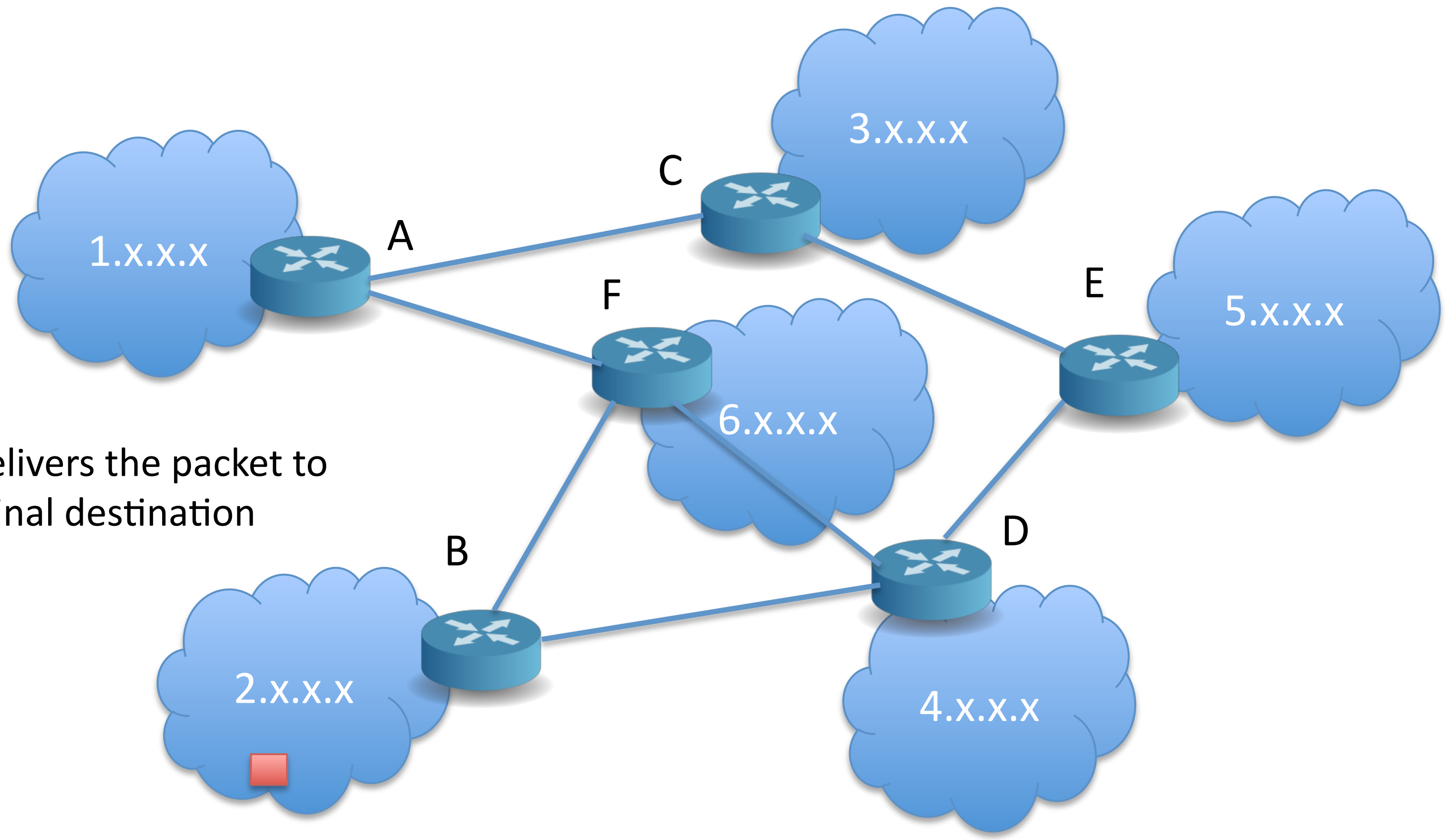
F looks in its routing table to find the best match for the destination

IP Routing



F forwards the packet to B

IP Routing



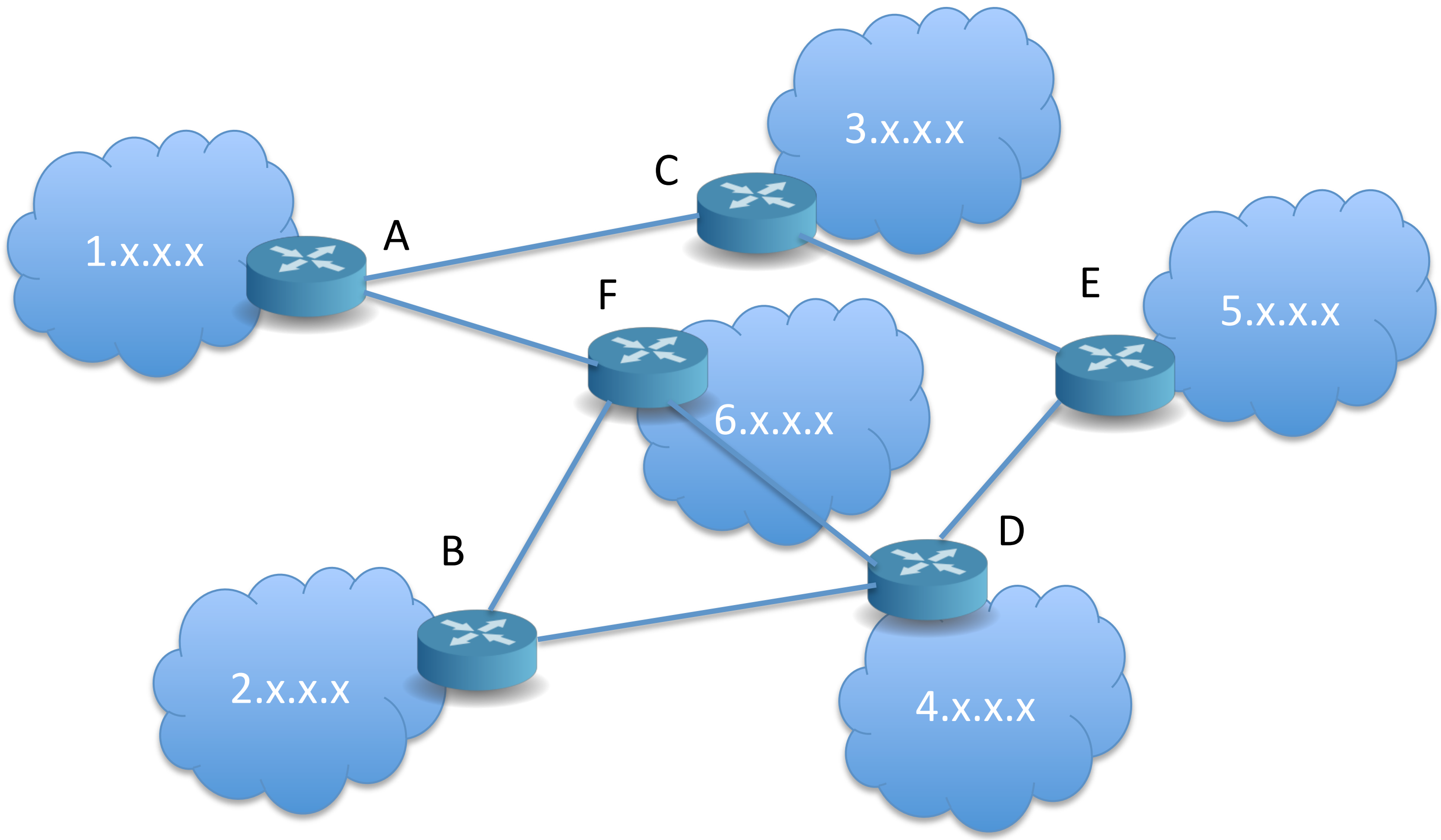
B delivers the packet to its final destination

BGP: Internet Routing

- BGP – Border Gateway Protocol
  - Used by big networks on the Internet to decide how they will route traffic
  - Each network sets up its own routing tables independently, but uses information provided by others

BGP Example

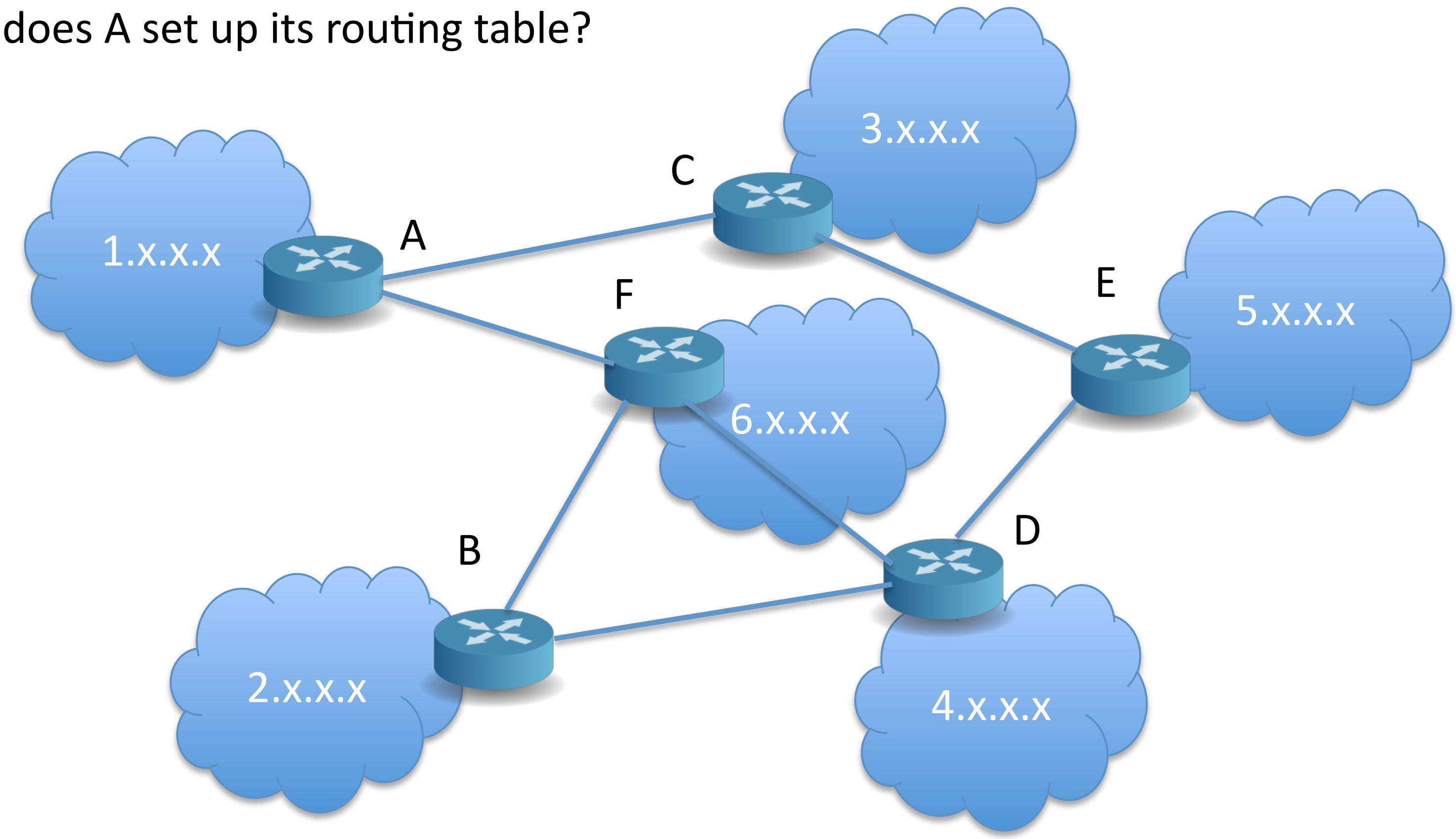
(greatly simplified)



BGP Example

(greatly simplified)

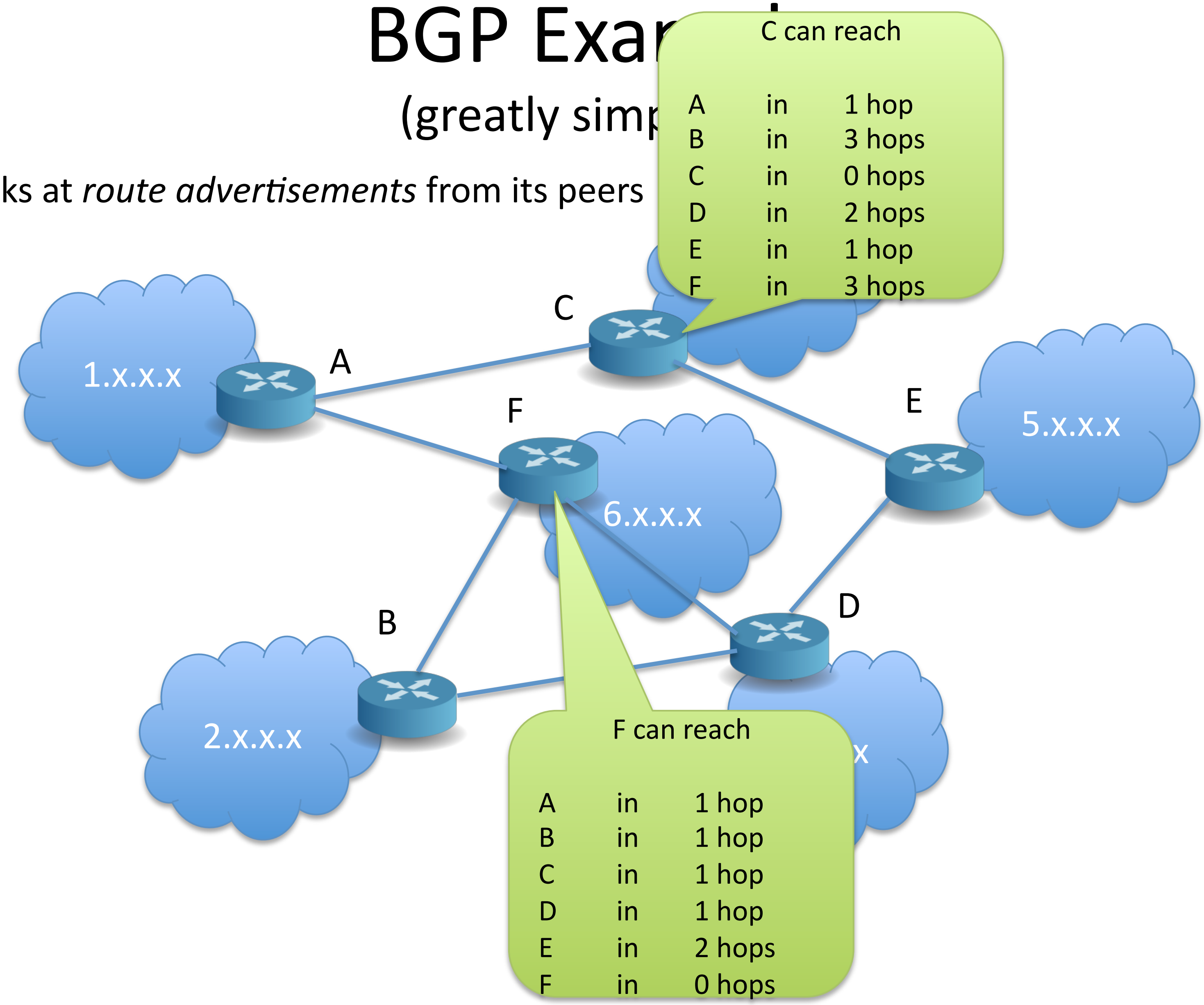
How does A set up its routing table?



BGP Exar

(greatly simp

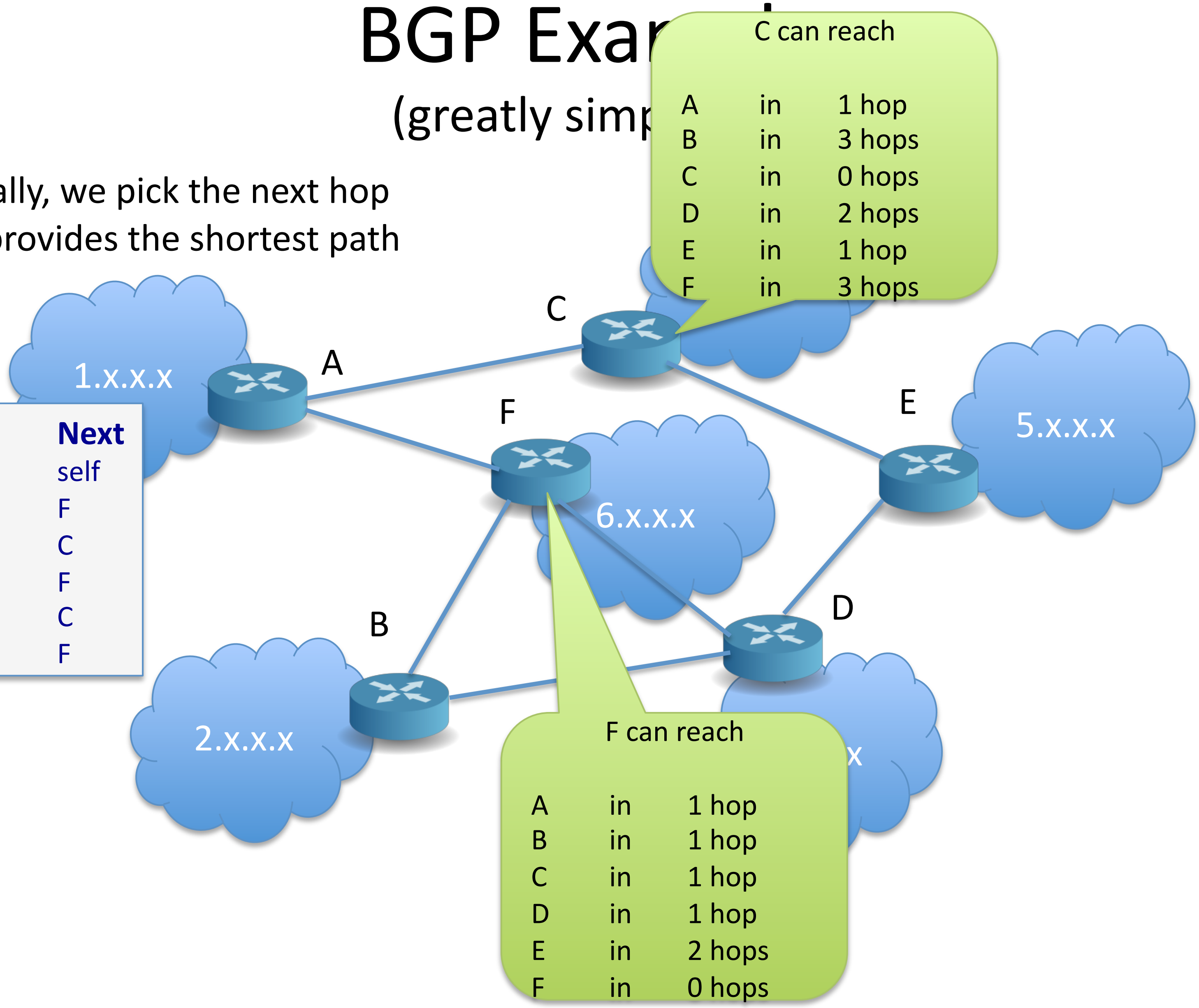
It looks at *route advertisements* from its peers



BGP Exar…

(greatly simp…

Typically, we pick the next hop that provides the shortest path



Dst	**Next**
1.*.*.*	self
2.*.*.*	F
3.*.*.*	C
4.*.*.*	F
5.*.*.*	C
6.*.*.*	F

How does a packet actually traverse the network?

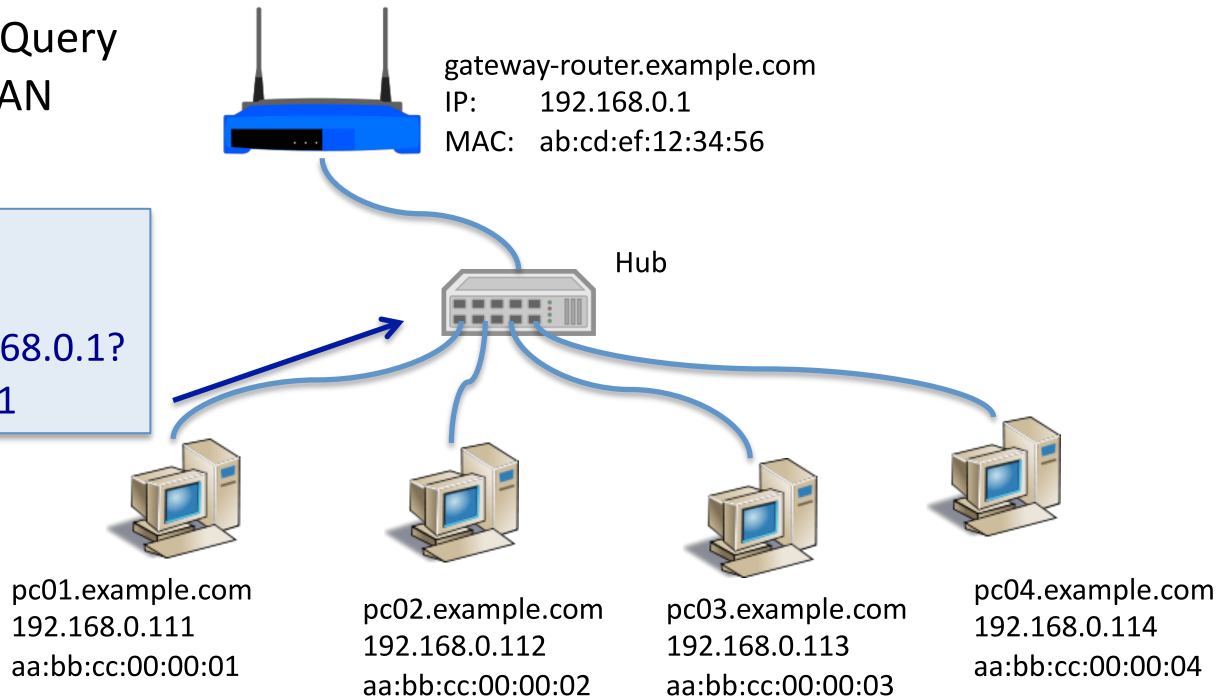
- Ethernet gives us a local area network
- IP gives us a worldwide network of networks

- How do we connect the two?
- How does a packet make it out of the LAN, and into the greater Internet beyond?

ARP: Address Resolution Protocol

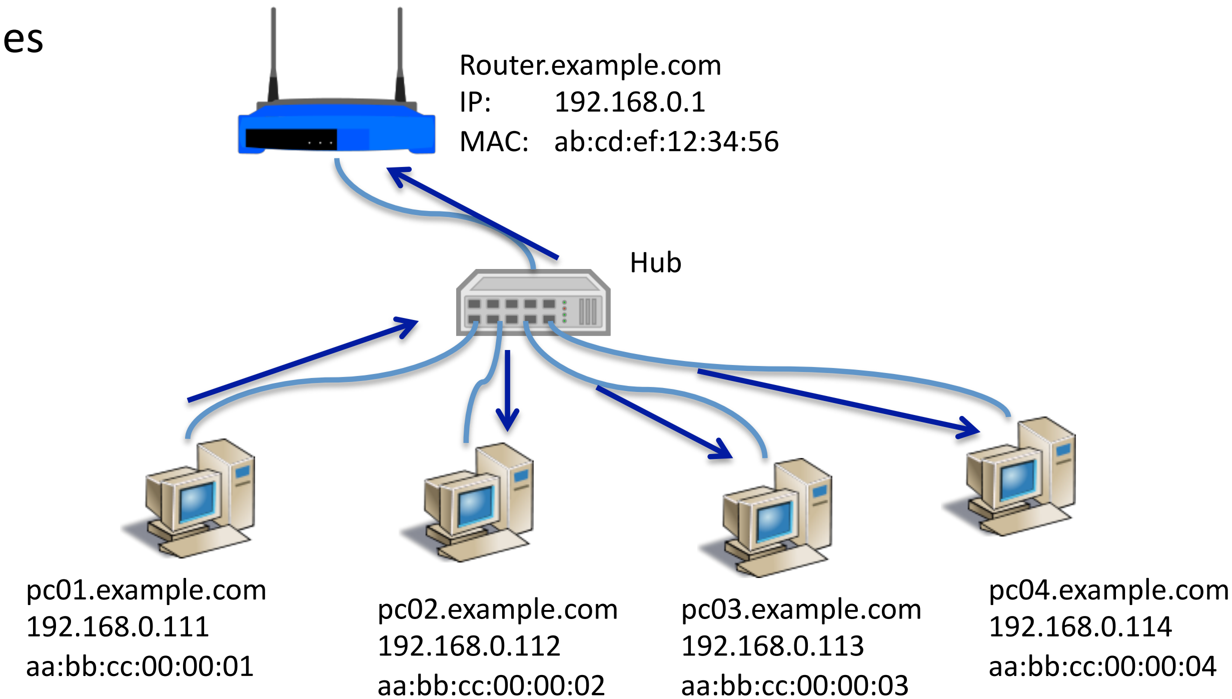
1. Initiator sends ARP Query over the Ethernet LAN

To: *:*:*:*:*:*
From: aa:bb:c:00:01
ARP – Who has 192.168.0.1?
Tell: aa:bb:cc:00:00:01



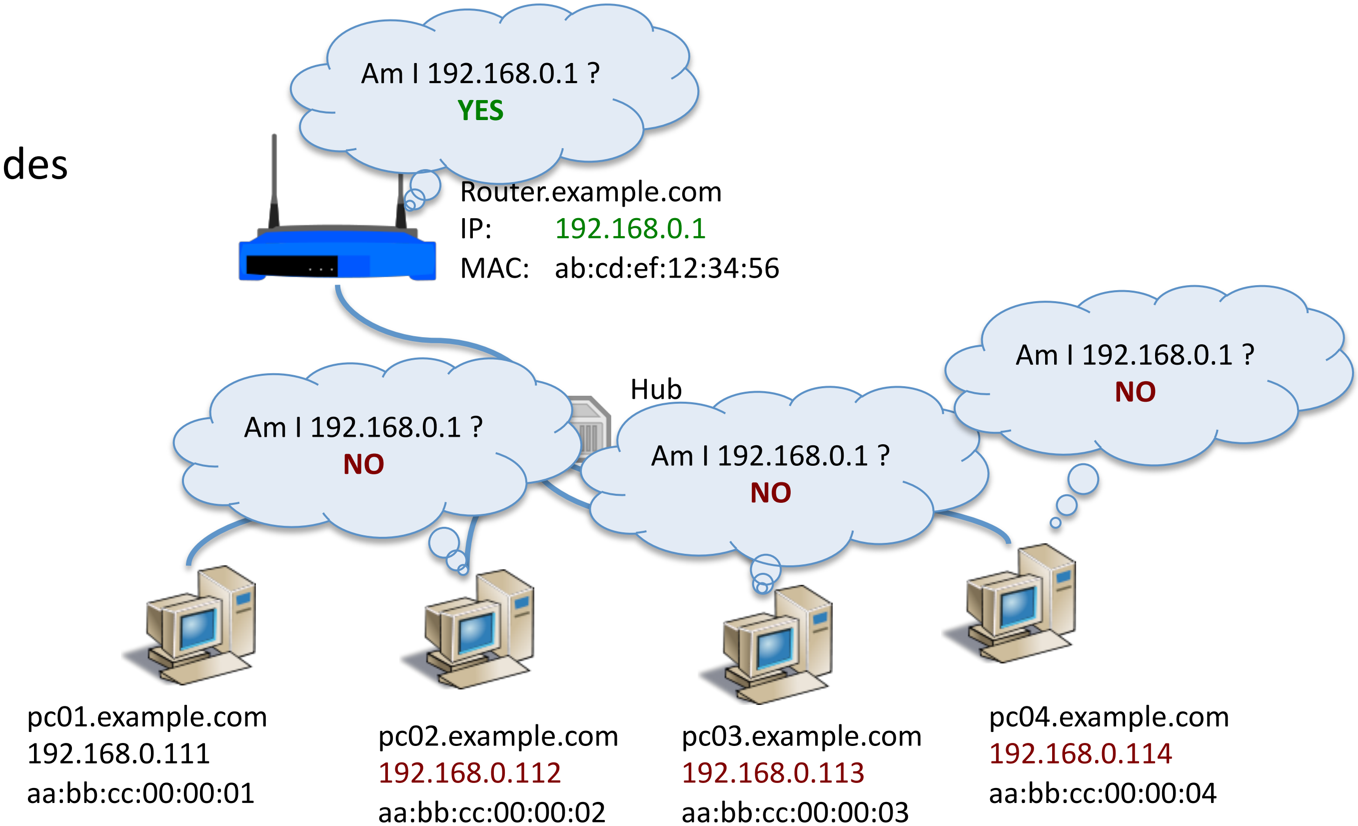
ARP: Address Resolution Protocol

2. Ethernet hubs copies the message to all connected devices



ARP: Address Resolution Protocol

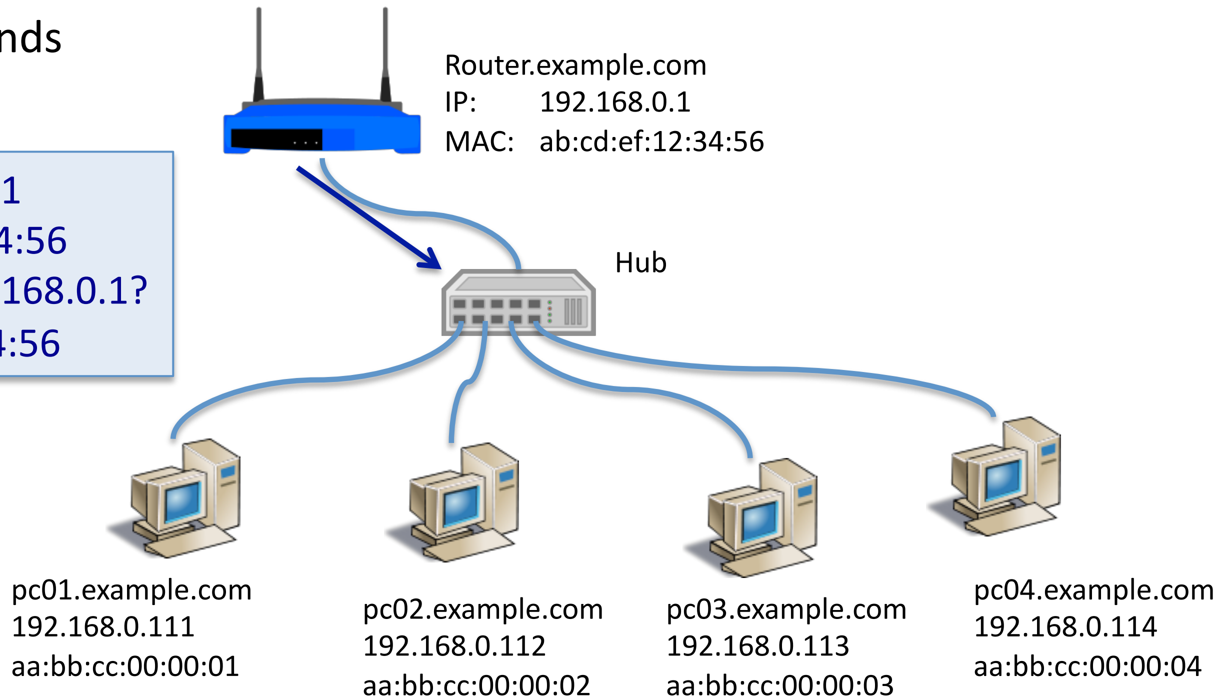
3. Each recipient decides whether or not to respond



ARP: Address Resolution Protocol

4. Matching device sends ARP Response

To: aa:bb:cc:00:00:01
From: ab:cd:ef:12:34:56
ARP – Who has 192.168.0.1?
MAC: ab:cd:ef:12:34:56

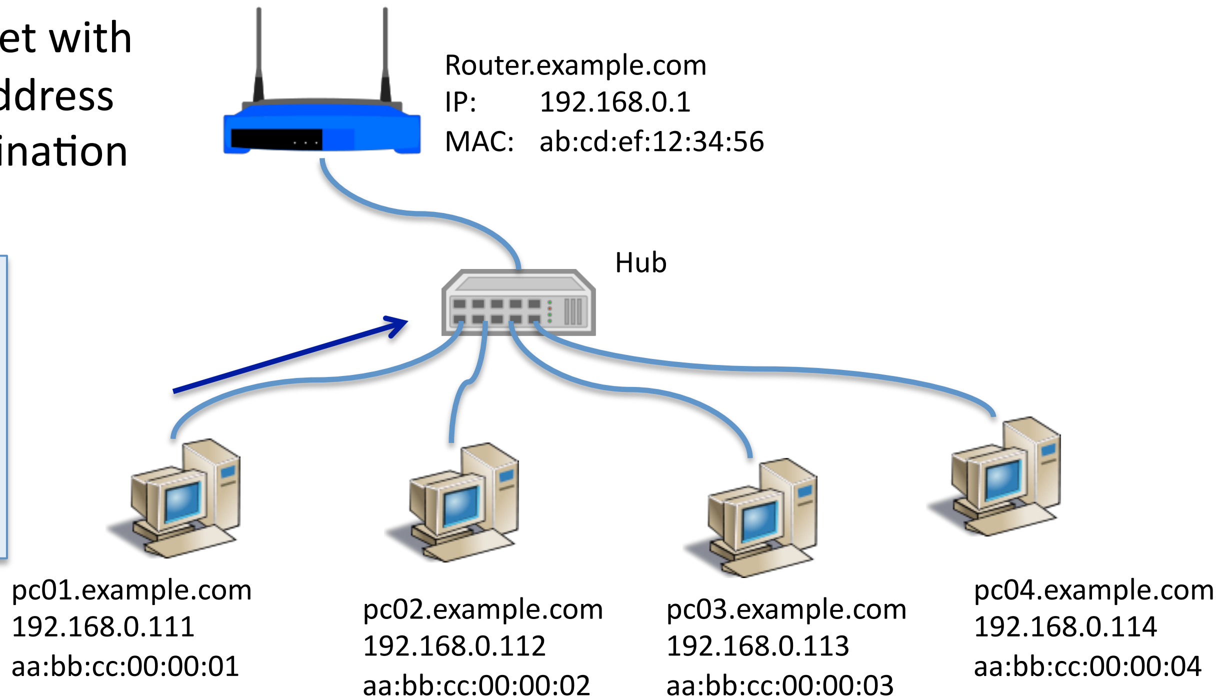


ARP: Address Resolution Protocol

5. PC sends its IP packet with the router's MAC address as its Ethernet destination

To: ab:cd:ef:12:34:56
From: aa:bb:cc:00:00:01

To: 10.1.2.3
From: 192.168.0.111
…



Checkpoint

- We’ve seen how Ethernet works for LANs
- We’ve seen how IP works for the Internet
- We’ve seen how computers on a LAN can send packets out to the Internet

- This is a good time for questions
  – Next, we’ll talk about what we can **do** with these networks

Network Protocol “Stacks”

Application Layer: HTTP, SMTP, …
- Send messages to people, send files, …

Transport Layer: TCP
- Stream-oriented I/O

Network Layer: IP
- Sends messages across multiple hops

Data/Link Layer: Ethernet, Wifi, GSM…
- Sends messages over a single hop

Physical Layer: Copper, Fiber, Wireless…
- Transmits bits through space

UDP: User Datagram Protocol

- UDP provides a thin layer on top of IP for connectionless protocols
  - An alternative to TCP, which we'll talk about soon

- Ports
  - Like different suite numbers in the office building
  - Enables many different services to run over UDP on the same machine, at the same IP address
  - Each port has a 16-bit identifier
  - 4-tuple of (src IP, dst IP, src port, dst port) identifies packets used to communicate between two programs

DNS: Domain Name System

- DNS provides a mapping from human-readable hostnames to numeric IP addresses

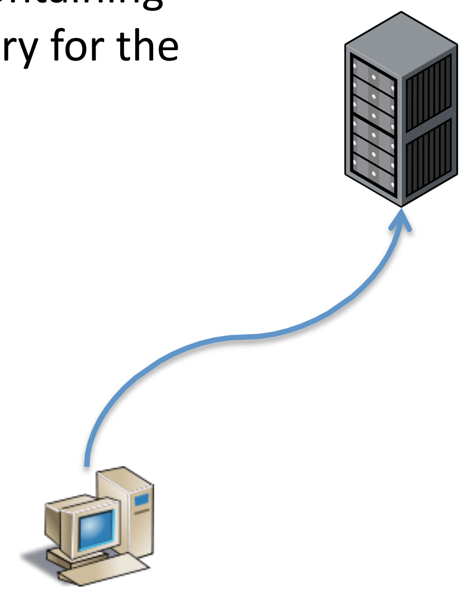
- Example:
  - I type in www.cs.pdx.edu in my browser
  - My laptop sends IP packets to 131.252.208.58

DNS Message Format

- Header section
  - Query ID – uniquely identifies this query
- Question section
  - Lists domain names to resolve
- Answer section
  - Gives answers to queries
- Authority section
  - Lists the DNS servers responsible for domains
- Additional section
  - Contains other records that might be of interest

DNS Query

Client sends UDP packet containing a DNS message with a Query for the desired DNS name



DNS Server

Client

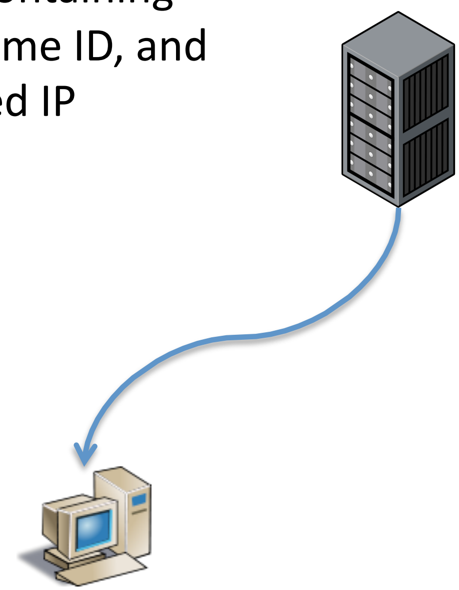
To: DNS Server IP
From: Client IP

Src Port: Client Port
Dst Port: 53 (DNS)

DNS Message ID = 12345
Query
Address for www.cs.pdx.edu

DNS Response

Server sends UDP packet containing
a DNS message with the same ID, and
an answer for the requested IP



To: Client IP
From: DNS Server IP

Src Port: 53 (DNS)
Dst Port: Client Port

DNS Message ID = 12345
Answer
www.cs.pdx.edu is 1.2.3.4

How does the server know the answer?
It probably queries other DNS servers,
using the same protocol!

TCP: Transmission Control Protocol

- Adds reliable, in-order data transfer to IP
  - Stream-oriented I/O
  - Acts like a file descriptor
- Also does congestion control and flow control

- Ports
  - Just like in UDP

TCP: 3-way Handshake

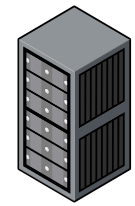
- Goal: establish a reliable connection over an unreliable packet network (IP)

- Need to make sure both sides know whether or not the connection was successfully set up

TCP: 3-way Handshake

Client

Server



`connect()`

Client creates new socket with 4-tuple ID (client IP, client port, server IP, server port)

`listen()`

Server waits for new connections

`SYN` →

Server creates new socket with 4-tuple ID (client IP, client port, server IP, server port)

Client knows connection is established

← **`SYN/ACK`**

`ACK` →

Server knows connection is established

Checkpoint

- Now we've talked about how computers resolve hostnames to IP addresses
- And how programs can get a file descriptor-like interface to talk over the network

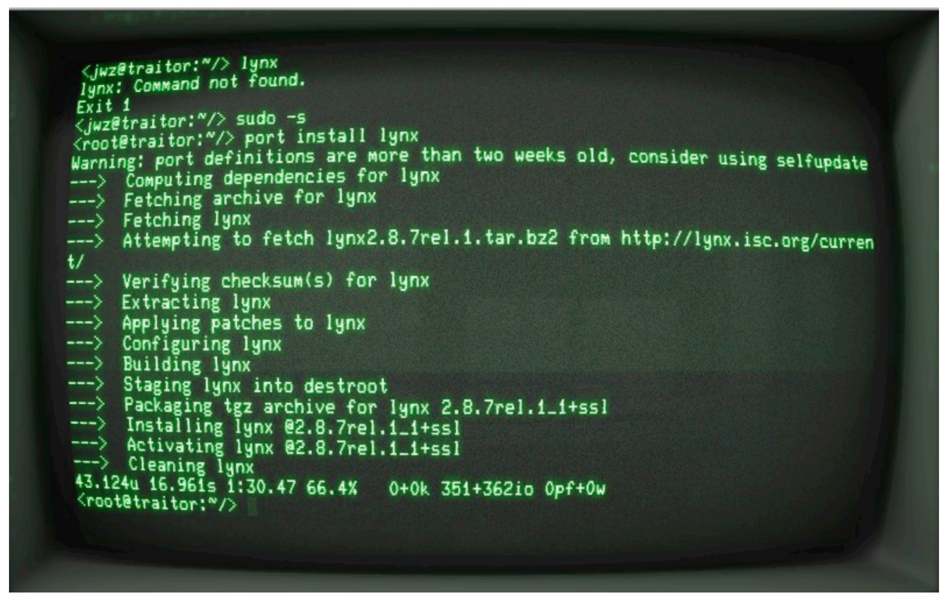
- Questions?

- Next, we talk about some actual applications

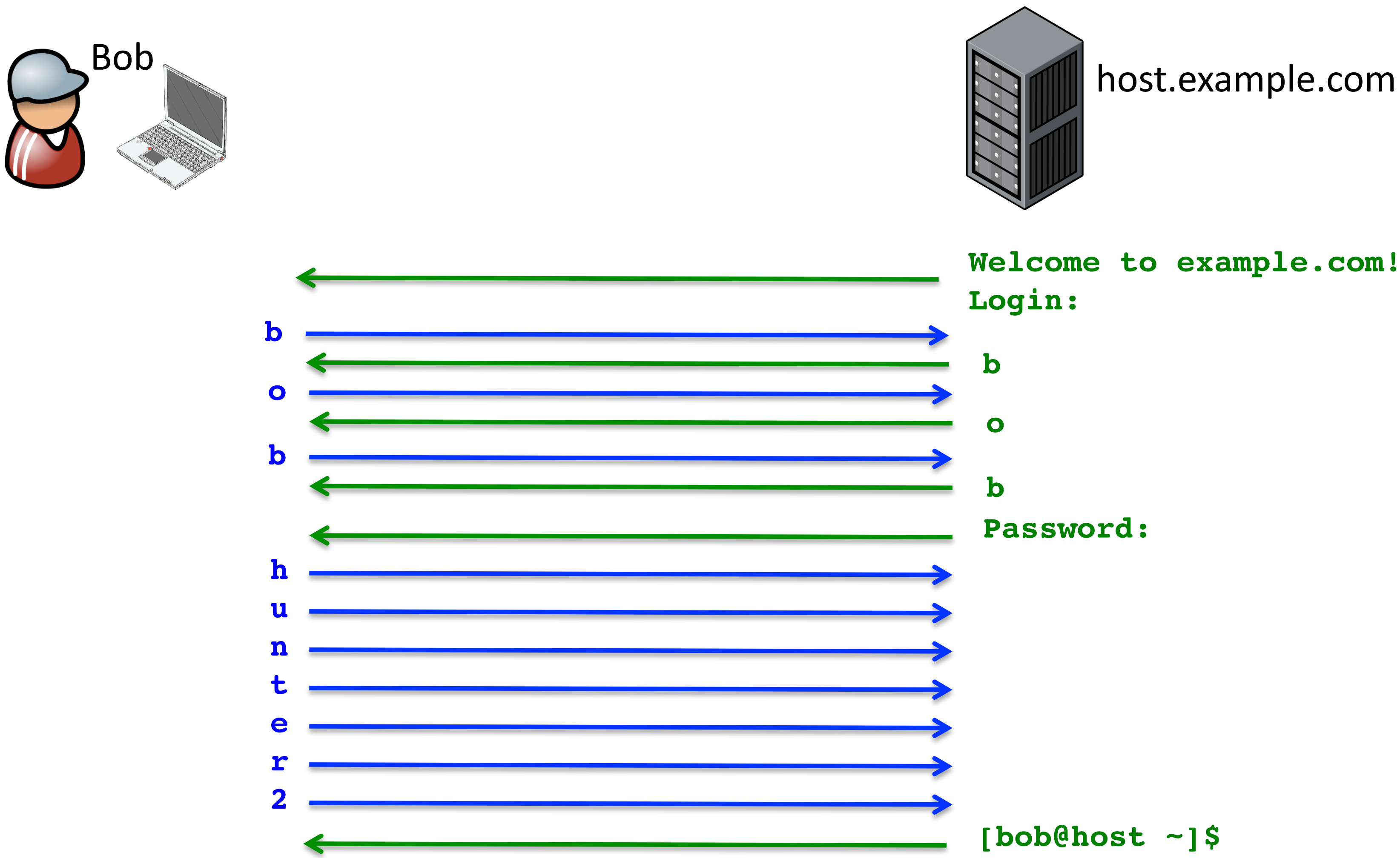
Telnet, rlogin, rsh: Remote Access

- Telnet
  - RFC 137 ← Really old!!! (1971)
  - Lets a remote user log in to a server as if he were sitting locally at a (text-only) terminal
  - For the most part, Telnet simply sends raw characters via TCP stream

DEC VT100 terminal, ca 1978



Telnet: Basic Example

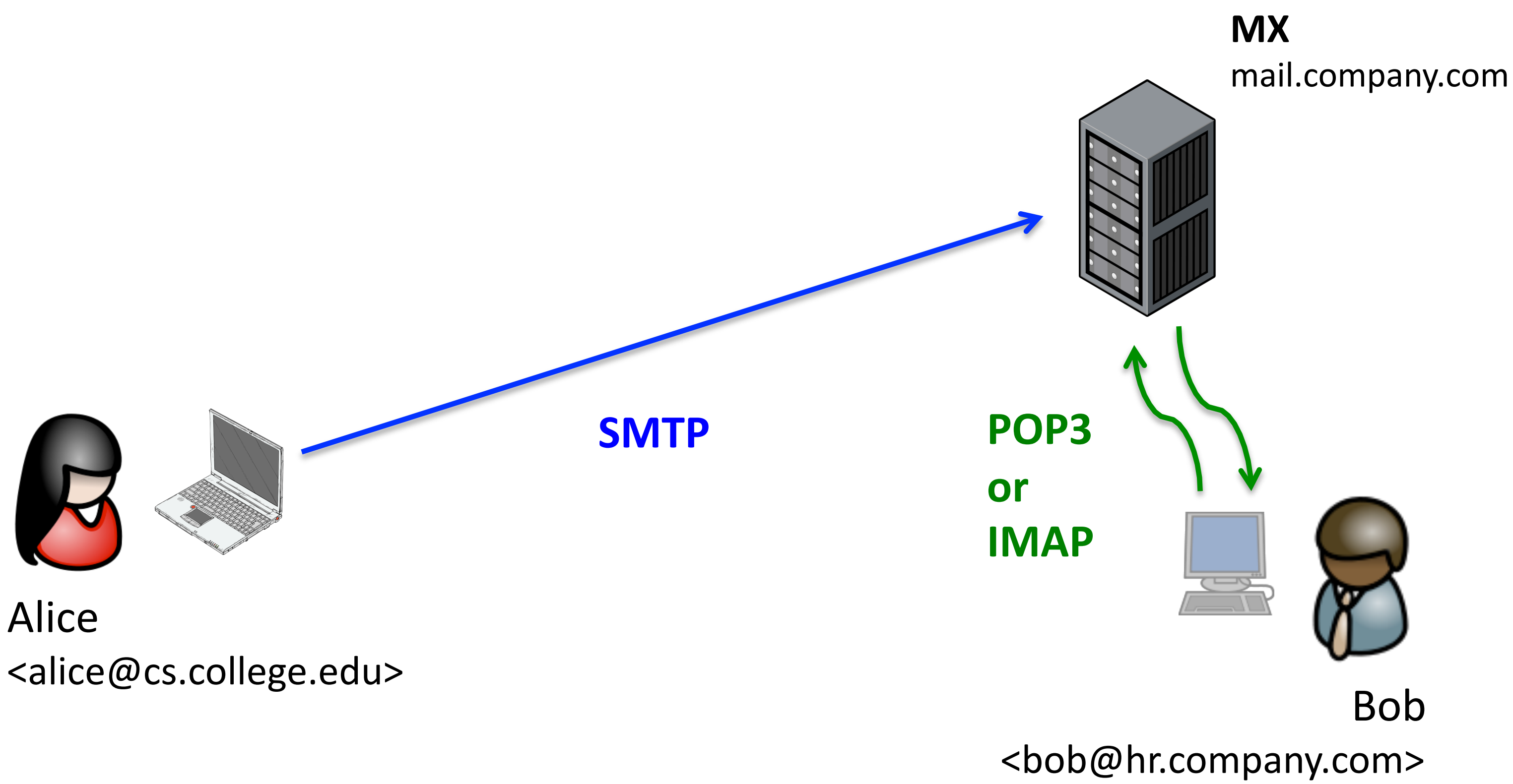


Email

- One protocol for sending mail
  - SMTP: Simple Mail Transfer Protocol
  - RFC 821 ← Really old!!! (1982)
  - RFC 2821, 5321

- Two distinct protocols for checking mail
  - POP3: Post Office Protocol version 3
  - IMAP: Internet Message Access Protocol

SMTP: Simple Mail Transfer Protocol



SMTP Example

S: 220 smtp.example.com ESMTP Postfix
C: HELO relay.example.org
S: 250 Hello relay.example.org, I am glad to meet you
C: MAIL FROM:<bob@example.org>
S: 250 Ok
C: RCPT TO:<alice@example.com>
S: 250 Ok
C: RCPT TO:<theboss@example.com>
S: 250 Ok
C: DATA
S: 354 End data with <CR><LF>.<CR><LF>
C: From: "Bob Example" <bob@example.org>
C: To: "Alice Example" <alice@example.com>
C: Cc: theboss@example.com
C: Date: Tue, 15 January 2008 16:02:43 -0500
C: Subject: Test message
C:
C: Hello Alice.
C: This is a test message with 5 header fields and 4 lines in the message body.
C: Your friend,
C: Bob
C: .
S: 250 Ok: queued as 12345
C: QUIT
S: 221 Bye {The server closes the connection}

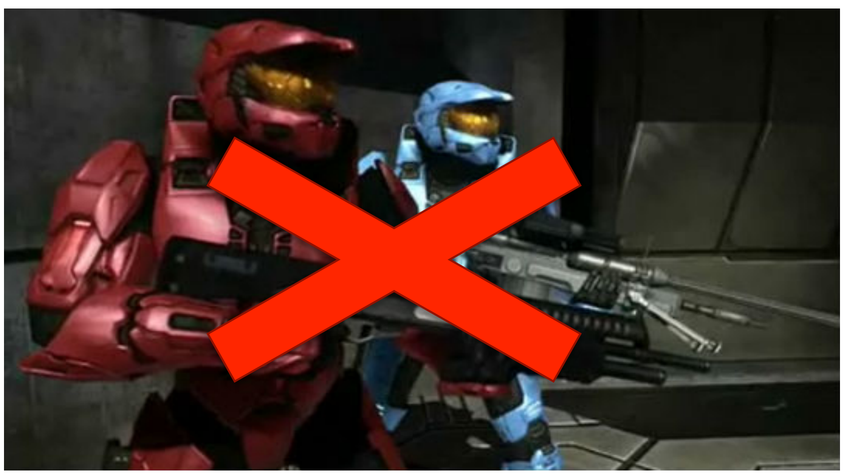
Everybody take 10…

BREAK

~~Announcements~~ Shameless Plug

- Interested in learning more about security?
- Have you heard about Capture the Flag?

Not this kind of CTF



This kind! Solving security challenges!

Portland State CTF Team

- I'm interested in organizing a team of students to compete regionally and nationally
- In the mean time, there's the PoliCTF Nov 17
  - Organized by Politecnico di Milano in Italy
  - Open to all, not just students
  - www.polictf.it
- Email me (cvwright@cs.pdx.edu) if you're interested in participating in either

Roadmap for today's talk

- A brief history lesson
  – Graybeards and green screens
- Basic network protocols
  – TCP/IP and friends.. The series of tubes
- **What could possibly go wrong?**
  – Quite a lot, actually…

Quite a lot, actually…

WHAT COULD POSSIBLY GO WRONG?

Threat Model

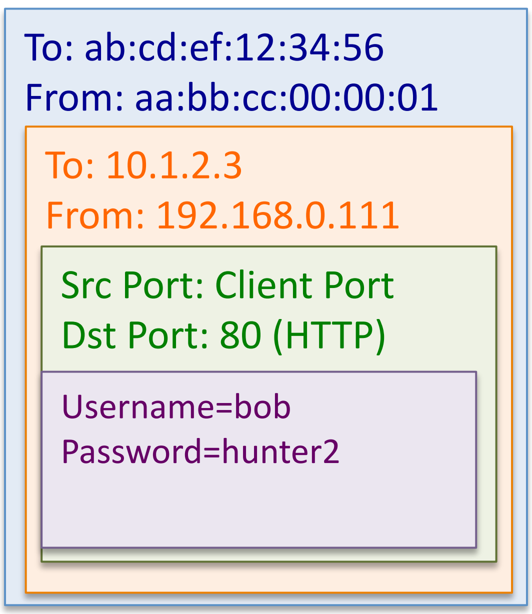
- We know now that not all users are friendly
- And many devices may be compromised
  - Desktop, laptops, even switches and routers!
- Need to be able to operate in a hostile environment

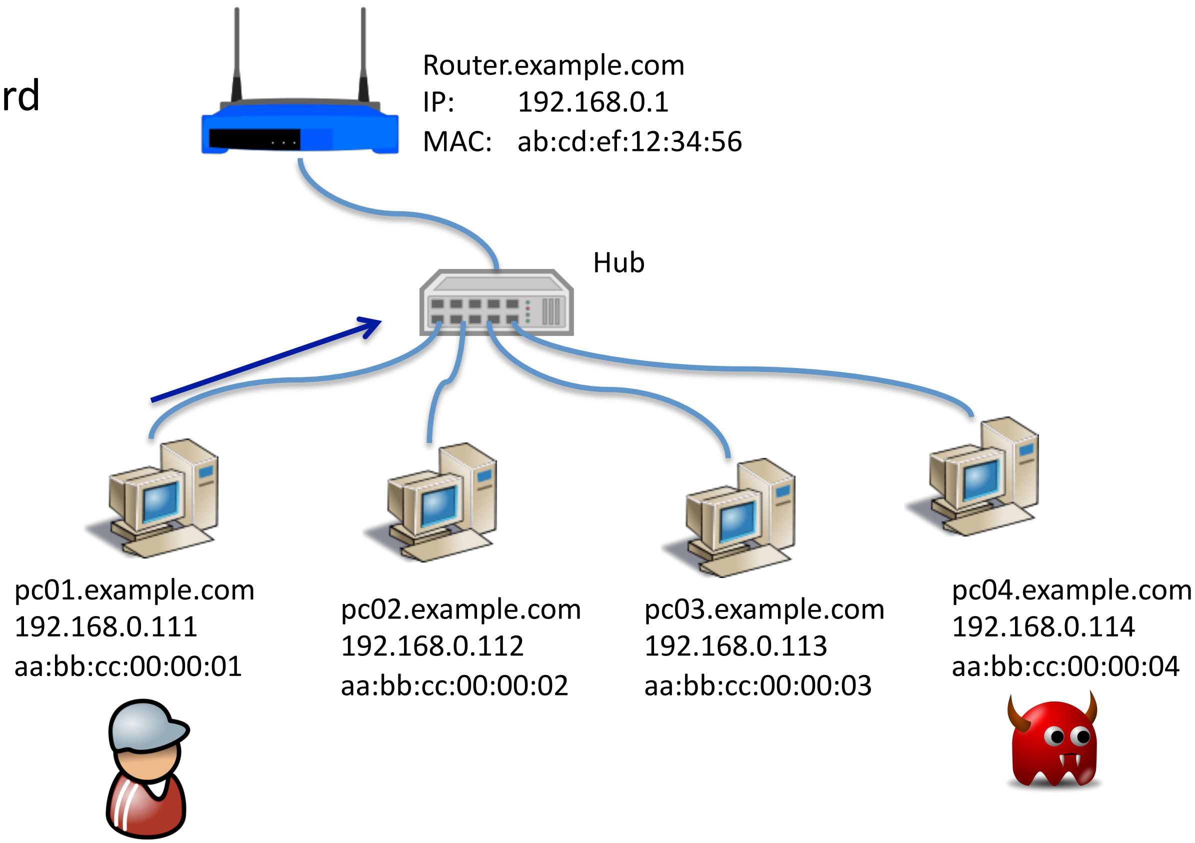
Dolev-Yao Attacker Model

- Let’s make that more concrete. We’ll assume:
  - Attacker can **read any packet** while in transit
  - Attacker can **modify packets** arbitrarily in transit
  - Attacker can **inject new packets** of his choosing

Ethernet: Snooping

PC01 sends an IP packet containing Bob's password





Ethernet: Snooping

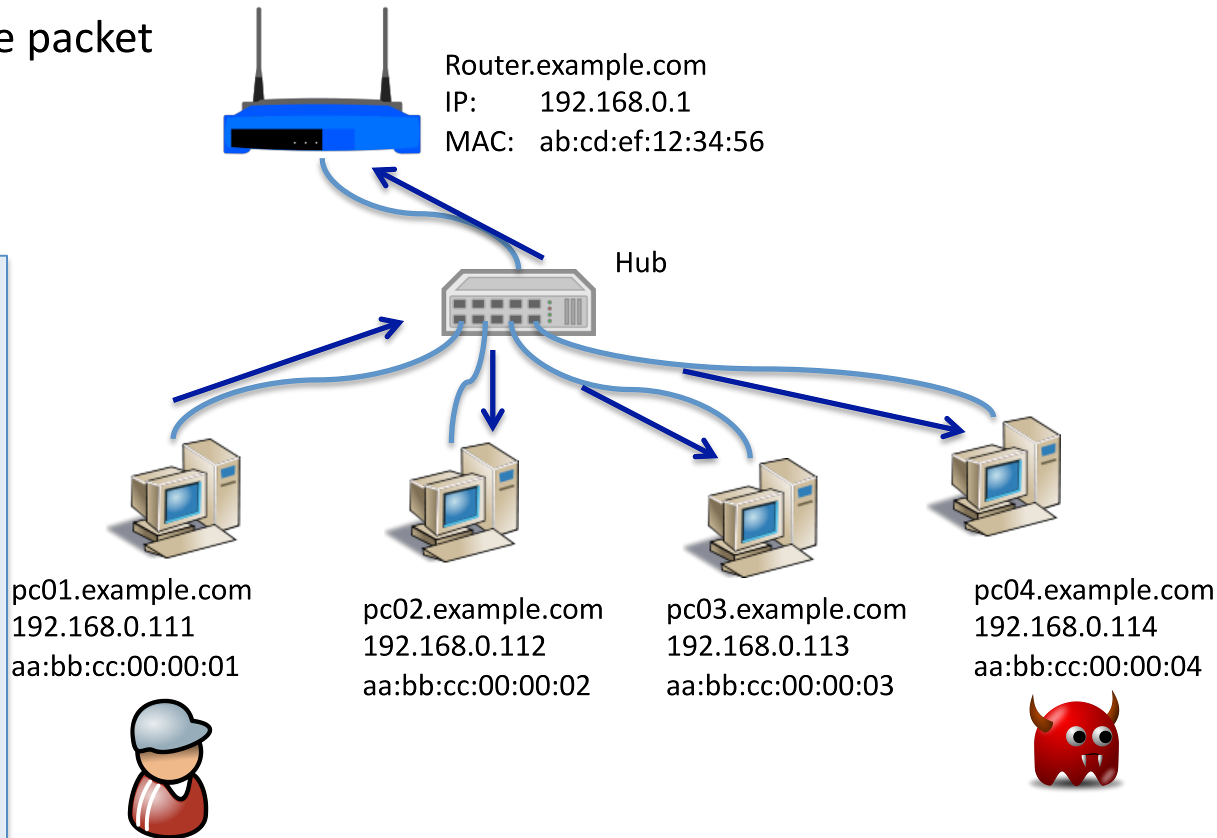
Ethernet hub copies the packet
to all other ports

To: ab:cd:ef:12:34:56
From: aa:bb:cc:00:00:01

To: 10.1.2.3
From: 192.168.0.111

Src Port: Client Port
Dst Port: 80 (HTTP)

Username=bob
Password=hunter2



Ethernet: Snooping

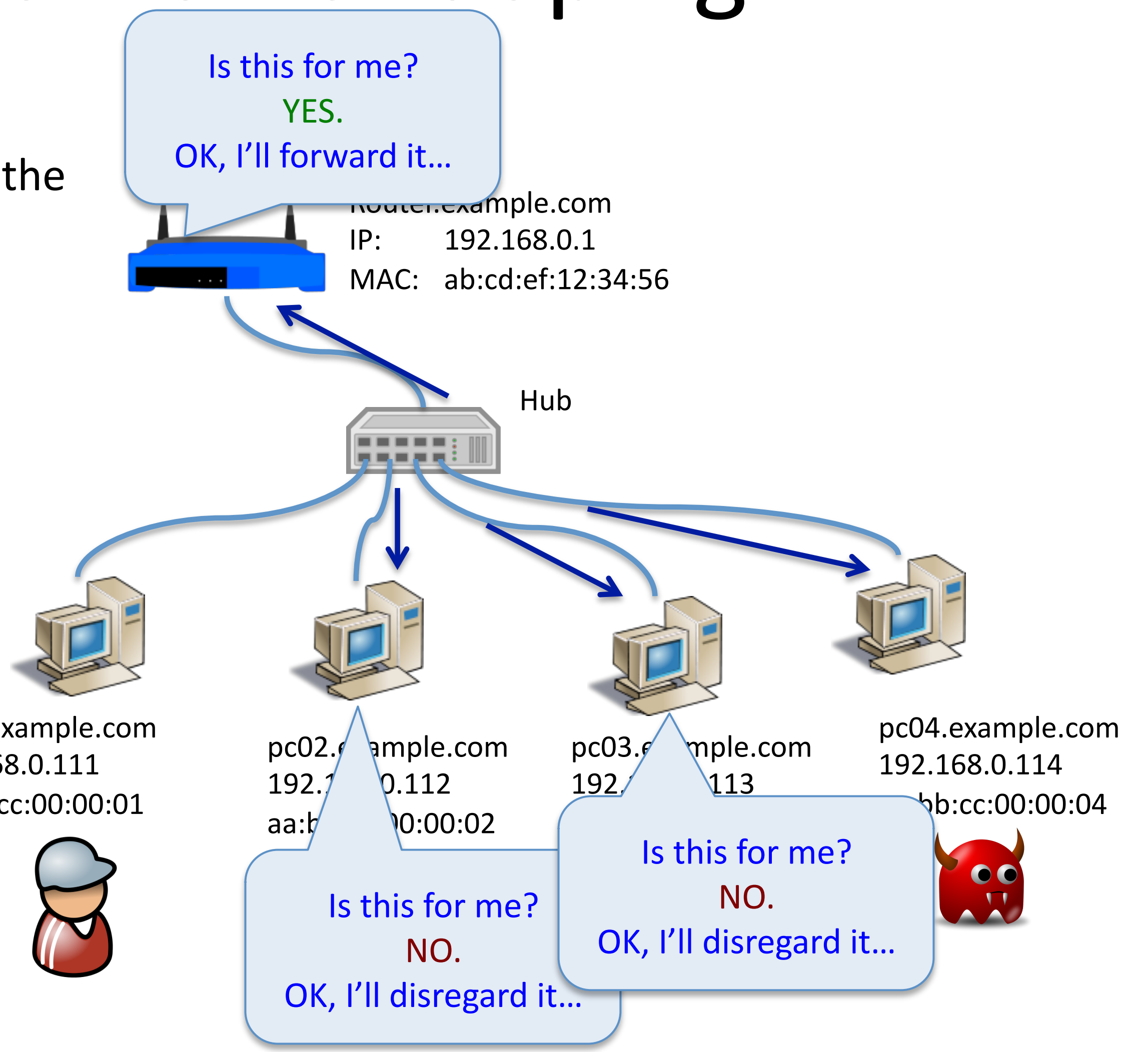
Normally, nodes who aren't the Ethernet destination simply drop the packet

To: ab:cd:ef:12:34:56
From: aa:bb:cc:00:00:01

To: 10.1.2.3
From: 192.168.0.111

Src Port: Client Port
Dst Port: 80 (HTTP)

Username=bob
Password=hunter2



Ethernet: Snooping

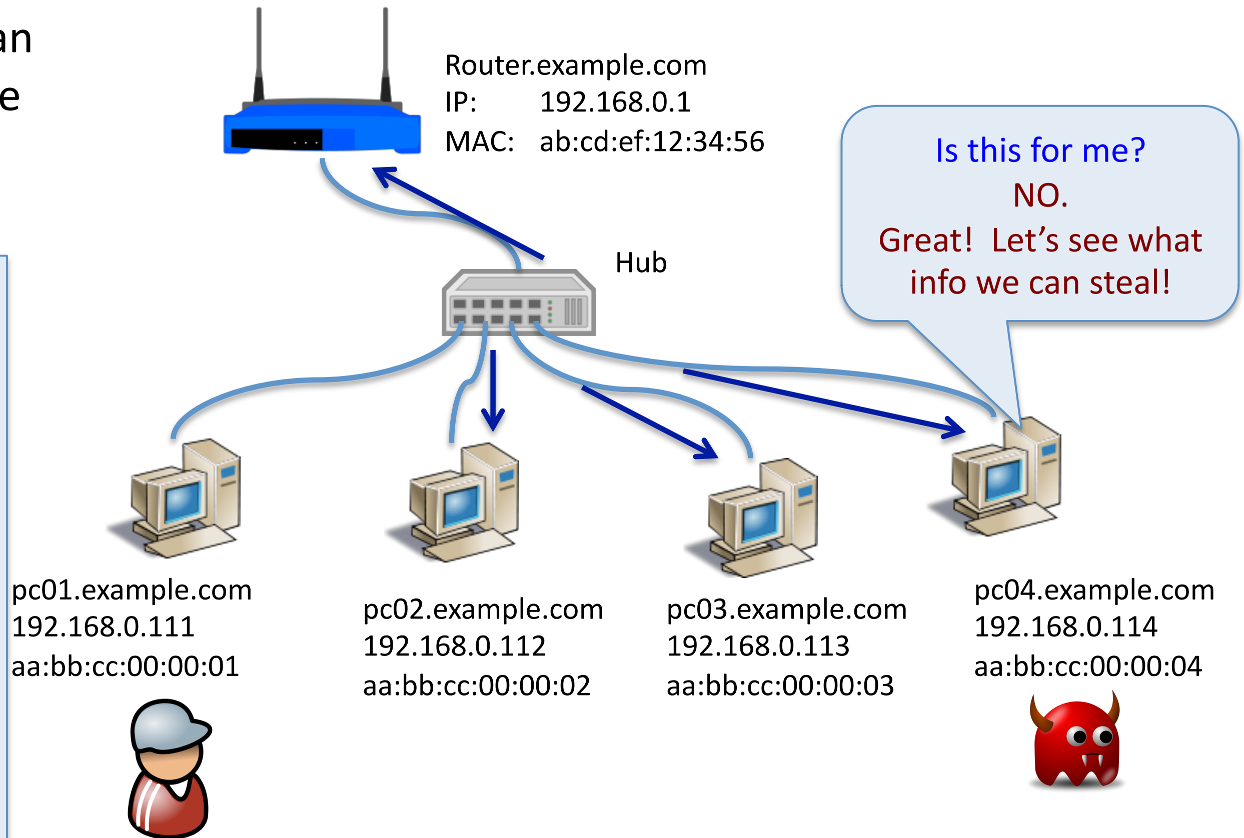
But a malicious node can still look at what's in the packet anyway

To: ab:cd:ef:12:34:56
From: aa:bb:cc:00:00:01

To: 10.1.2.3
From: 192.168.0.111

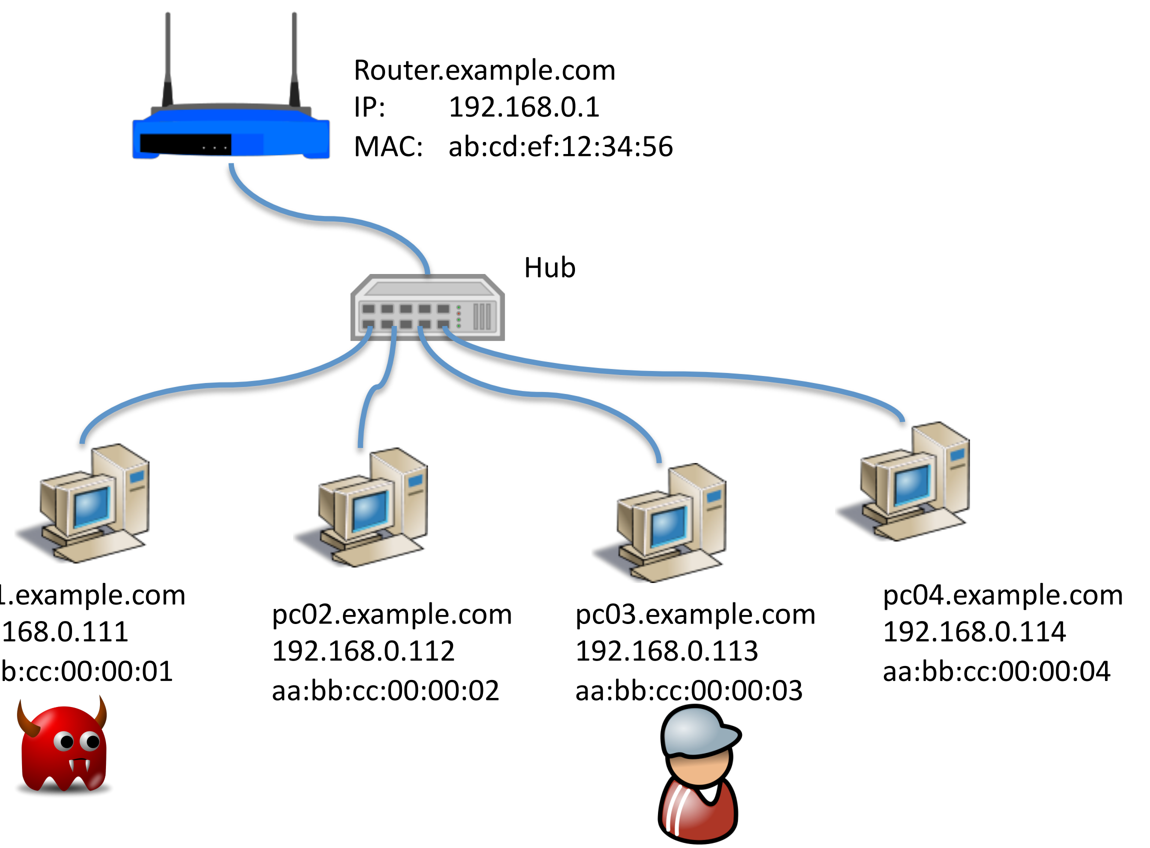
Src Port: Client Port
Dst Port: 80 (HTTP)

Username=bob
Password=hunter2



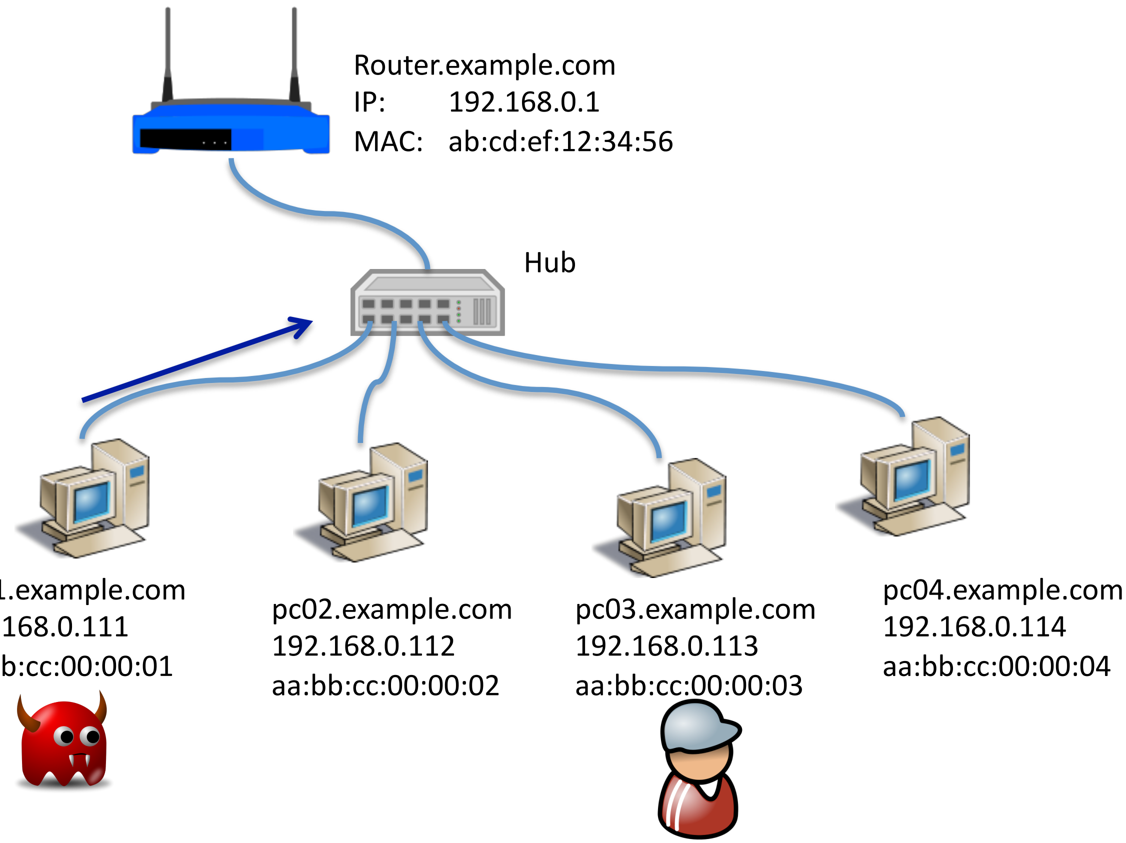
IP: Address Spoofing Attack

Malicious user can forge his source address to frame some other IP or machine



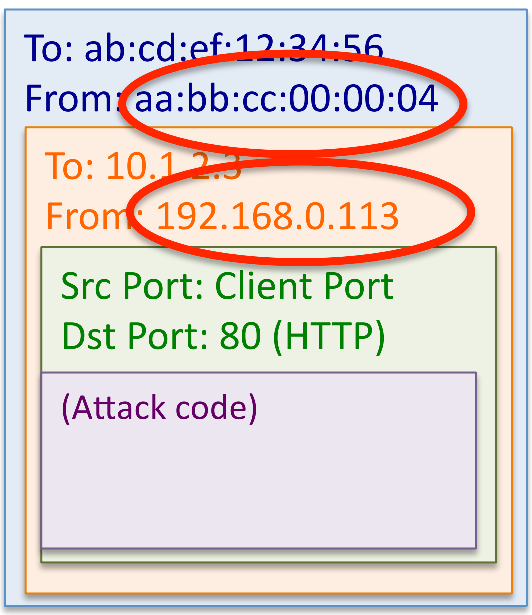
IP: Address Spoofing Attack

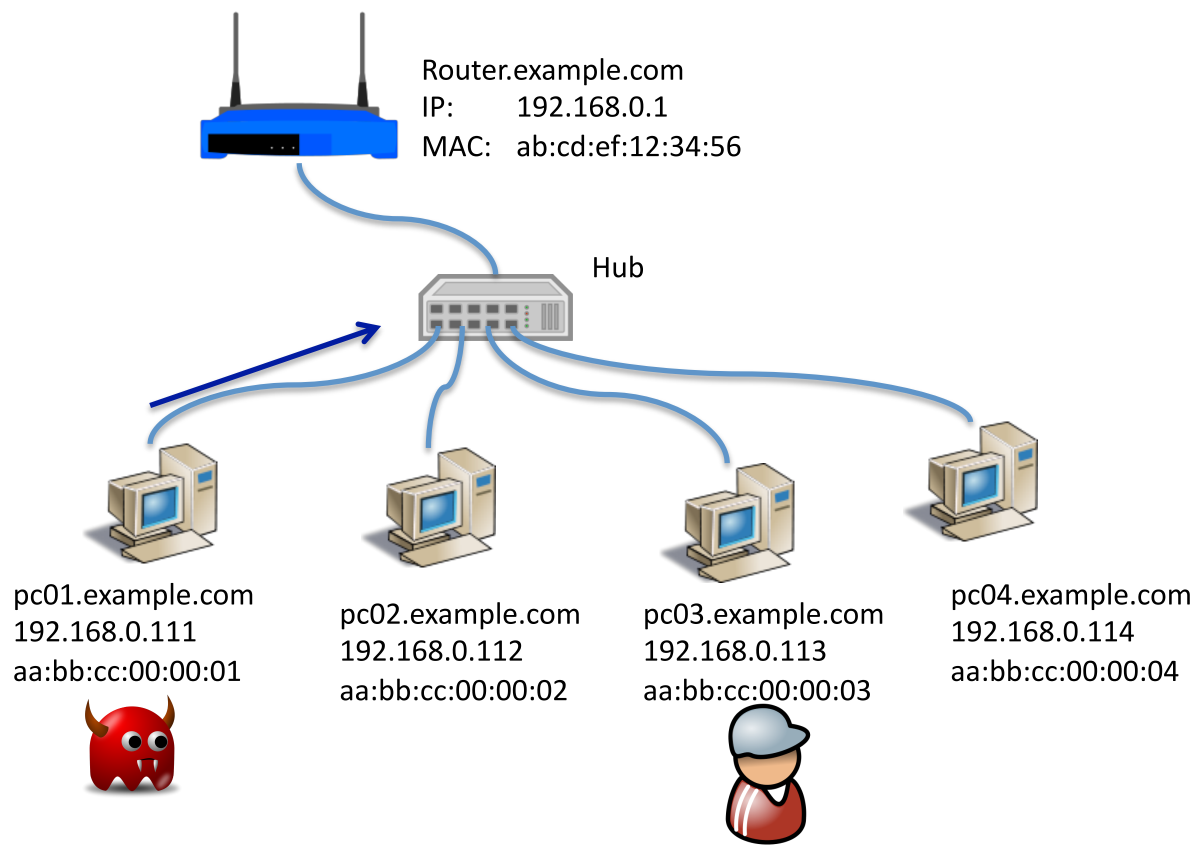
Malicious user can forge his source address to frame some other IP or machine



IP: Address Spoofing Attack

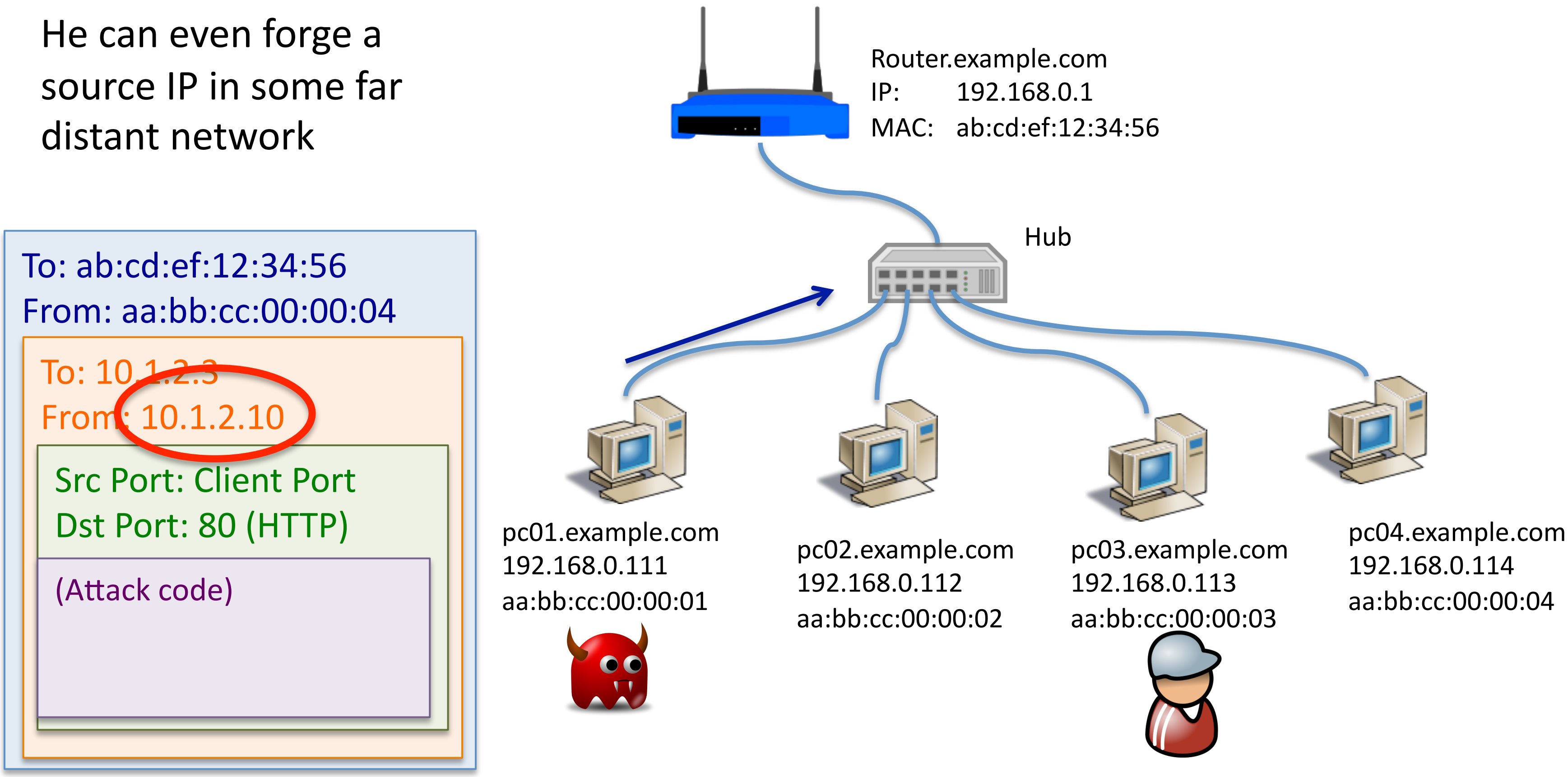
Anyone who checks will think this attack came from Bob





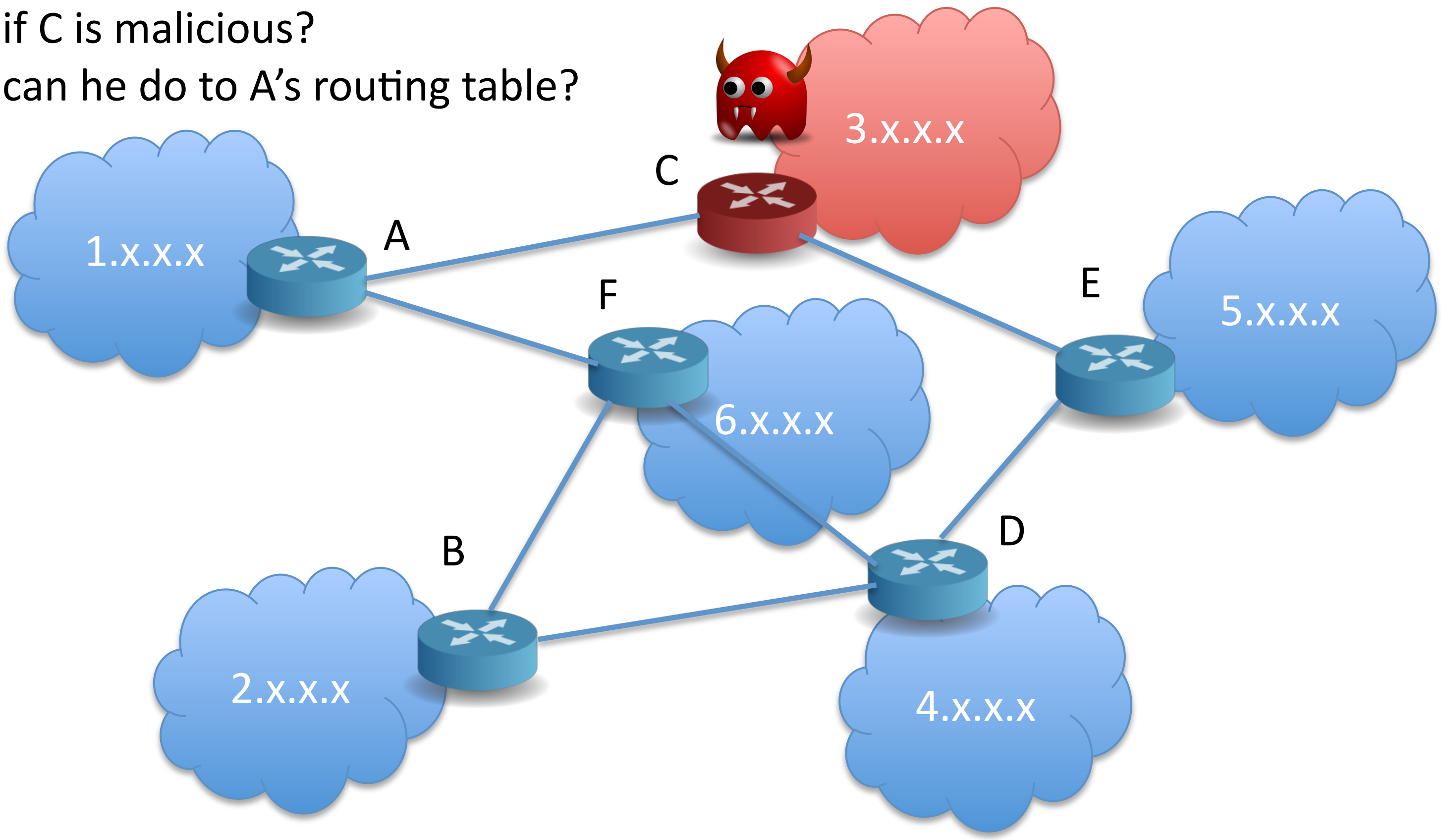
IP: Address Spoofing Attack

He can even forge a source IP in some far distant network



IP Routing: “Blackhole” Attack

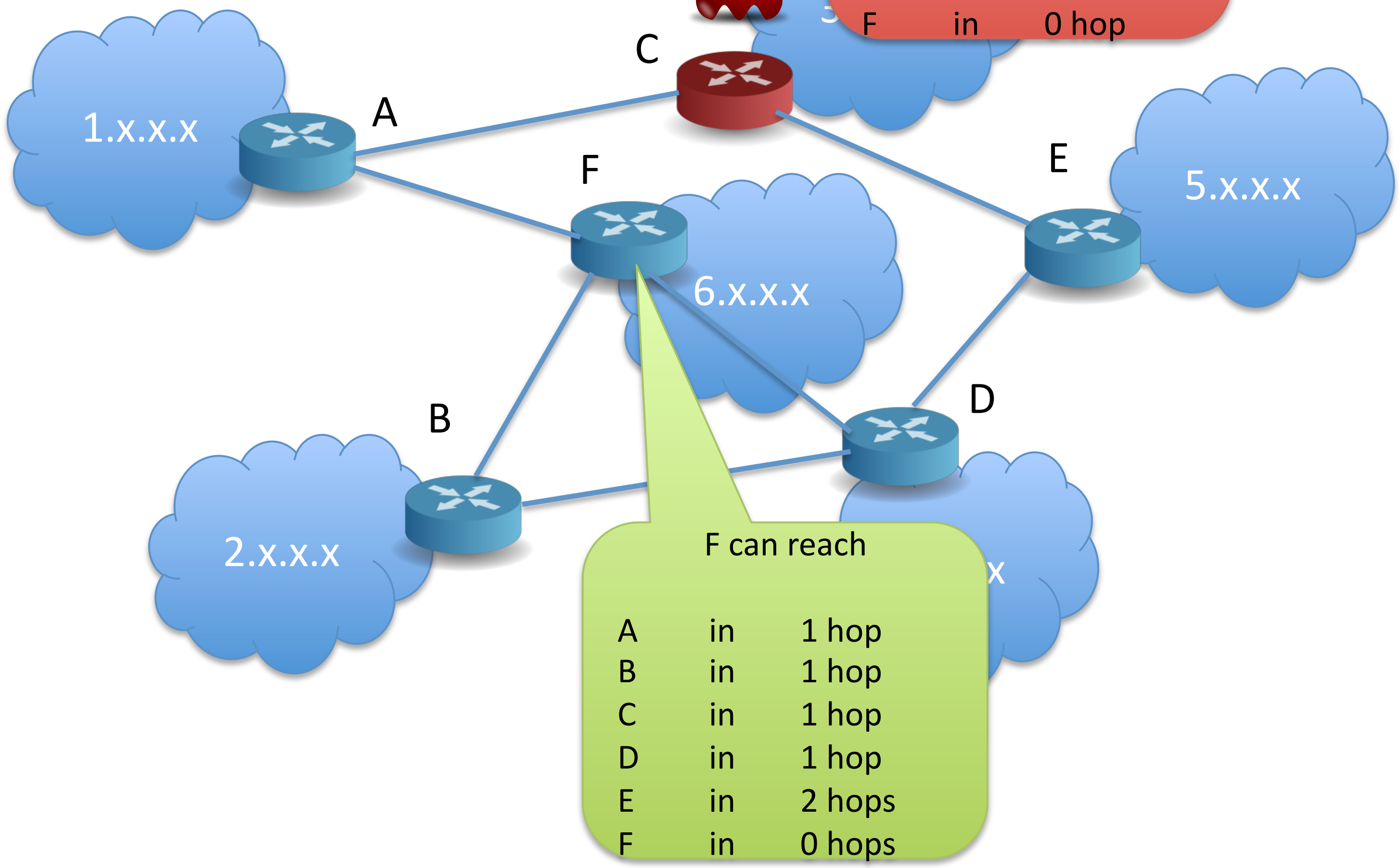
What if C is malicious?
What can he do to A’s routing table?



IP Routing: “Blackh ack

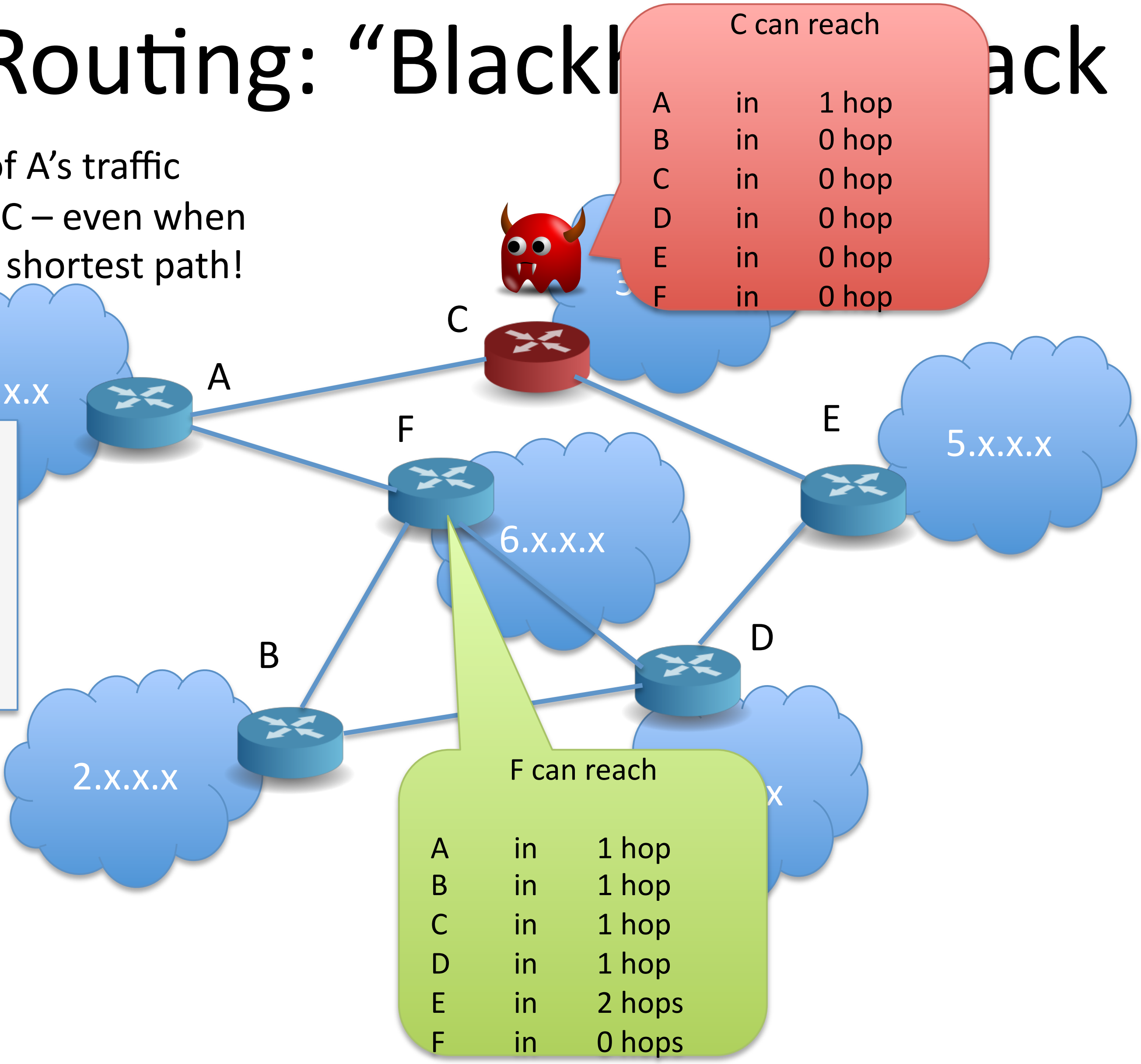
What if C is malicious?

What can he do to A’s routing table?



IP Routing: “Black ack

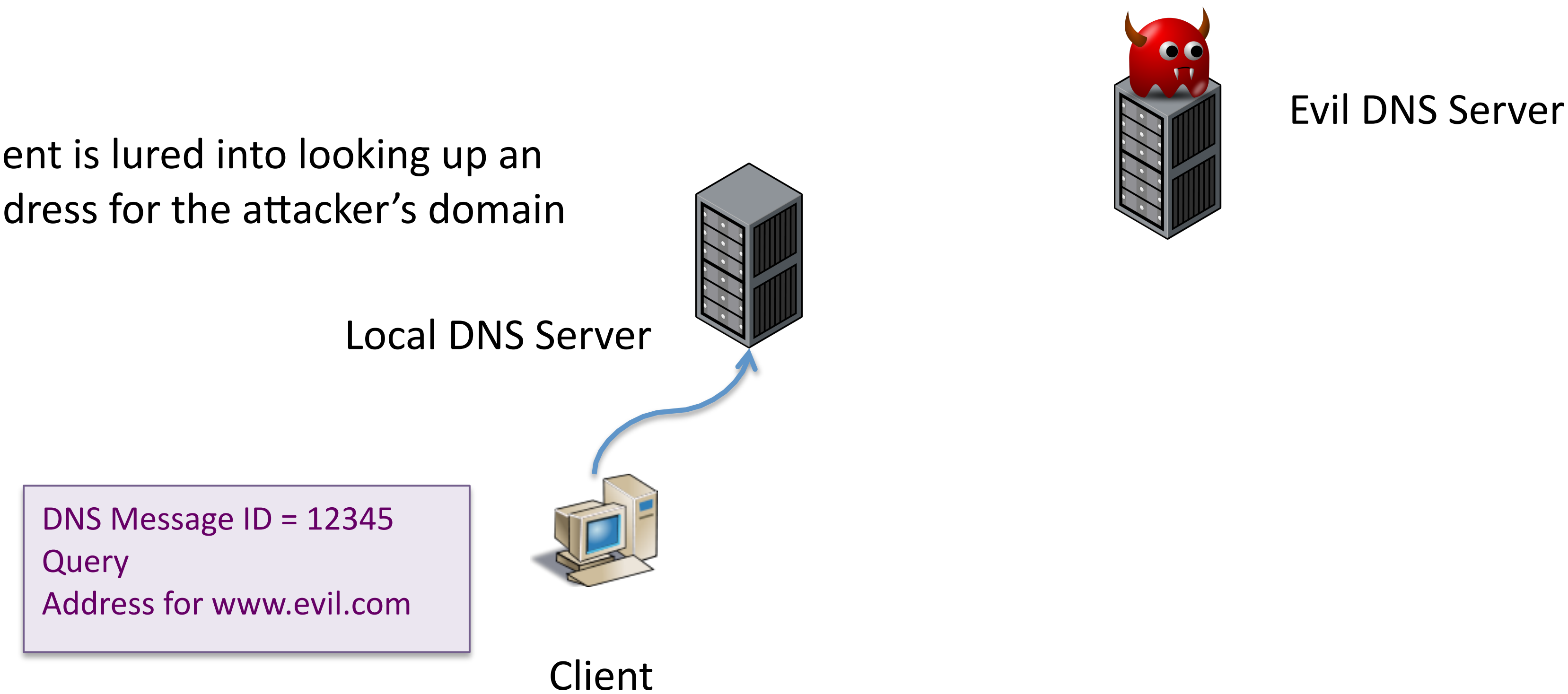
Suddenly all of A’s traffic goes through C – even when that’s not the shortest path!



Dst	Next
1.*.*.*	self
2.*.*.*	C
3.*.*.*	C
4.*.*.*	C
5.*.*.*	C
6.*.*.*	C

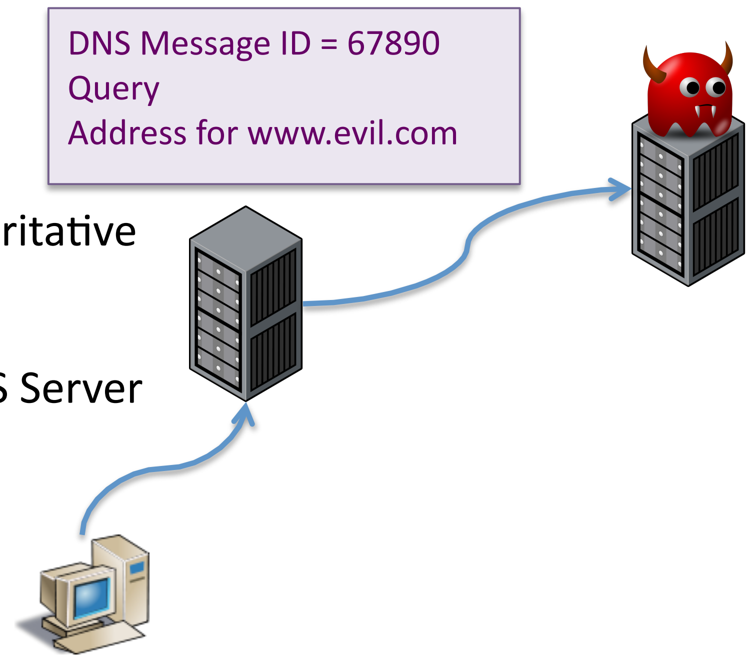
DNS Poisoning

Client is lured into looking up an address for the attacker's domain



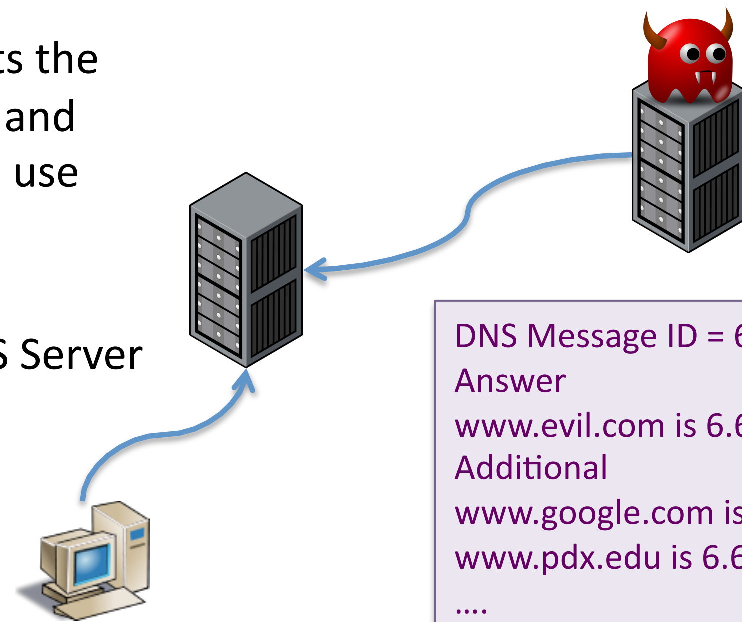
DNS Poisoning

Local DNS server asks the authoritative DNS server for the answer



DNS Poisoning

Local DNS server accepts the helpfully-provided info, and remembers it for future use



Evil DNS Server

Local DNS Server

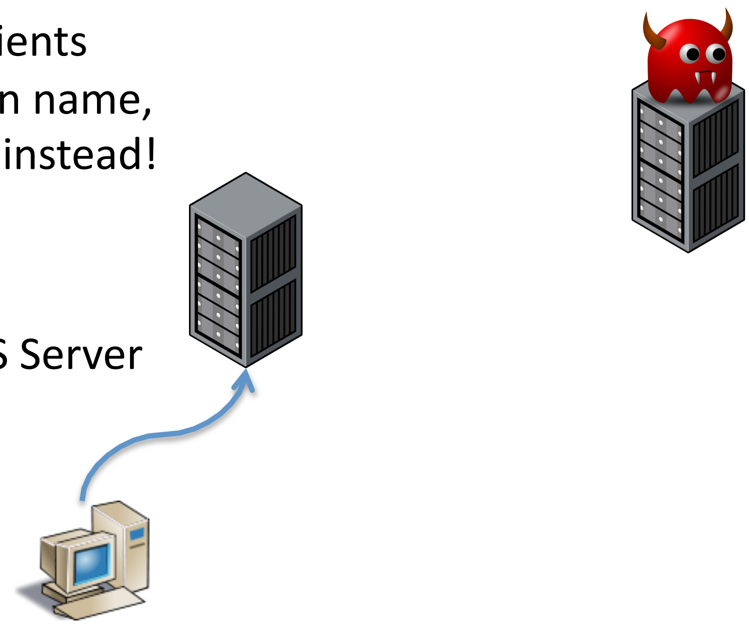
DNS Message ID = 67890
Answer
www.evil.com is 6.6.6.6
Additional
www.google.com is 6.6.6.6
www.pdx.edu is 6.6.6.6
….

Client

Evil DNS server returns the correct answer, along with some extra fun

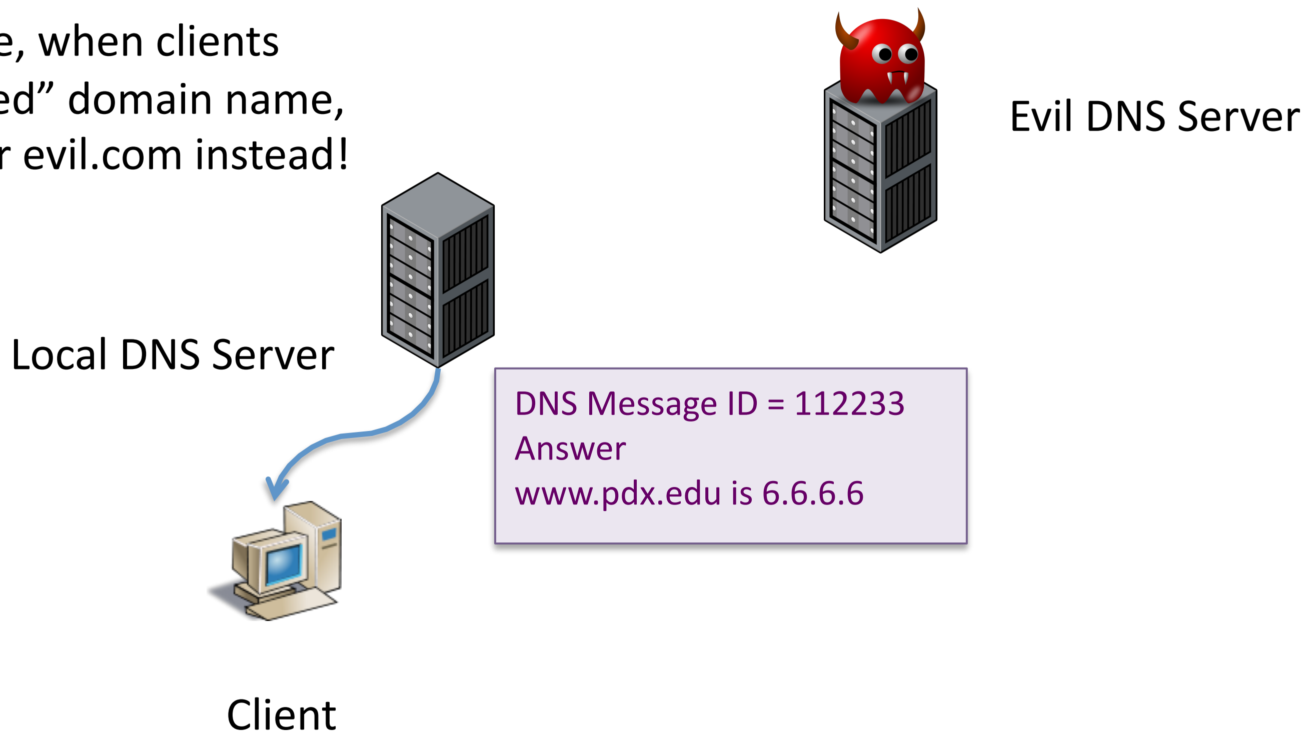
DNS Poisoning

Then in the future, when clients ask for a "poisoned" domain name, they get the IP for evil.com instead!



DNS Poisoning

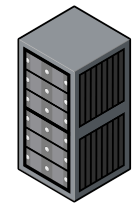
Then in the future, when clients ask for a "poisoned" domain name, they get the IP for evil.com instead!



TCP: SYN Flood Attack

Attacker

Server



Attacker sends new SYN packets to the server, using a **new client port** each time. He doesn't actually create any local state for new sockets.

`listen()`

Server waits for new connections

SYN →

Server creates new socket with 4-tuple ID (client IP, client **port1**, server IP, server port)

← **SYN/ACK**

Attacker ignores SYN/ACK packets from server

SYN →

← **SYN/ACK**

Server creates new socket with 4-tuple ID (client IP, client **port2**, server IP, server port)

Attacker continues until server runs out of file descriptors

SYN →

SYN →

Server has no resources to handle legitimate clients ☹

Telnet: What could possibly go wrong?

S: Welcome to host.example.com
S: login:
C: b
S: b
C: o
S: o
C: b<CR><LF>
S: b
S: password:
C: <sends each character of the password>
S: <checks password; if password matches, prints message of the day, etc, …, starts a shell for bob>
S: [bob@host ~]$
C: <sends each character as Bob types commands>
S: <echoes each character>
S: <prints output from running the command>
C: …
S: …

S: Welcome to host.example.com

S: login:

C: b

S: b

C: o

S: o

C: b<CR><LF>

S: b

S: password:

C: <sends each character of the password>

Attacker can read Bob's password!

S: <checks password; if password matches, prints message of the day, etc, …, starts a shell for bob>

S: [bob@host ~]$

C: <sends each character as Bob types commands>

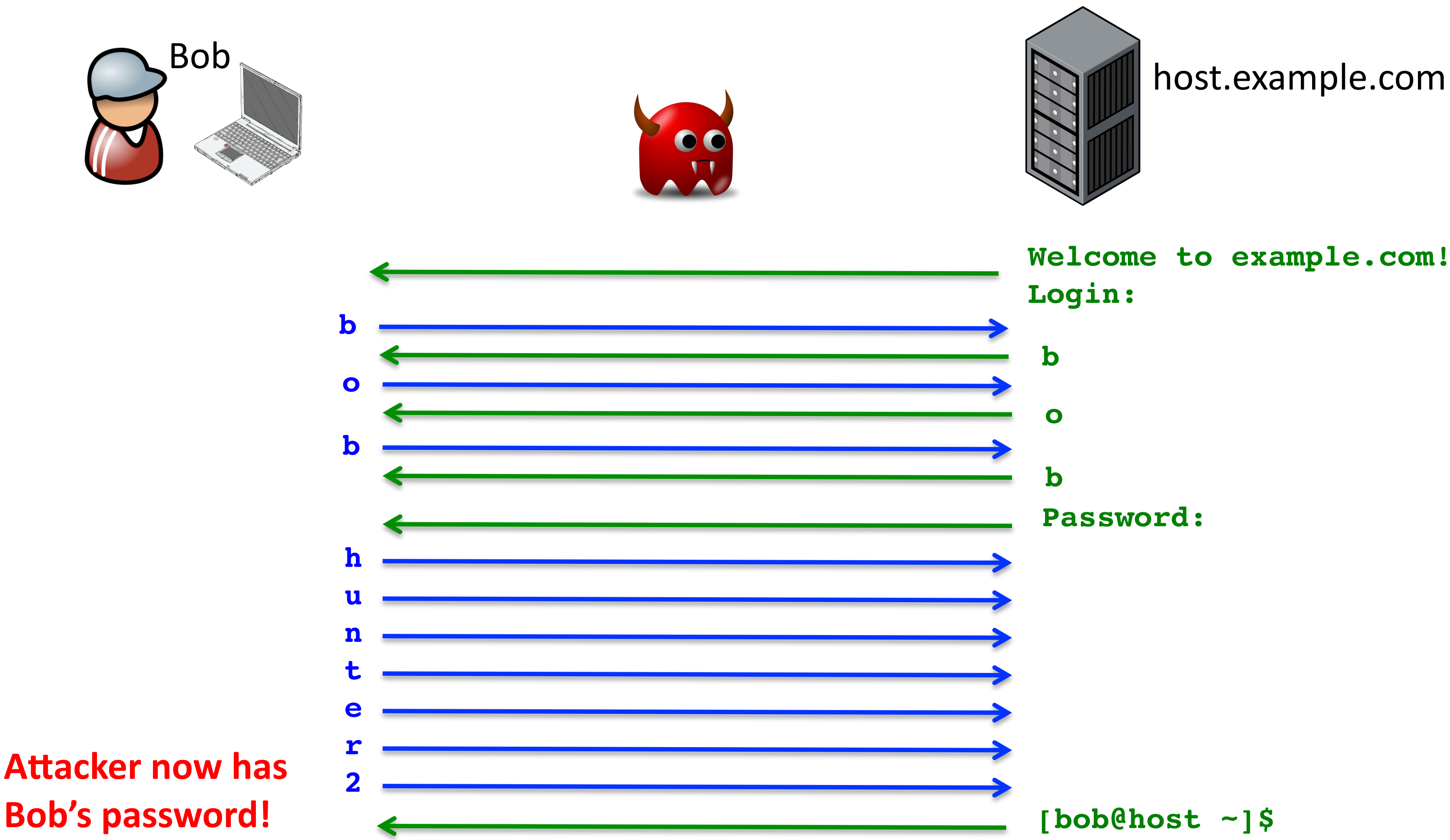
S: <echoes each character>

S: <prints output from running the command>

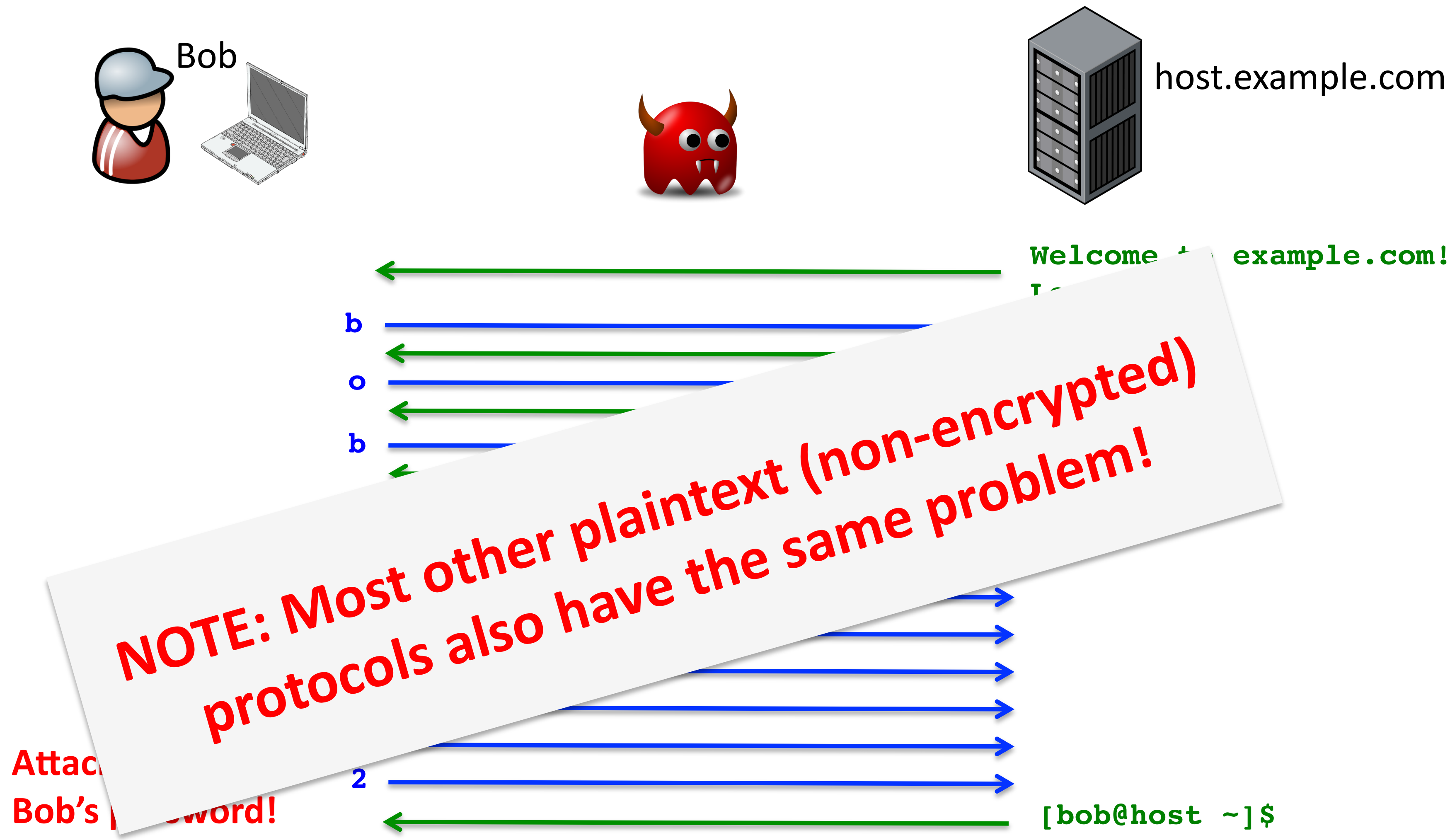
C: …

S: …

Telnet: Password Snooping



Telnet: Password Snooping



S: Welcome to host.example.com

S: login:

C: b

S: b

C: o

S: o

C: b<CR><LF>

S: b

S: password:

C: <sends each character of the password>

S: <checks password; if password matches, prints message of the day, etc, …, starts a shell for bob>

S: [bob@host ~]$

C: <sends each character as Bob types commands>

S: <echoes each character>

S: <prints output from running the command>

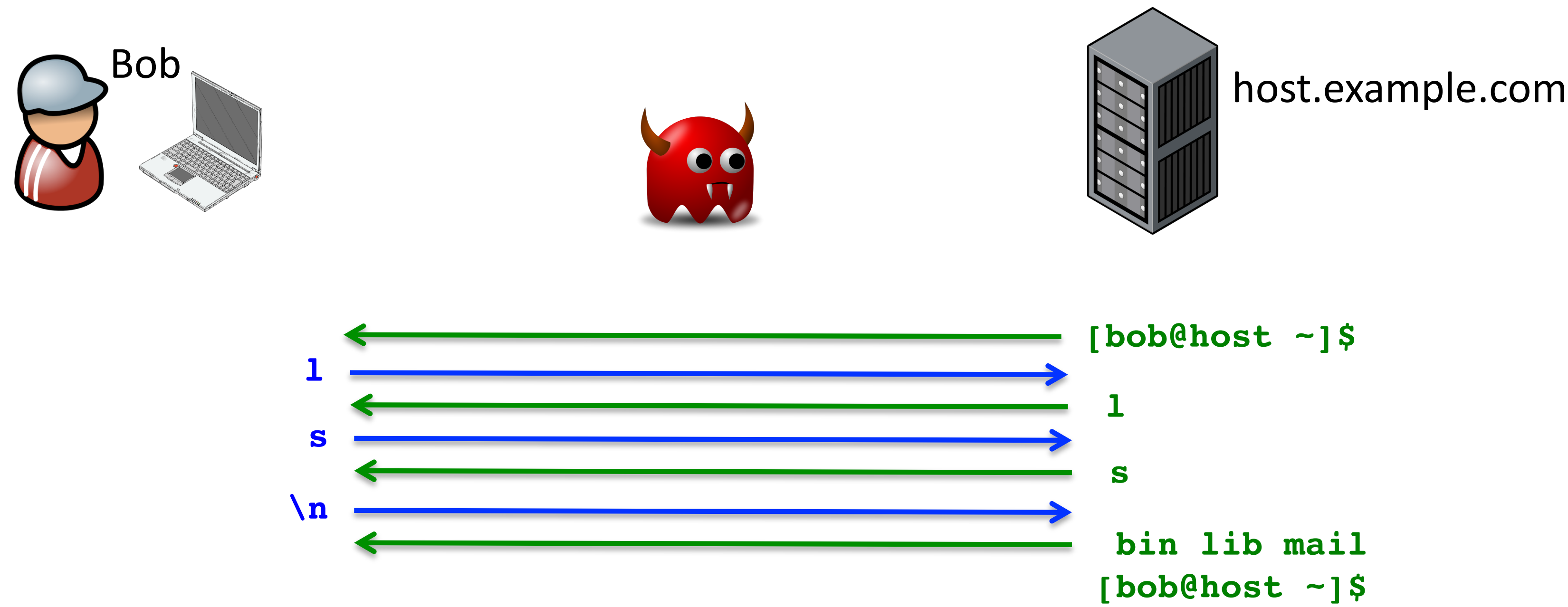
C: …

S: …

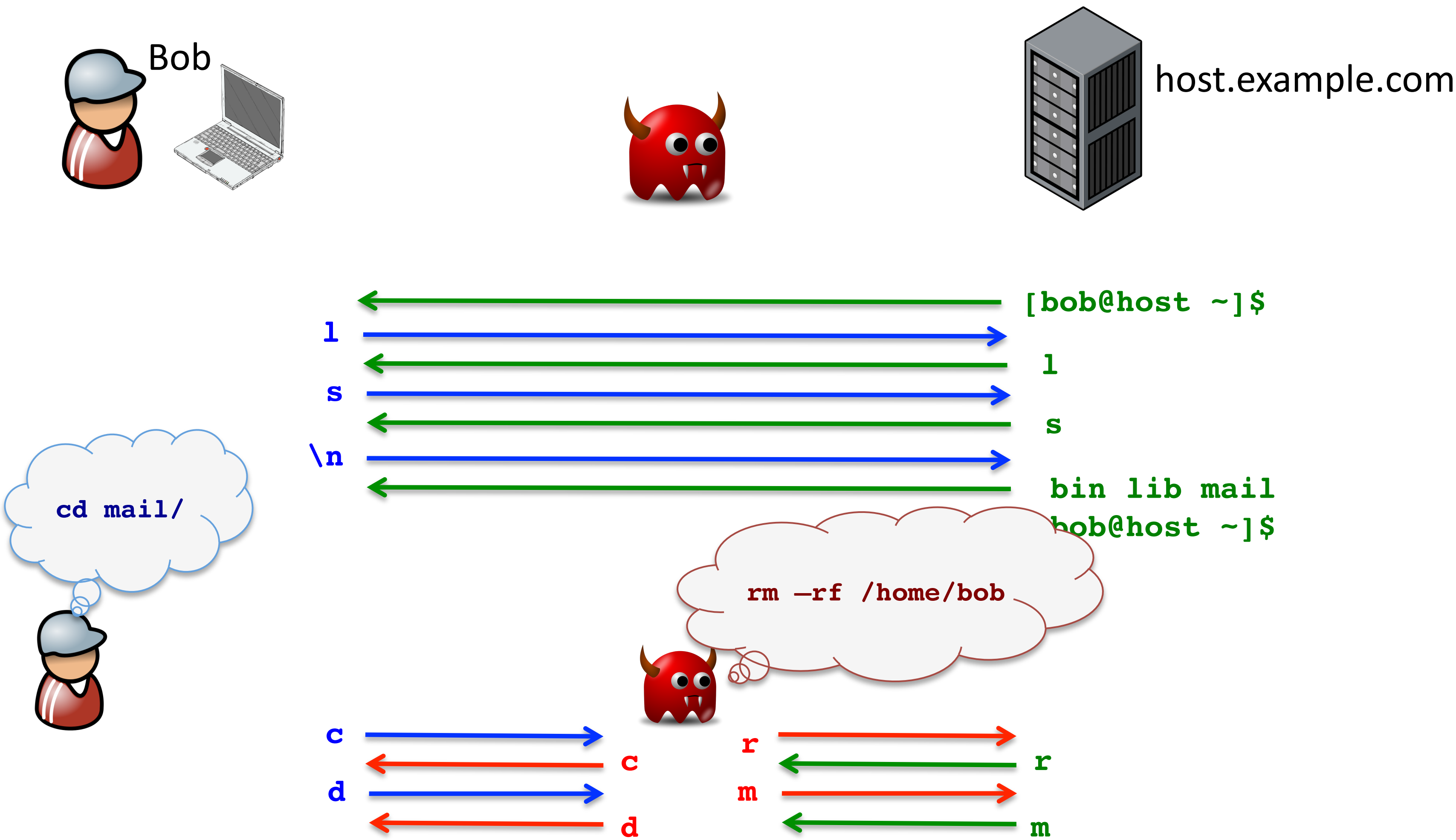
Attacker can inject his own commands!

And if the attacker is clever, Bob will never know it's happening!

Telnet: Man-in-the-Middle Attack

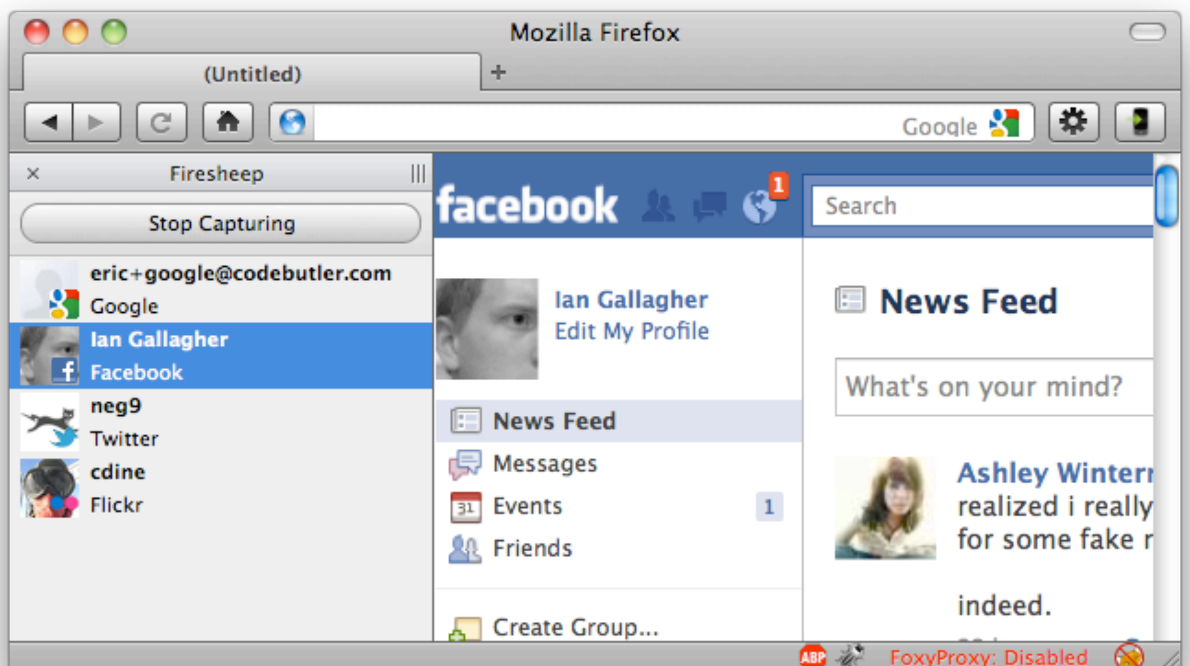


Telnet: Man-in-the-Middle Attack



HTTP: More Credential Snooping

- Is snooping unique to passwords? No! Example: Firesheep
  - Firefox extension
  - Snoops on shared wireless links (like at a coffee shop)
  - Copies session cookies from victim's web browsing traffic
  - Gives attacker (temporary) access to victim's accounts

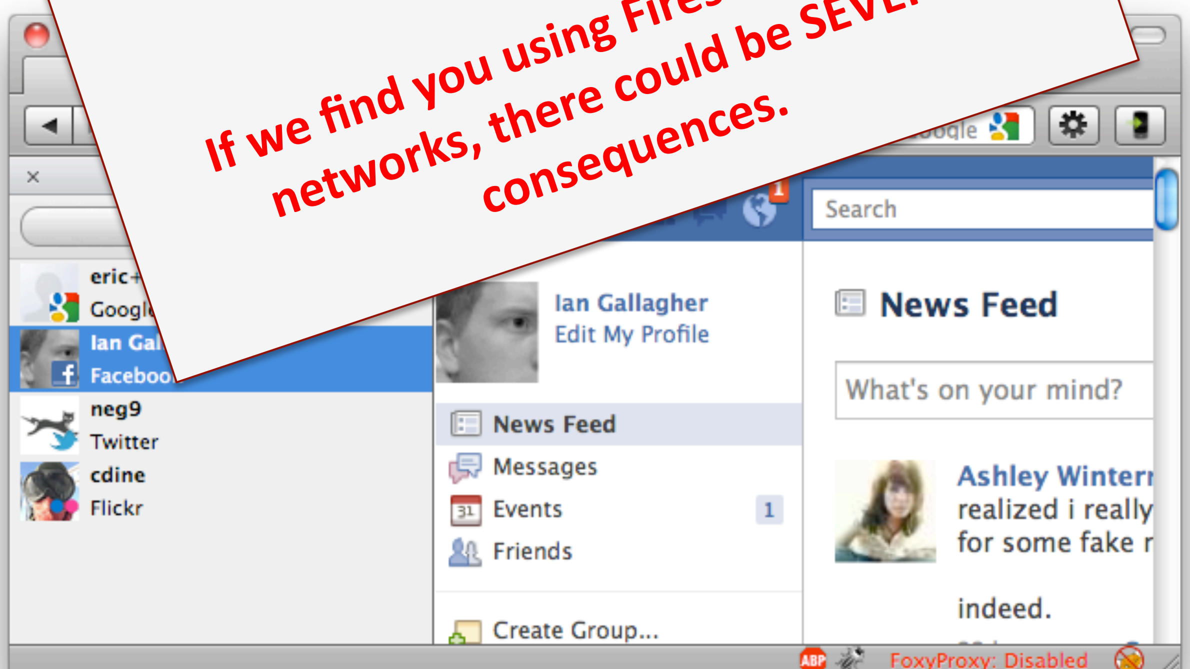


HTTP: More Credential Snooping

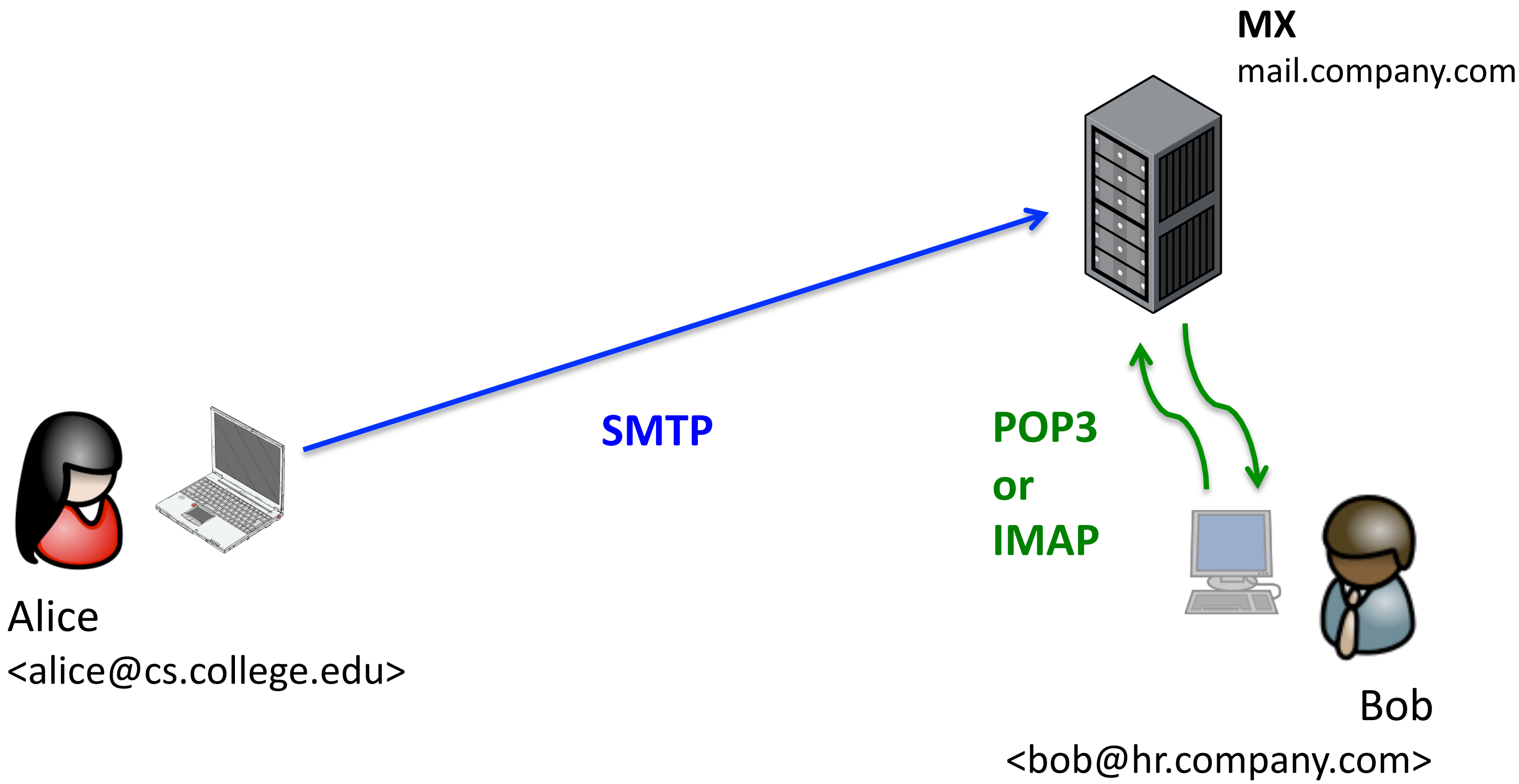
- Is snooping unique to passwords? No! … mple: Firesheep
  - Firefox extension
  - Snoops on shared w… …pp)
  - Copies s… …traffic
  - Giv… …s

Use this information to protect yourself, NOT to hurt others!

If we find you using Firesheep on real networks, there could be SEVERE consequences.



SMTP: What could go wrong?



SMTP: What could possibly go wrong?

S: 220 smtp.example.com ESMTP Postfix
C: HELO relay.example.org
S: 250 Hello relay.example.org, I am glad to meet you
C: MAIL FROM:<alice@cs.college.edu>
S: 250 Ok
C: RCPT TO:<bob@hr.company.com>
S: 250 Ok
C: RCPT TO:<theboss@example.com>
S: 250 Ok
C: DATA
S: 354 End data with <CR><LF>.<CR><LF>
C: From: ”Alice Student" <alice@cs.college.edu>
C: To: ”Bob Manager" <bob@hr.company.com>
C: Cc: theboss@example.com
C: Date: Tue, 15 January 2008 16:02:43 -0500
C: Subject: Test message
C:
C: Hello Bob.
C: This is a test message with 5 header fields and 4 lines in the message body.
C: Sincerely,
C: Alice
C: .
S: 250 Ok: queued as 12345
C: QUIT
S: 221 Bye {The server closes the connection}

S: 220 smtp.example.com ESMTP Postfix

C: HELO relay.example.org

S: 250 Hello relay.example.org, I am glad to meet you

C: MAIL FROM:<alice@cs.college.edu>

S: 250 Ok

C: RCPT TO:<bob@hr.company.com>

S: 250 Ok

C: RCPT TO:<theboss@example.com>

S: 250 Ok

C: DATA

S: 354 End data with <CR><LF>.<CR><LF>

C: From: "Alice Student" <alice@cs.college.edu>

C: To: "Bob Manager" <bob@hr.company.com>

C: Cc: theboss@example.com

C: Date: Tue, 15 January 2008 16:02:43 -0500

C: Subject: Test message

C:

C: Hello Bob.

C: This is a test message with 5 header fields and 4 lines in the message body.

C: Sincerely,

C: Alice

C: .

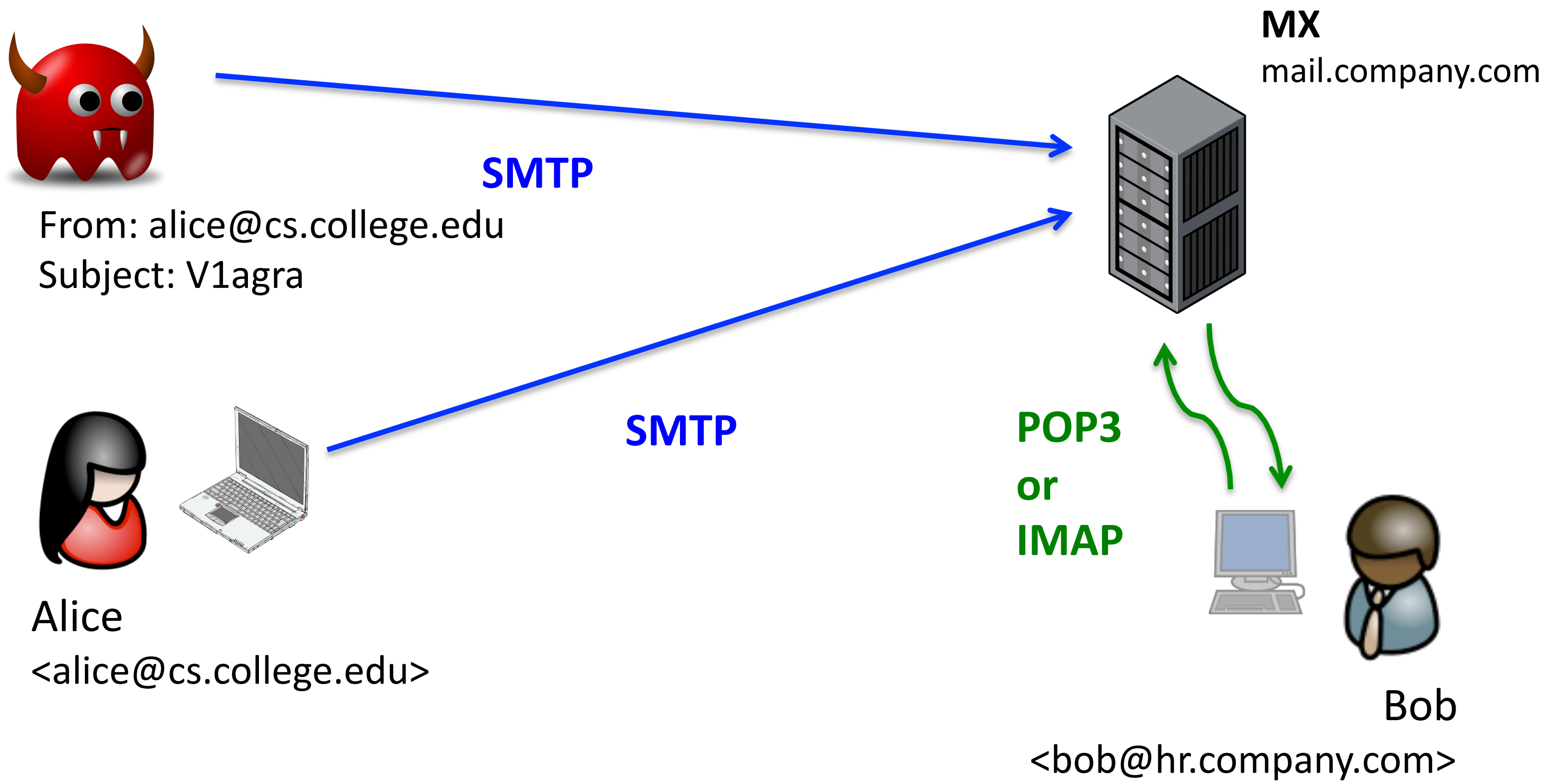
S: 250 Ok: queued as 12345

C: QUIT

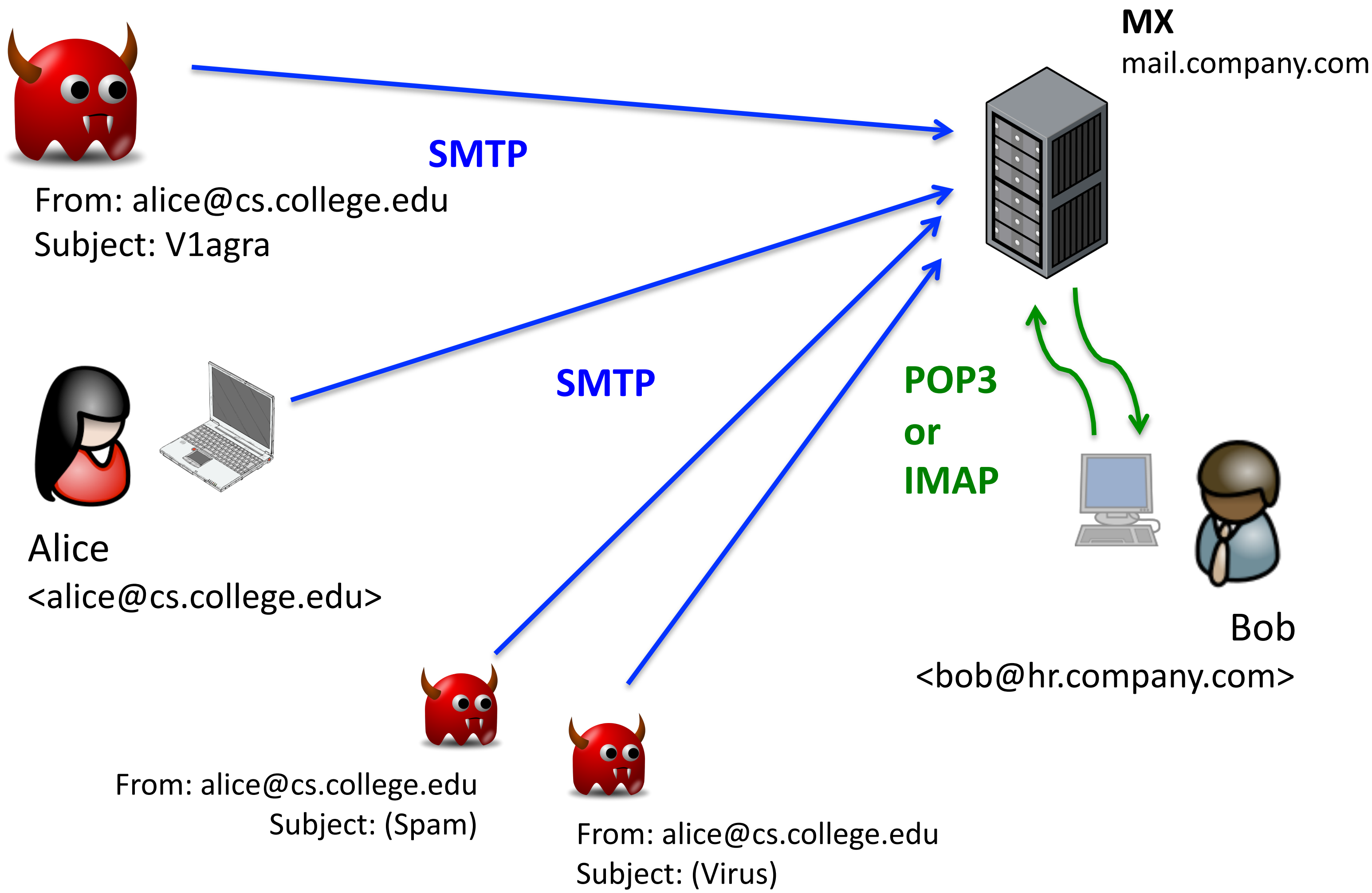
S: 221 Bye {The server closes the connection}

How do we know this is really from Alice?

SMTP: “Joe Job” Attack



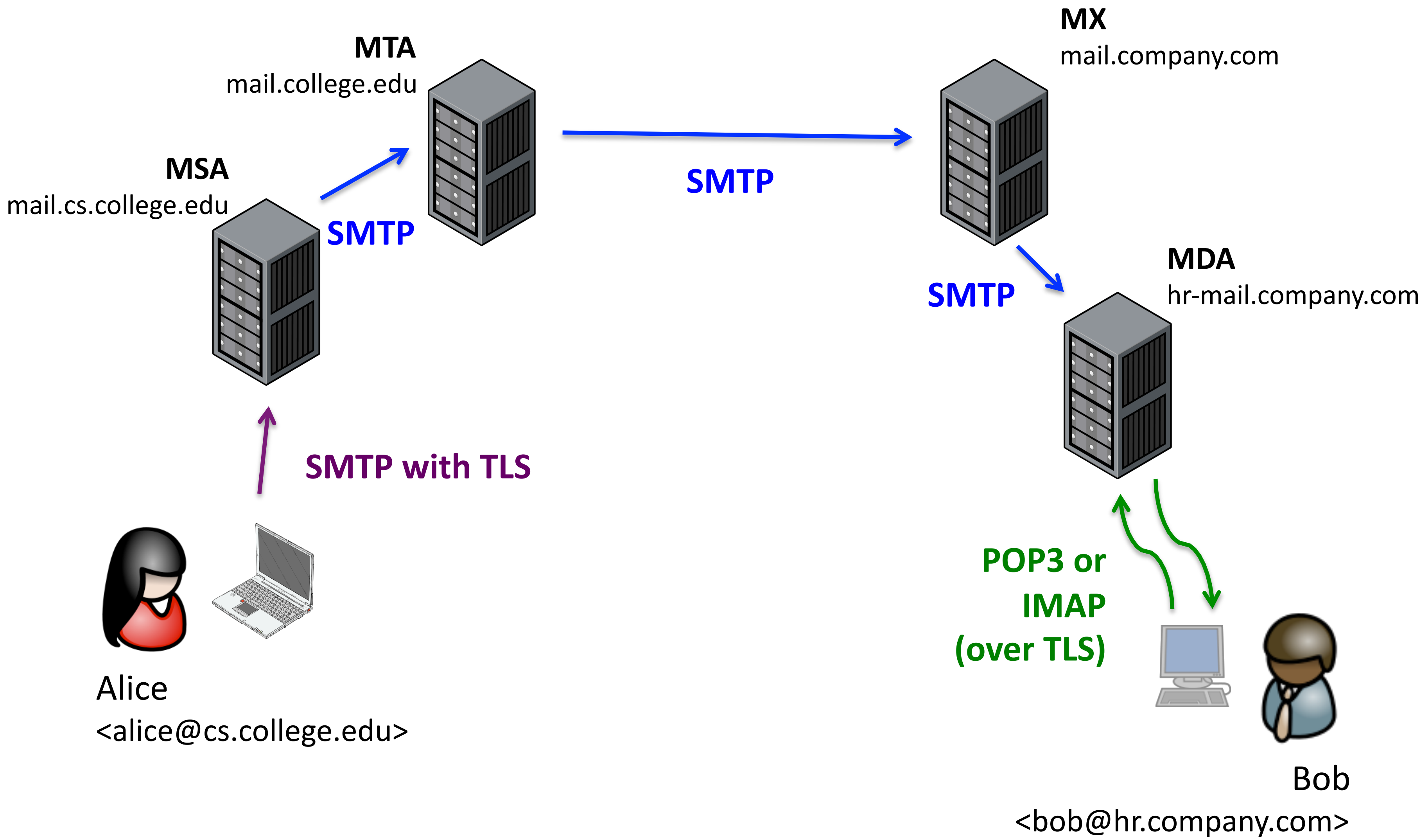
SMTP: "Joe Job" Attack



S: 220 smtp.example.com ESMTP Postfix
C: HELO relay.example.org
S: 250 Hello relay.example.org, I am glad to meet you
C: MAIL FROM: randomguy@untrusted.com
S: 250 Ok
C: RCPT TO:<bob@hr.company.com>
S: 250 Ok
C: RCPT TO:<theboss@example.com>
S: 250 Ok
C: DATA
S: 354 End data with <CR><LF>.<CR><LF>
C: From: ”Alice Student" goodguy@college.edu
C: To: ”Bob Manager" <bob@hr.company.com>
C: Cc: theboss@example.com
C: Date: Tue, 15 January 2008 16:02:43 -0500
C: Subject: Test message
C:
C: Hello Bob.
C: This is a test message with 5 header fields and 4 lines in the message body.
C: Sincerely,
C: Alice
C: .
S: 250 Ok: queued as 12345
C: QUIT
S: 221 Bye {The server closes the connection}

Does Bob see the same sender address that his mail server sees?

SMTP: Modern Usage



What did we learn today?

WRAP-UP

Why do these old protocols fail?

- First, they're awfully gullible
  - Assume everybody is your friend
  - Take input at face value
  - Don't verify its origin or its authenticity
  - Little or no sanity checking

Why do these old protocols fail?

- Second, they don’t protect sensitive data
  - Passwords and other info are sent “in the clear”
  - NOTE: Most of these were designed before modern cryptography even existed

BACKUP SLIDES