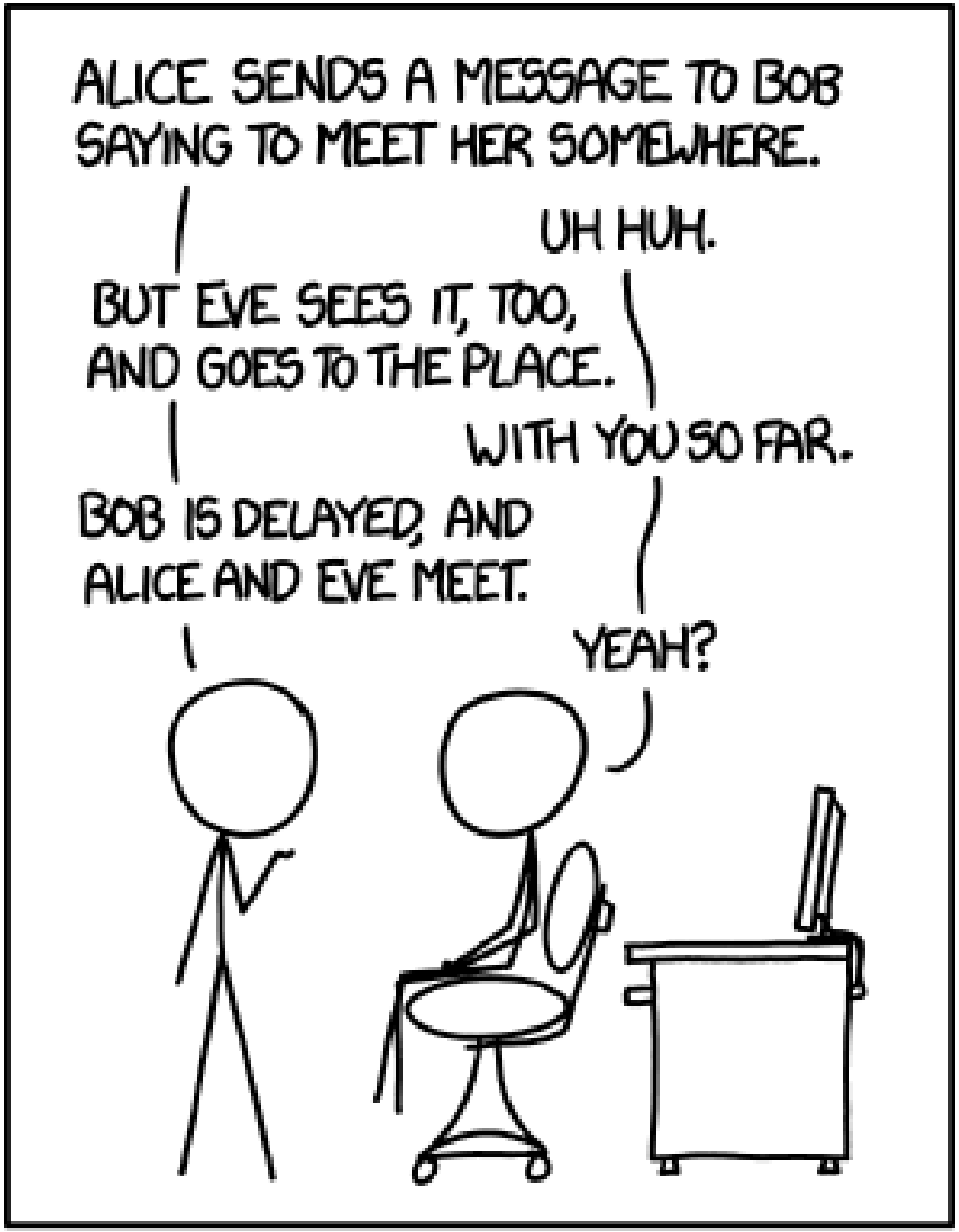


I'VE DISCOVERED A WAY TO GET COMPUTER SCIENTISTS TO LISTEN TO ANY BORING STORY.

Source: https://xkcd.com/1323/

CS 60:
Computer Networks

Application layer

Review from last class

1. Network layers are independent,
   - Can change a lower layer, but higher layers work the same
   - Example: Use Wi-Fi or Ethernet, Layer 3 doesn't care which was used
2. Layer 4: transport using TCP or UDP
   - TCP – requires 3-way handshake, detects and resends dropped packets, reassembles in order
   - UDP – no set up, does not resend or reorder packets
3. We use sockets to communicate over the network using a client/server model
4. Ports are an integer that identify a socket an application is using
5. To send to server:
   - We:
     - Open a TCP or UDP socket
     - Provide destination IP address and port
   - OS provides:
     - Source IP address and port
     - Layer 2 MAC addresses (dst is next hop router's MAC)
6. We can use Scapy to easily read layer header fields or spoof them!

Review: Application layer sends messages across a network

Conceptual network layers

7) Application

Interacts with application programs to send ***messages***
Applications assigned a port, multiple instances can run (many browser pages)
Examples: HTTP, SSH, FTP, SMTP, DNS

4) Transport

Moves ***segments (or datagrams)***
May provide error control, flow control, application addressing (ports)
Examples: TCP (connection-oriented), UDP (connectionless)
TCP provides sequencing, dropped packet resend, traffic congestion routing

3) Network (IP)

Moves ***packets*** between local area networks (routing)
Each computer on the Internet identified by an IP address (IP v4 or v6)
Also called Layer 3 or IP layer (ICMP Ping is here)

2) Link (MAC)

Moves ***frames*** within a local area network (switching)
Each computer identified by a MAC address on its Network Interface Card (NIC)
Also called Layer 2, MAC layer, Data Link layer, or Ethernet layer

1) Physical

How data is physically transmitted

- Transmitter converts logical 1 and 0 ***bits*** to electrical/light pulses or phase/amplitude of radio frequency (RF) and sends down wire or over air
- Receiver converts electrical/light or RF back to logical 1 and 0 bits

https://www.ibm.com/think/topics/osi-model

Agenda



1. IP addresses: NAT
2. Messages
3. DNS
4. Email
5. Web
6. Exercises

IP addresses have a consistent format

Example: **129.170.224.138**

IP addresses are fixed length of 32 bits (4 bytes)

Each IP has four octets, where octets range from 0 to 255 (8 bits)

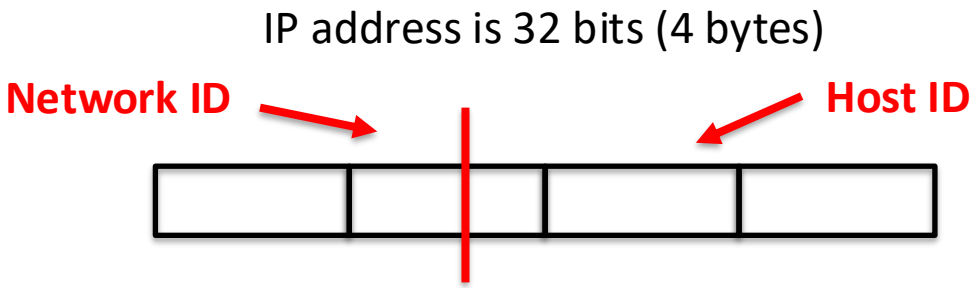
Address are written with a dot in between octets

Hierarchical, read IP address from left to right

- Left portion gives the network number
- Right portion is the specific host on that network

IP addresses contain the network and device ID

Classless Interdomain Routing (CIDR, 1993+)



Split wherever we'd like

- Specify how many bits make up the network ID, the rest of the address specifies a device on that network
- 128.224.115.0/24 means left 24 bits are network ID
  Here 32-24=8 bits (range 0 to 255) are used to identify a device on the 128.224.115.0 network
- 0 and 255 are reserved, so devices can really only be 1 to 254
- Split does not have to occur on 8-bit boundaries
- 128.224.0.0/12 means left most 12 bits are the network ID
- /12 is called the subnet mask

Some IP addresses are reserved for private use

Private IP address

10.0.0.0/8 (=$2^{32-8=24}$ devices)

172.16.0.0/12 ($2^{32-12=20}$ devices)

192.168.0.0/16 ($2^{32-16=16}$ devices)

Lookback address

127.0.0.0/8

Commonly used 127.0.0.1 (localhost)

Private addresses are used to provide a network for devices in home or office

Private addresses are non-routable on Internet (multiple homes or offices may use the same address inside)

ISP issues one IP address for the home, and the home uses private addresses on the inside

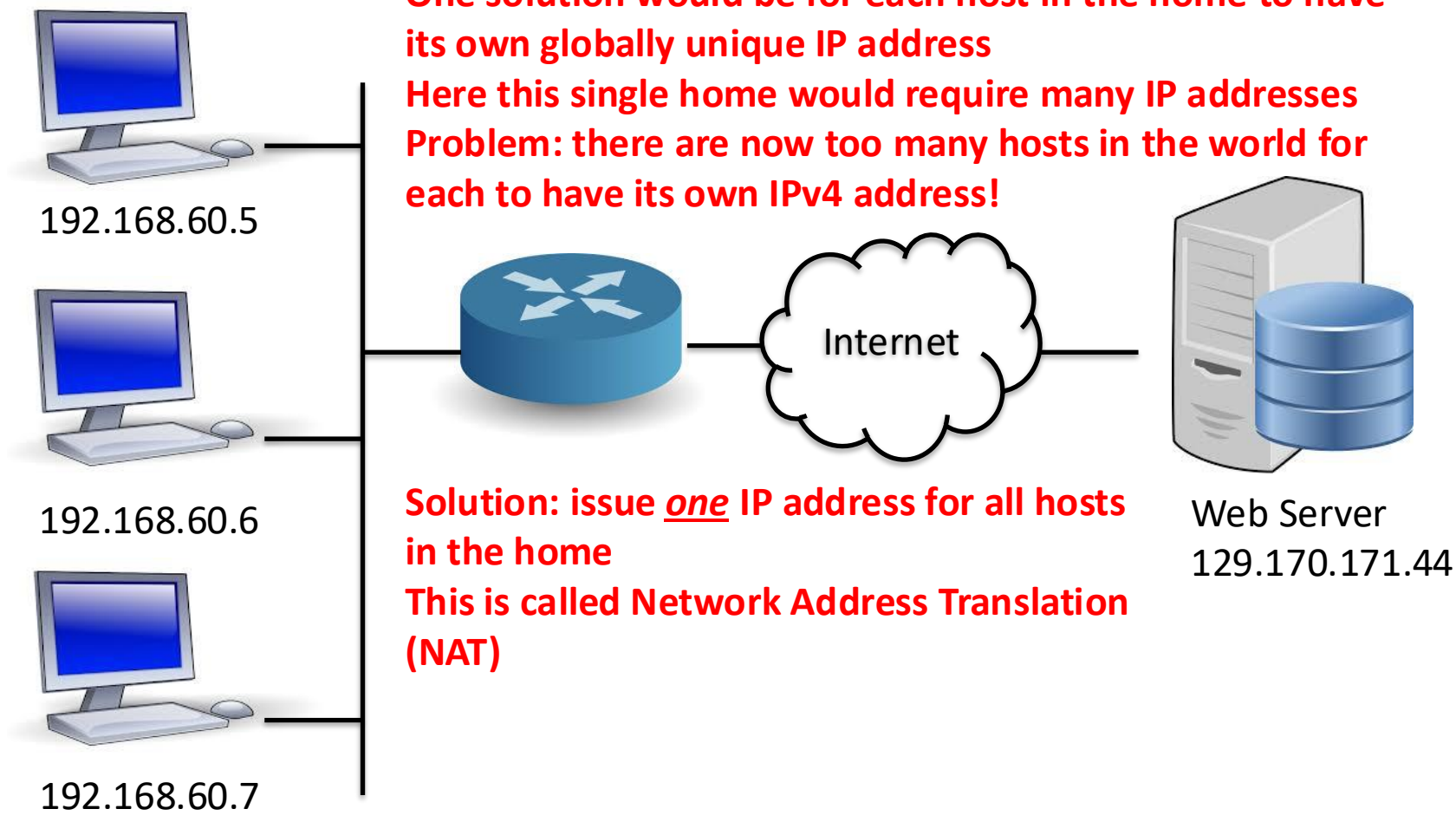
Why?

We ran out of IPv4 addresses!

Only have 255^4 possible addresses ≅ 4 billion addresses

This arrangement is called Network Address Translation (NAT)

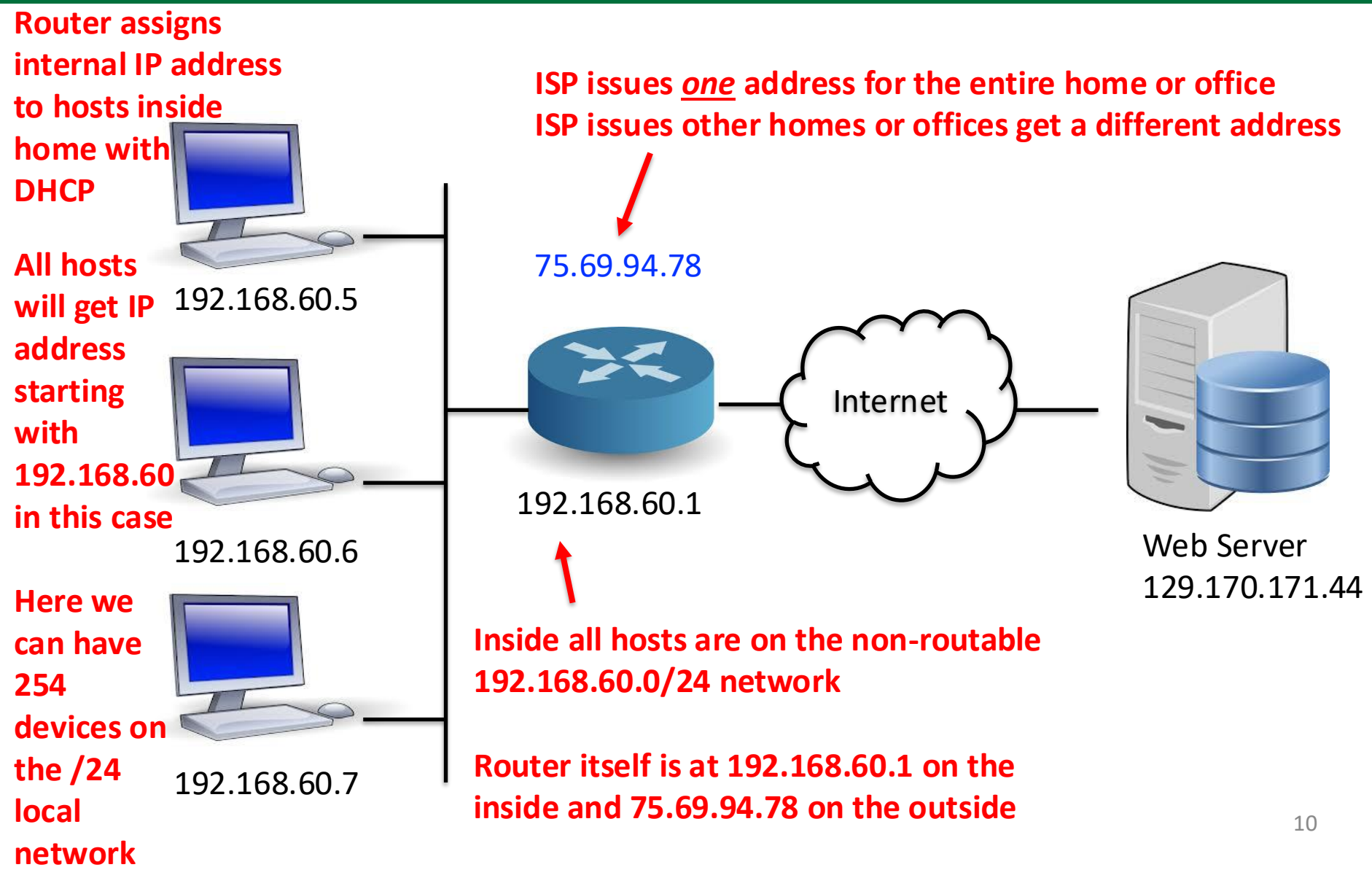
Consider a home network where there are many devices



Home network has many hosts (here only 3 shown, suppose there are more devices not shown)
One solution would be for each host in the home to have its own globally unique IP address
Here this single home would require many IP addresses
Problem: there are now too many hosts in the world for each to have its own IPv4 address!

Solution: issue *one* IP address for all hosts in the home
This is called Network Address Translation (NAT)

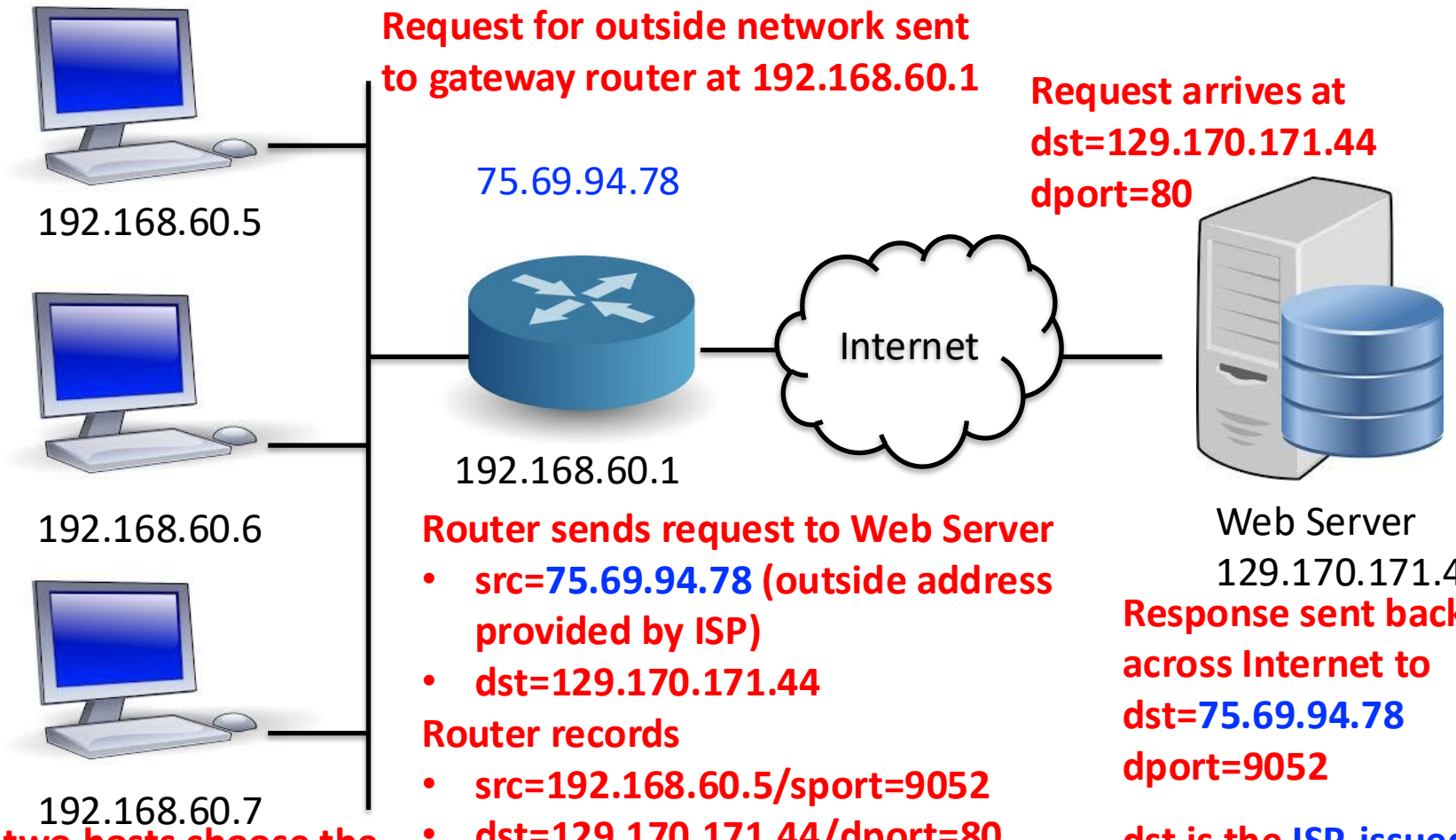
Network Address Translation (NAT) handles translation between inside and outside



Network Address Translation (NAT) handles translation between inside and outside

Hosts make requests to a server on Internet

src=192.168.60.5 on sport=9052 requests web page from Web Server at dst=129.170.171.44 dport=80



Router sends request to Web Server

- **src=75.69.94.78 (outside address provided by ISP)**
- **dst=129.170.171.44**

Router records

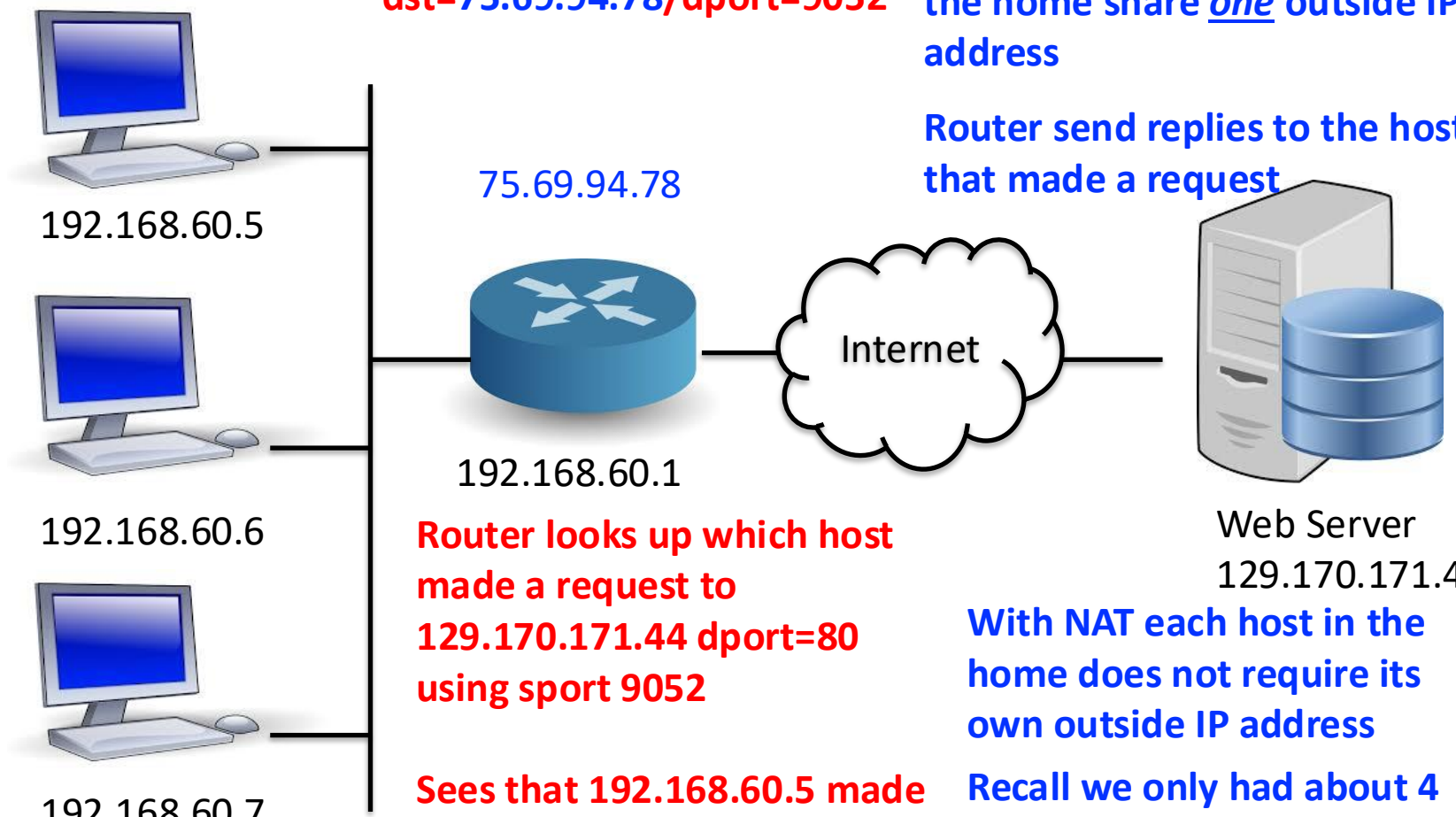
- **src=192.168.60.5/sport=9052**
- **dst=129.170.171.44/dport=80**

Response sent back across Internet to dst=75.69.94.78 dport=9052

dst is the ISP-issued address

What if two hosts choose the same sport to same dst?

Network Address Translation (NAT) handles translation between inside and outside



Agenda

1. IP addresses: NAT
2. Messages
3. DNS
4. Email
5. Web
6. Exercises

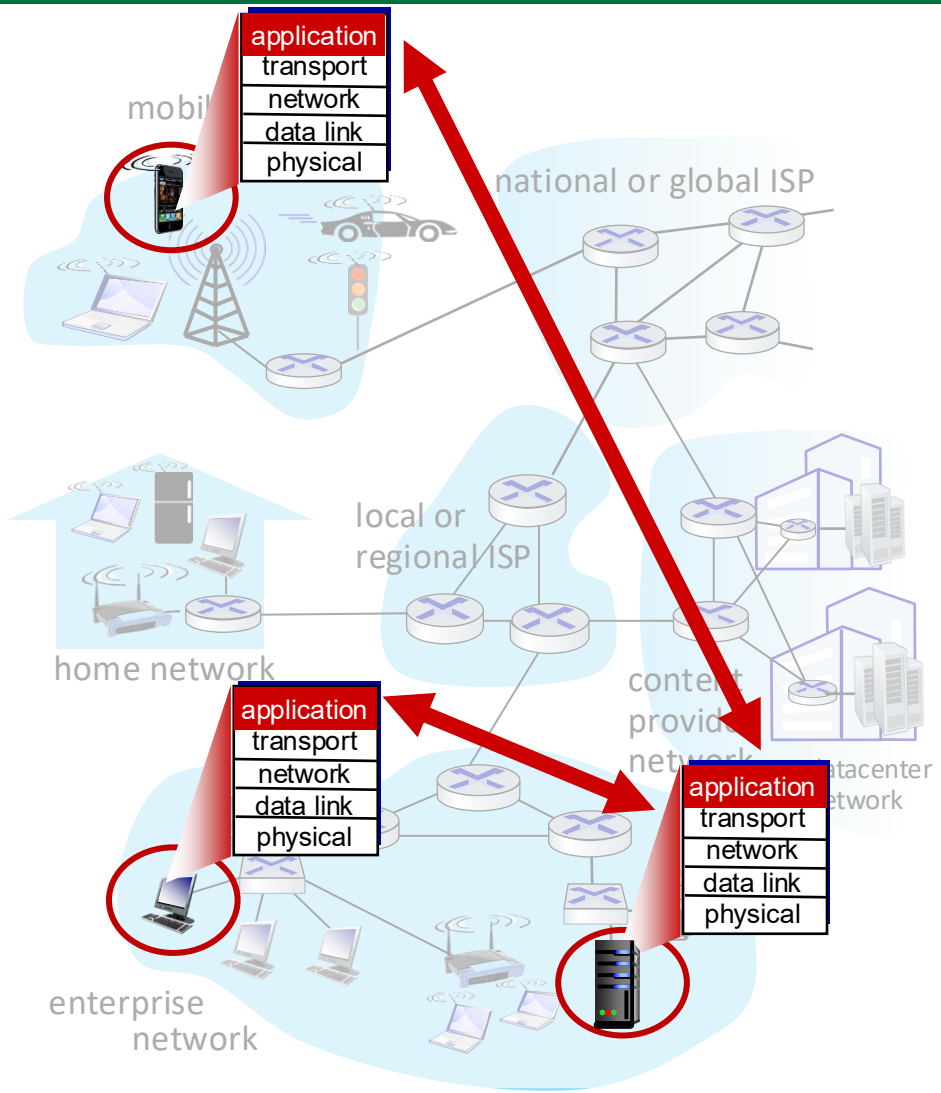
Applications can define data formats and (mostly) need not worry about network

Write programs that:

- Run on (different) end systems
- Communicate over network
- e.g., web server software communicates with browser software

No need to write software for network-core devices

- Network-core devices do not run user applications
- Applications on end systems only allows for rapid app development, propagation



We will assume a client/server paradigm

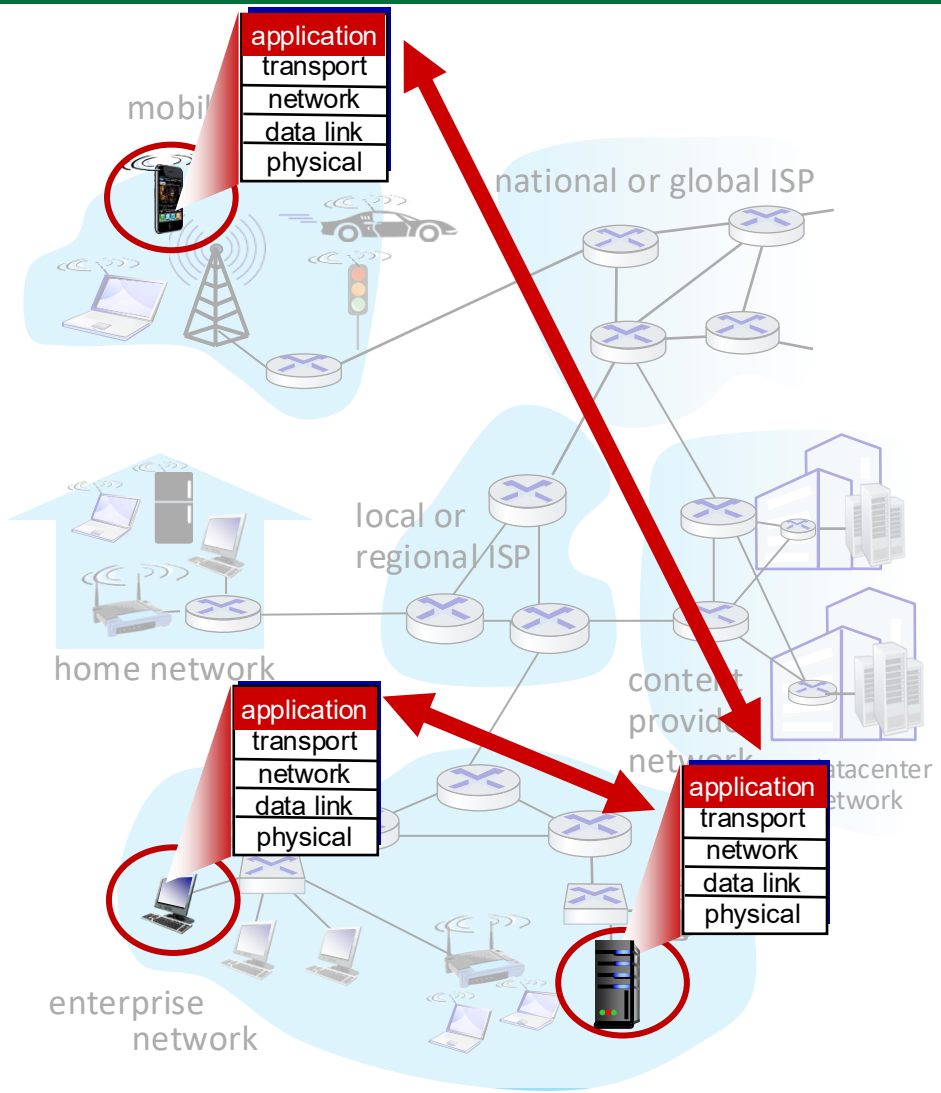
Server:

- Always-on host
- Permanent IP address
- Often in data centers

Clients:

- Contact and communicate with server
- May be intermittently connected
- May have dynamic IP addresses
- Do *not* communicate directly with each other

- Examples: HTTP, IMAP, FTP



Adapted from Kurose and Ross: Computer Networking: A Top-Down Approach

Process will typically run on separate clients that communicate over the network

Process: program running within a host

- Within same host, two processes communicate using inter-process communication (defined by OS)
- Processes in different hosts communicate by exchanging messages over the network

Clients, Servers

Client process: process that *initiates* communication

Server process: process that *waits* to be contacted

Adapted from Kurose and Ross: Computer Networking: A Top-Down Approach

Let’s define our own message protocol for the Blackjack card game

Blackjack

Goal: Get as close to 21 points **without going over**

Points are calculated as follows:

- Each numbered card (2 through 10) has the number of points on the card (suit does not matter). For example, the Seven of Diamonds counts for 7 points
- Each "face card" (Jack, Queen, King) counts for 10 points
- Each Ace counts for either 1 or 11 points (whichever is best for the player, the choice of either 1 or 11 can change as the game progresses)

Let's define our own Layer 7 message protocol for the Blackjack card game

Blackjack
Goal: Get as close to 21 points **without going over**



Process:

1. Players join table
2. Game begins when 1 to 6 players have joined
3. Dealer deals from standard 52-card deck:
    1. Each player bets
    2. Each player gets one card face up
    3. Dealer gets one card face down (unknown to players and dealer)
    4. Each player gets one card face up
    5. Dealer gets one card face up
4. Each player, in turn, decides to:
    - HIT – take another card
    - STAND – stop taking cards
    - Repeat until STAND or BUST (points > 21)
    - If BUST, player loses bet
5. Dealer HITs until it has at least 17 points, then STANDs (or BUSTs)
6. Outcome
    - If Dealer BUSTs, non busted players win
    - If Player has more points than dealer (but less than 22), player wins
    - If Player and Dealer have same points, it is a push
    - Else Player looses

Assume:

- Dealer in on one host
- Players are each on separate hosts

What could a protocol for this game look like?

NOTE: ignoring splits and double downs

Let's define our own Layer 7 message protocol for the Blackjack card game

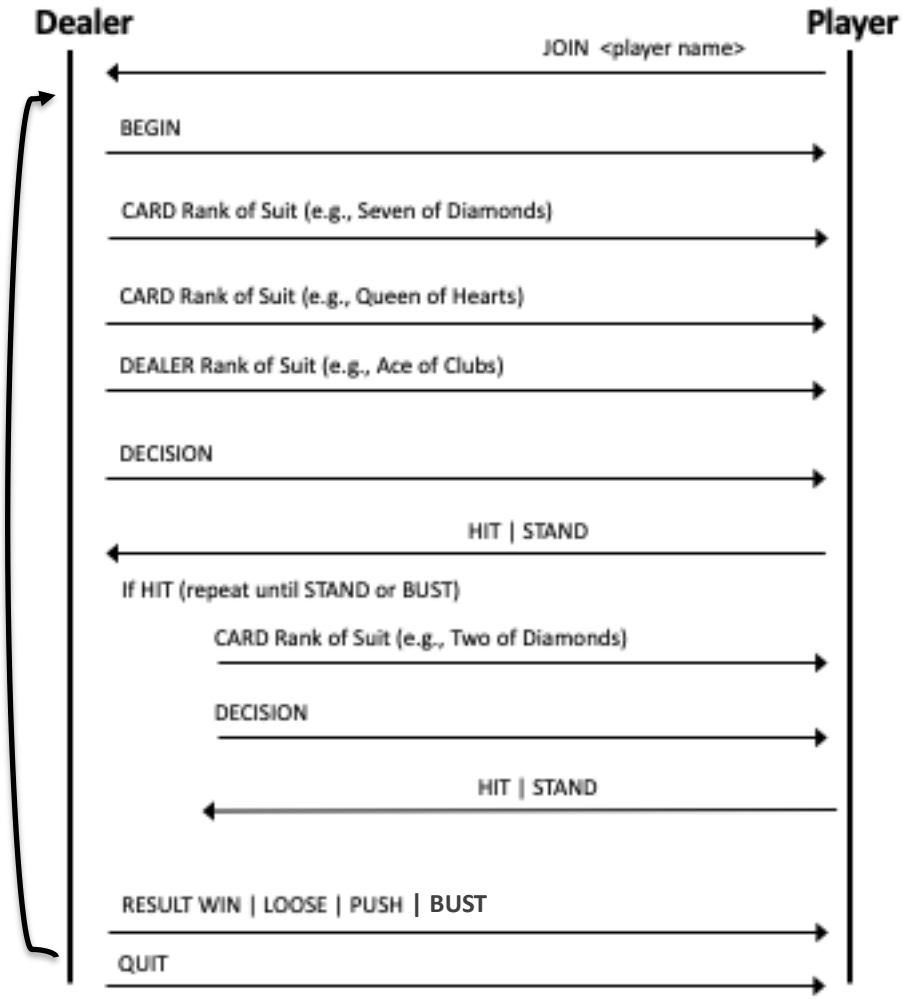
Blackjack

Also, players never bet

If you design your own protocol, think carefully!

This protocol never shows the player what cards the Dealer got

Just assumes Dealer will give the honest result



Example messages:

JOIN Bob\r\n

CARD Seven of Diamonds\r\n

CARD Queen of Hearts\r\n

DEALER Ace of Clubs\r\n

HIT\r\n

CARD Two of Diamonds\r\n

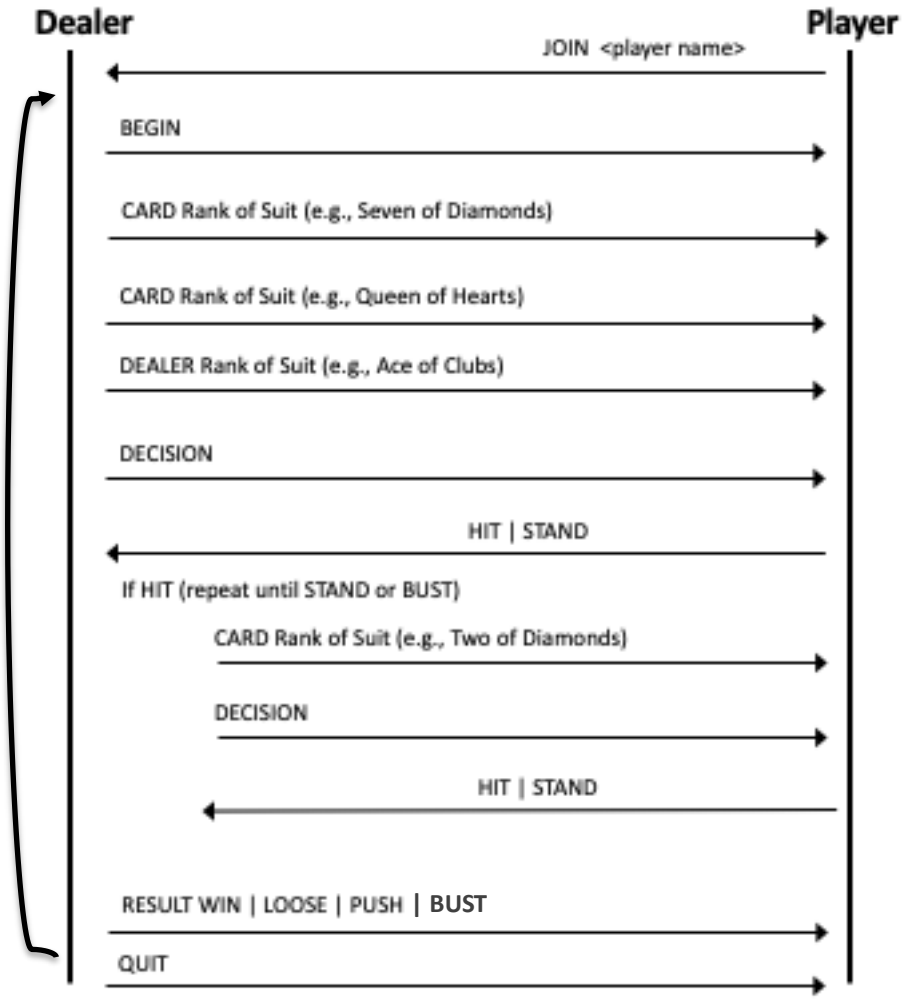
STAND\r\n

RESULT WIN\r\n

Layer 7 messages are often formatted using JSON, but do not have to be

Be careful to define what to do if things go wrong

Blackjack



Or JOIN but never get a BEGIN

Might want a timeout

What if HIT, but get nothing back?

Example messages:

JOIN Bob\r\n

CARD Seven of Diamonds\r\n

CARD Queen of Hearts\r\n

DEALER Ace of Clubs\r\n

HIT\r\n

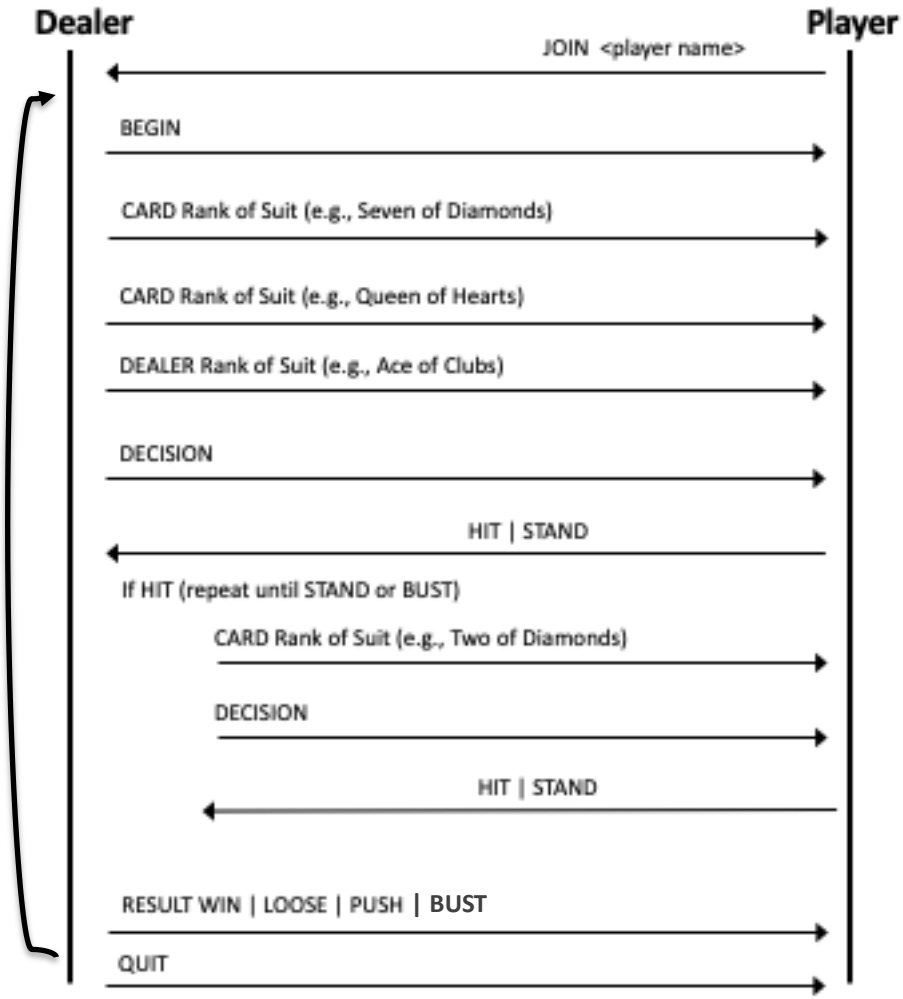
CARD Two of Diamonds\r\n

STAND\r\n

RESULT WIN\r\n

JSON Version is more structured than just text

Blackjack



Example messages:

{“type”:”JOIN”, “player”:“Bob”}\r\n

{“type”:”BEGIN”}\r\n

{“type”:“CARD”, “card”:“7D”}\r\n

{“type”:“CARD”, “card”:“QH”}\r\n

{“type”:“DEALER”, “card”: “AC”}\r\n

{“type”:”DECISION”}\r\n

{“type”:“ACTION”, ”value”:“HIT”}\r\n

{“type”:“CARD”, “card”:“2D”}\r\n

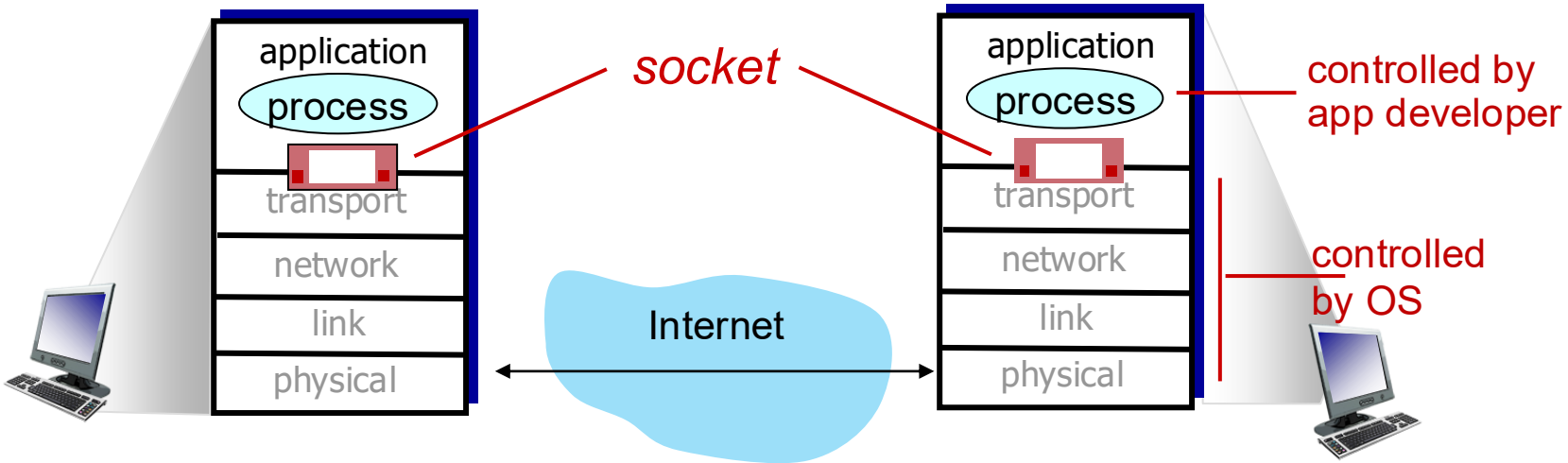
{“type”:”DECISION”}\r\n

{“type”:“ACTION”, “value”:“STAND”}\r\n

{“type”:“RESULT”, “value”:“WIN”}\r\n

Processes use sockets to communicate over the network

- Process sends/receives messages to/from its socket
- Socket analogous to door
  - Sending process sends message out the door
  - Sending process relies on transport infrastructure on other side of door to deliver message to socket at receiving process
  - Two sockets involved: one on each side



Adapted from Kurose and Ross: Computer Networking: A Top-Down Approach

Processes use IP address and port to identify each other

- To receive messages, process must have *identifier*
- Host device has unique 32-bit IP address
- *Q:* Does IP address of host on which process runs suffice for identifying the process?
  - *A:* No, *many* processes can be running on same host

- *Identifier* includes both IP address and port numbers associated with process on host
- Example port numbers:
  - HTTP server: 80
  - Mail server: 25
- To send HTTP message to dartmouth.edu web server:
  - IP address: 129.170.224.138
  - Port number: 80
- More shortly…

Adapted from Kurose and Ross: Computer Networking: A Top-Down Approach

Transport Layer Security (TLS) is often used to secure messages

Vanilla TCP & UDP sockets:

- No encryption
- Cleartext passwords sent into socket traverse Internet in cleartext (!)

Transport Layer Security (TLS)

- Provides encrypted TCP connections
- Data integrity
- End-point authentication

TLS implemented in application layer

- Apps use TLS libraries, that use TCP in turn
- Cleartext sent into socket traverses Internet *encrypted*
- Encrypted ciphertext decrypted at the receiver
- We will cover TLS in some detail in the Security portion of this class

Adapted from Kurose and Ross: Computer Networking: A Top-Down Approach

Agenda

1. IP addresses: NAT
2. Messages
3. DNS
4. Email
5. Web
6. Exercises

Humans are bad at remembering numbers, but we can remember hostnames





Humans are bad at remembering numbers, but we can remember hostnames



Bob's number is forgotten almost as soon as he says it

Hi, I'm
172.217.165.142
Nice to meet you



Humans are bad at remembering numbers, but we can remember hostnames





Humans are bad at remembering numbers, but we can remember hostnames



Alice can (likely) remember Bob's name



Humans are bad at remembering numbers, but we can remember hostnames



Likewise, Alice can remember:

- google.com
- amazon.com
- dartmouth.edu

But not:

- 172.217.165.142 (google.com)
- 205.251.242.103 (amazon.com)
- 129.170.224.138 (dartmouth.edu)

Domain Name System (DNS) is a way to convert amazon.com into 205.251.242.103

DNS works by passing messages that follow a pre-defined protocol



DNS is a distributed database critical for the Internet to function

Domain Name System (DNS)

- All computers on the Internet have an IP Address
- Given a hostname (google.com), DNS returns an IP Address (172.217.165.142) for that host
- Can also return mail server and can load balance over multiple servers using round robin approach
- DNS is an Application Layer (Layer 7) protocol: hosts, DNS servers communicate to resolve names (address/name translation)
  - Uses UDP on port 53
- Distributed database implemented in hierarchy of many name servers

Why distributed, why not one centralized DNS?

- **Centralized would be a single point of failure**
- **Traffic volume (Comcast alone does 600B queries/day, Akamai does 2.2 trillion DNS queries/day)**
- **One location would be distant for some places on Earth**
- **How to do maintenance?**

DNS provides many services, but must run quickly!

Humongous distributed database:

- Billions of records, each simple



Handles *trillions* of queries/day:

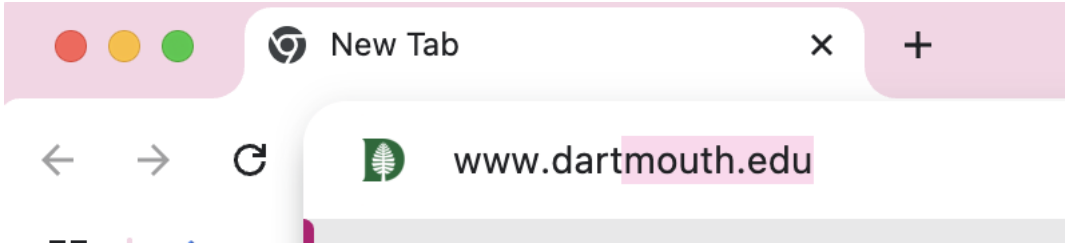
- *Many* more reads than writes
- *Performance matters:* almost every Internet transaction interacts with DNS
- Milliseconds count!
- Speed is why DNS uses UDP not TCP

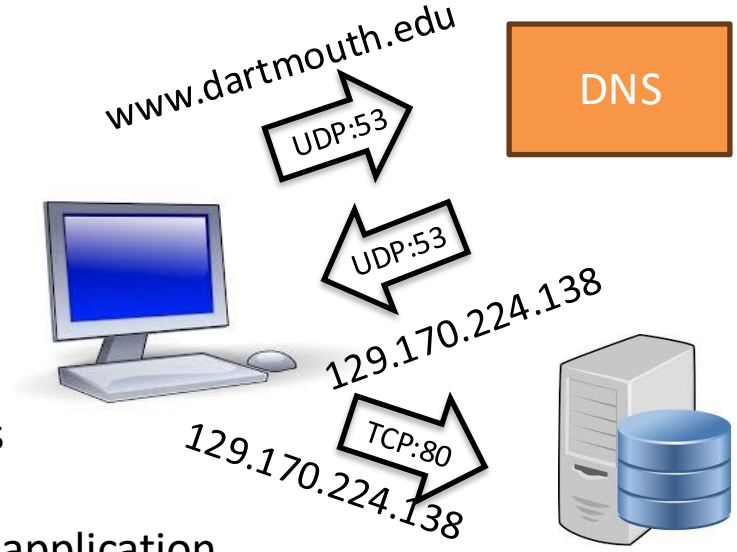
Organizationally, physically decentralized:

- Millions of different organizations responsible for their records

"Bulletproof": reliability, security

Browser example: type dartmouth.edu into URL





Type www.dartmouth.edu into browser
Need to convert www.dartmouth.edu into IP address

1. The user machine runs the client side of the DNS application
2. The browser extracts the hostname, www.dartmouth.edu, from the URL and passes the hostname to the client side of the DNS application
3. The DNS client sends a query containing the hostname to a DNS server
4. The DNS client eventually receives a reply, which includes the IP address for the hostname
5. Once the browser receives the IP address from DNS, it can initiate a TCP connection to the HTTP server process located at port 80 at that IP address

On Linux and Macs, can call *gethostbyname(<hostname>)* to resolve a hostname to an IP address
***gethostbyname* calls DNS server to resolve name**

How did host get address of DNS server?
Typically from DHCP

DNS provides additional services

Host aliasing

Server relay1.west-coast.enterprise.com might have aliases

- enterprise.com
- www.enterprise.com

DNS can return the same IP for any of those names

relay1-west-coast-enterprise.com is said to be the canonical name

Email server aliasing

Mail server might be relay1.west-coast.mail.enterprise.com but email address is bob@enterprise.com

DNS can provide MX record with name of email server (rely1-west-coast-mail.enterprise.com) for enterprise.com

Load distribution

An organization might have several web servers

DNS returns queries in round-robin fashion

Adapted from Kurose and Ross: Computer Networking: A Top-Down Approach

DNS provides additional services

Host aliasing

Server relay1.west-coast.enterprise.com might have aliases

- enterprise.com
- www.enterprise.com

DNS can return the same IP for any of those names

relay1-west-coast-enterprise.com is said to be the canonical name

Email server aliasing

Mail server might be relay1.west-coast.mail.enterprise.com but email address is bob@enterprise.com

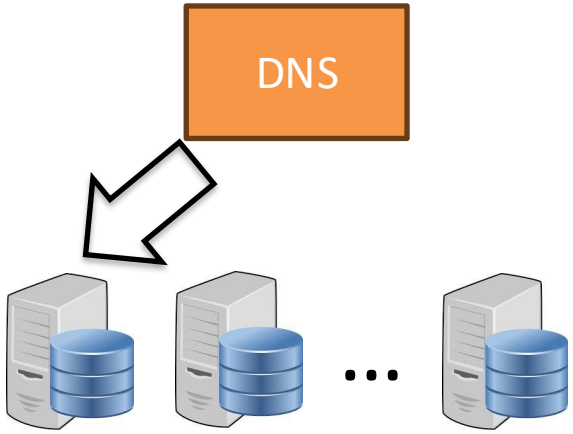
DNS can provide MX record with name of email server (rely1-west-coast-mail.enterprise.com) for enterprise.com

Load distribution

An organization might have several web servers

DNS returns queries in round-robin fashion

First DNS query sent to first web server



Adapted from Kurose and Ross: Computer Networking: A Top-Down Approach

DNS provides additional services

Host aliasing

Server relay1.west-coast.enterprise.com might have aliases

- enterprise.com
- www.enterprise.com

DNS can return the same IP for any of those names

relay1-west-coast-enterprise.com is said to be the canonical name

Email server aliasing

Mail server might be relay1.west-coast.mail.enterprise.com but email address is bob@enterprise.com

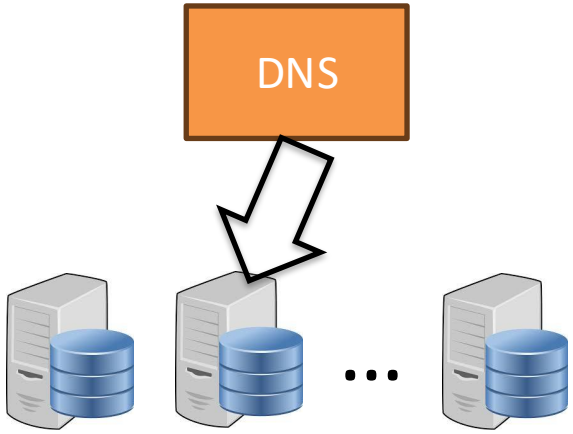
DNS can provide MX record with name of email server (rely1-west-coast-mail.enterprise.com) for enterprise.com

Load distribution

An organization might have several web servers

DNS returns queries in round-robin fashion

Second DNS query sent to second web server



Adapted from Kurose and Ross: Computer Networking: A Top-Down Approach

DNS provides additional services

Host aliasing

Server relay1.west-coast.enterprise.com might have aliases

- enterprise.com
- www.enterprise.com

DNS can return the same IP for any of those names

relay1-west-coast-enterprise.com is said to be the canonical name

Email server aliasing

Mail server might be relay1.west-coast.mail.enterprise.com but email address is bob@enterprise.com

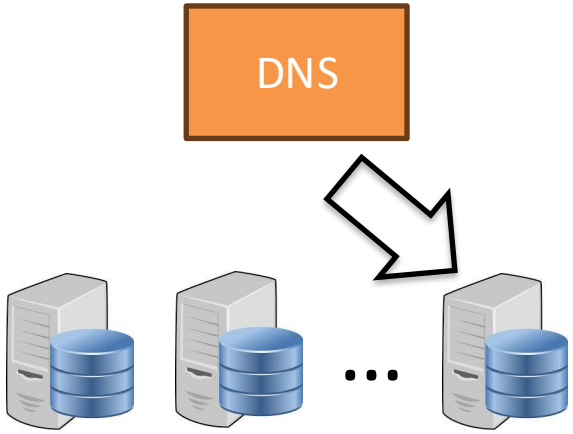
DNS can provide MX record with name of email server (rely1-west-coast-mail.enterprise.com) for enterprise.com

Load distribution

An organization might have several web servers

DNS returns queries in round-robin fashion

n^{th} DNS query sent to n^{th} web server



Adapted from Kurose and Ross: Computer Networking: A Top-Down Approach

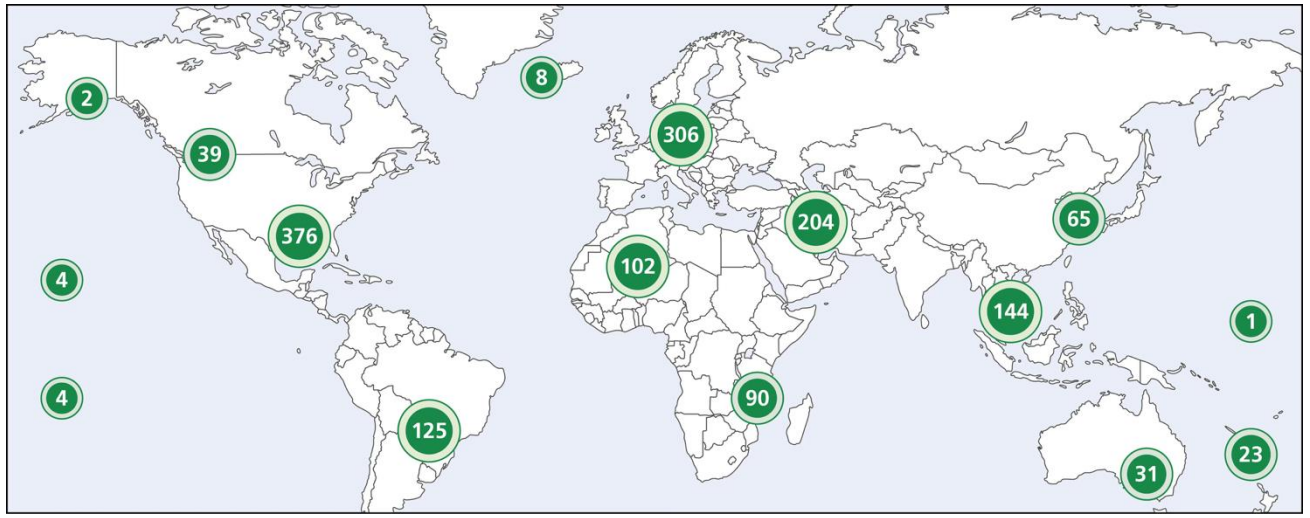
DNS is a hierarchical system starting with Root Servers



Root DNS Servers

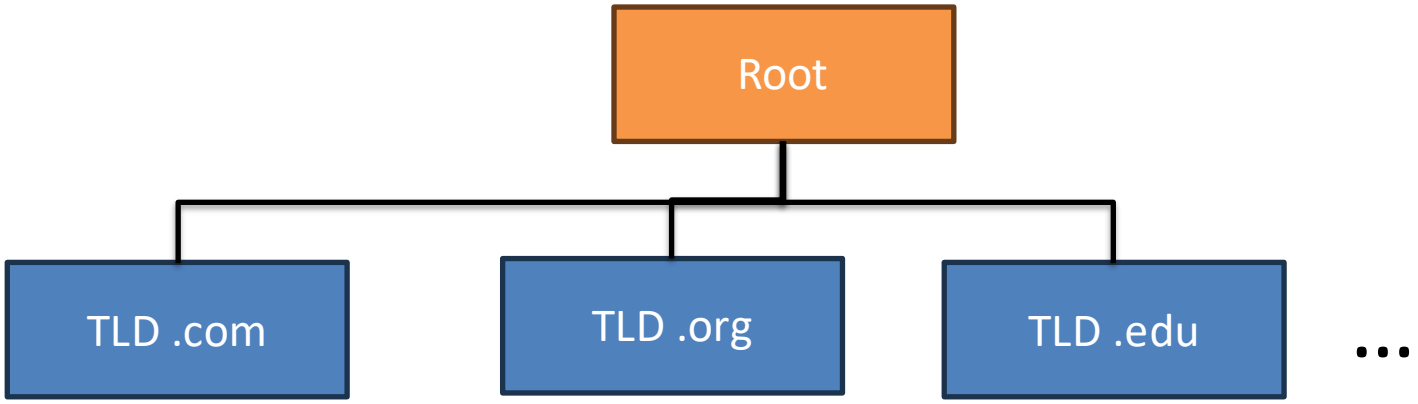
- There are about 2,000 Root Servers deployed worldwide
- Each a copy of one of 13 servers managed by Internet Assigned Numbers Authority (IANA)
- Root servers provide IP addresses of Top-Level Domain (TLD) Servers
- Root servers are official, contact-of-last-resort by name servers that can not resolve name

Root server locations



Adapted from Kurose and Ross: Computer Networking: A Top-Down Approach

Top Level Domain (TLD) servers are below Root severs



Top-Level Domain (TLD) DNS

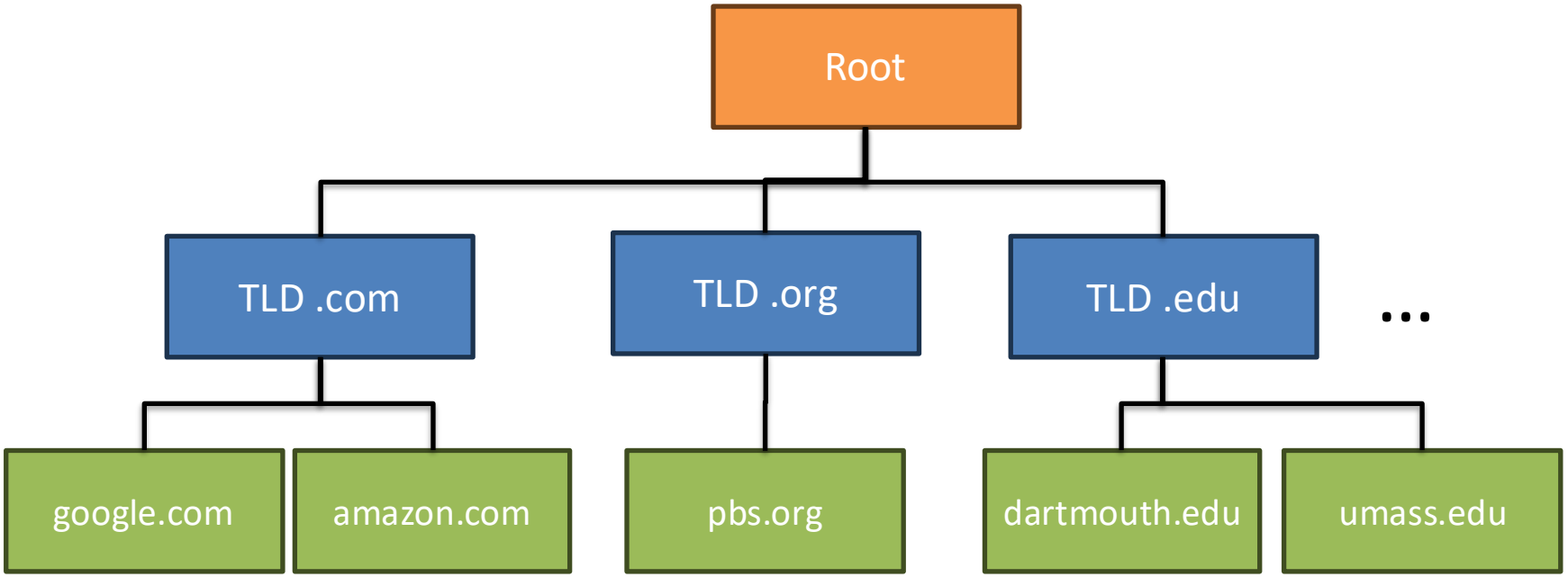
- Includes .com, .org, .net, .edu, .gov, and others
- Country TLDs such as uk, fr, ca, jp, and others
- For each TLD there is TLD server (or server cluster)
- Designated companies manage TLD DNS Servers
  - Verisign Global Registry Services maintains the TLD servers for .com
  - Educause maintains the TLD servers for .edu

TLD servers provide the IP addresses for authoritative DNS servers

Internet started with 6 TLDs
.com .org .gov .mil .net .edu
Now has more than 1500!

Adapted from Kurose and Ross: Computer Networking: A Top-Down Approach

Authoritative DNS servers are below TLD servers and provide definitive IP addresses



Authoritative DNS Servers

- Every organization with publicly accessible hosts (such as Web servers and mail servers) must provide records that map the names of those hosts to IP addresses
- An organization's authoritative DNS server houses these mapping records
- An organization can implement its own authoritative DNS server to hold these records; alternatively, the organization can pay to have these records stored by an ISP
- Most universities and large companies maintain their own primary and secondary (backup) authoritative DNS servers

Adapted from Kurose and Ross: Computer Networking: A Top-Down Approach

Each ISP maintains local name servers

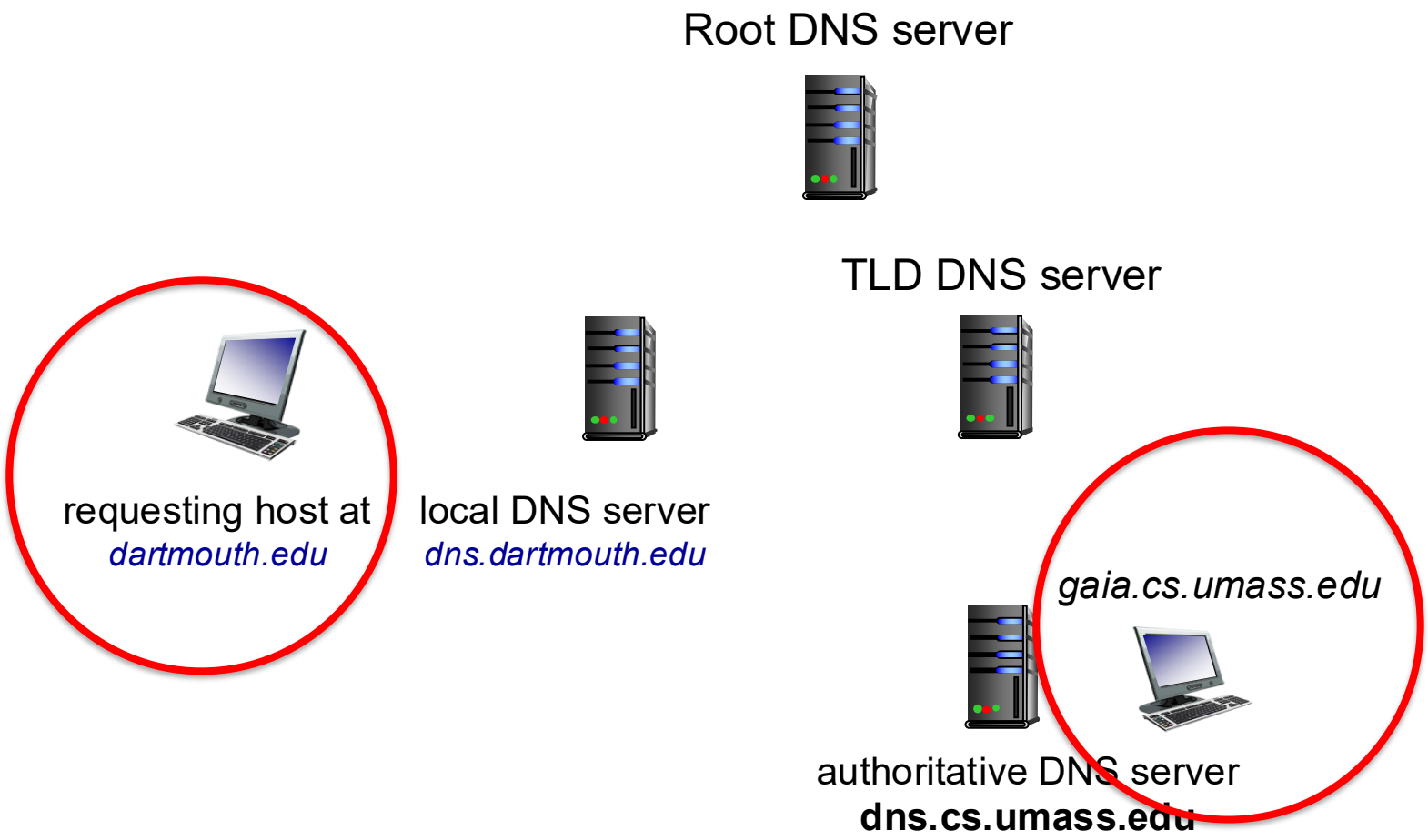
Local Name Servers

- Local name servers do not strictly belong to hierarchy
- Each ISP (residential ISP, company, university) has at least one name server
  - Also called “default name server”
- When host makes a DNS query, query is sent to its local DNS server
  - Local DNS server has a local cache of recent name-to-address translation pairs (but may be out of date!)
  - Acts as proxy, forwards query into DNS hierarchy if:
    - Local name server does not know the IP address of host
    - Cached name-to-address-pair is out of date

Adapted from Kurose and Ross: Computer Networking: A Top-Down Approach

DNS name resolution: iterated query

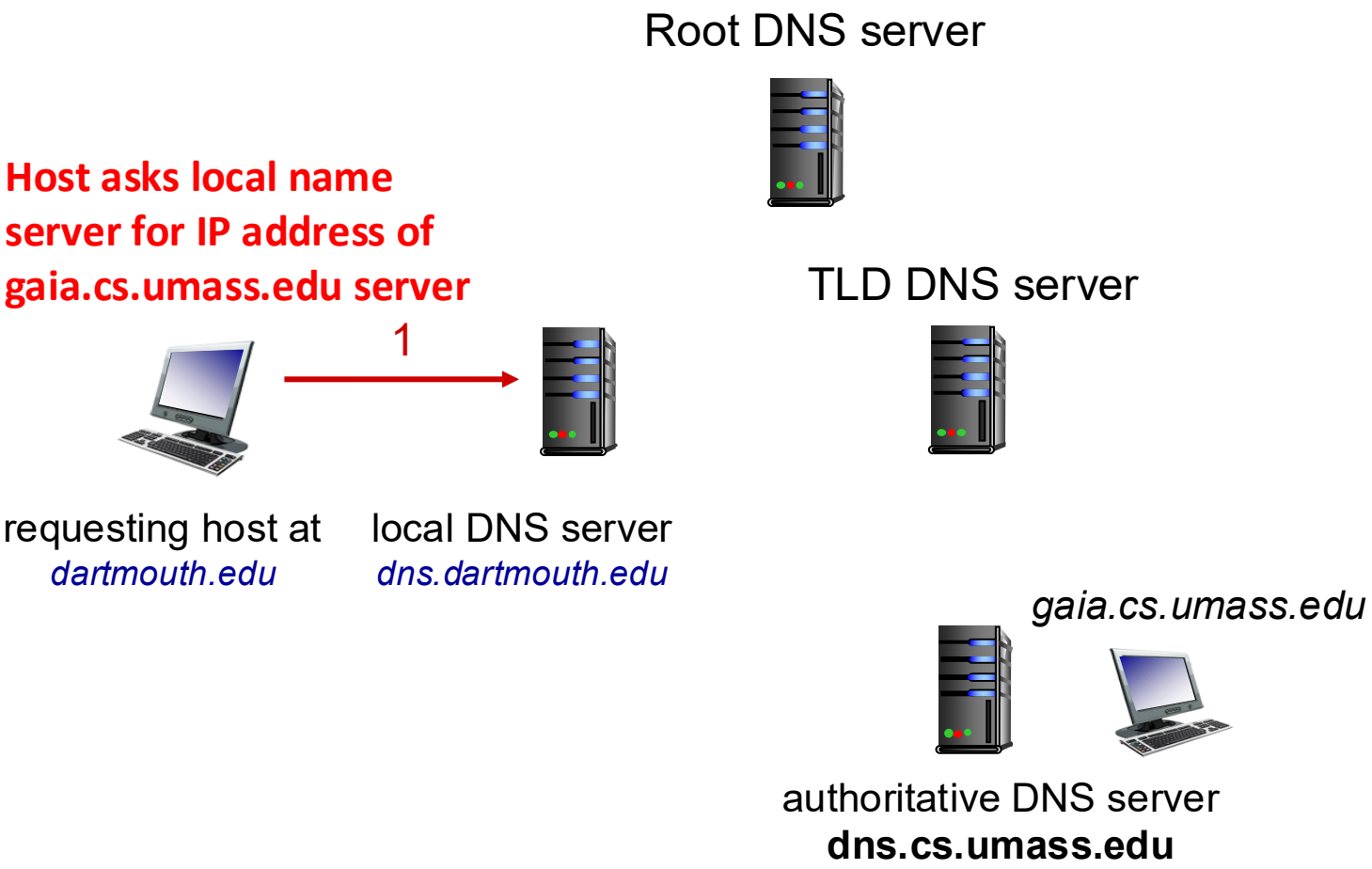
Example: host at *dartmouth.edu* wants IP address for *gaia.cs.umass.edu*



Adapted from Kurose and Ross: Computer Networking: A Top-Down Approach

DNS name resolution: iterated query

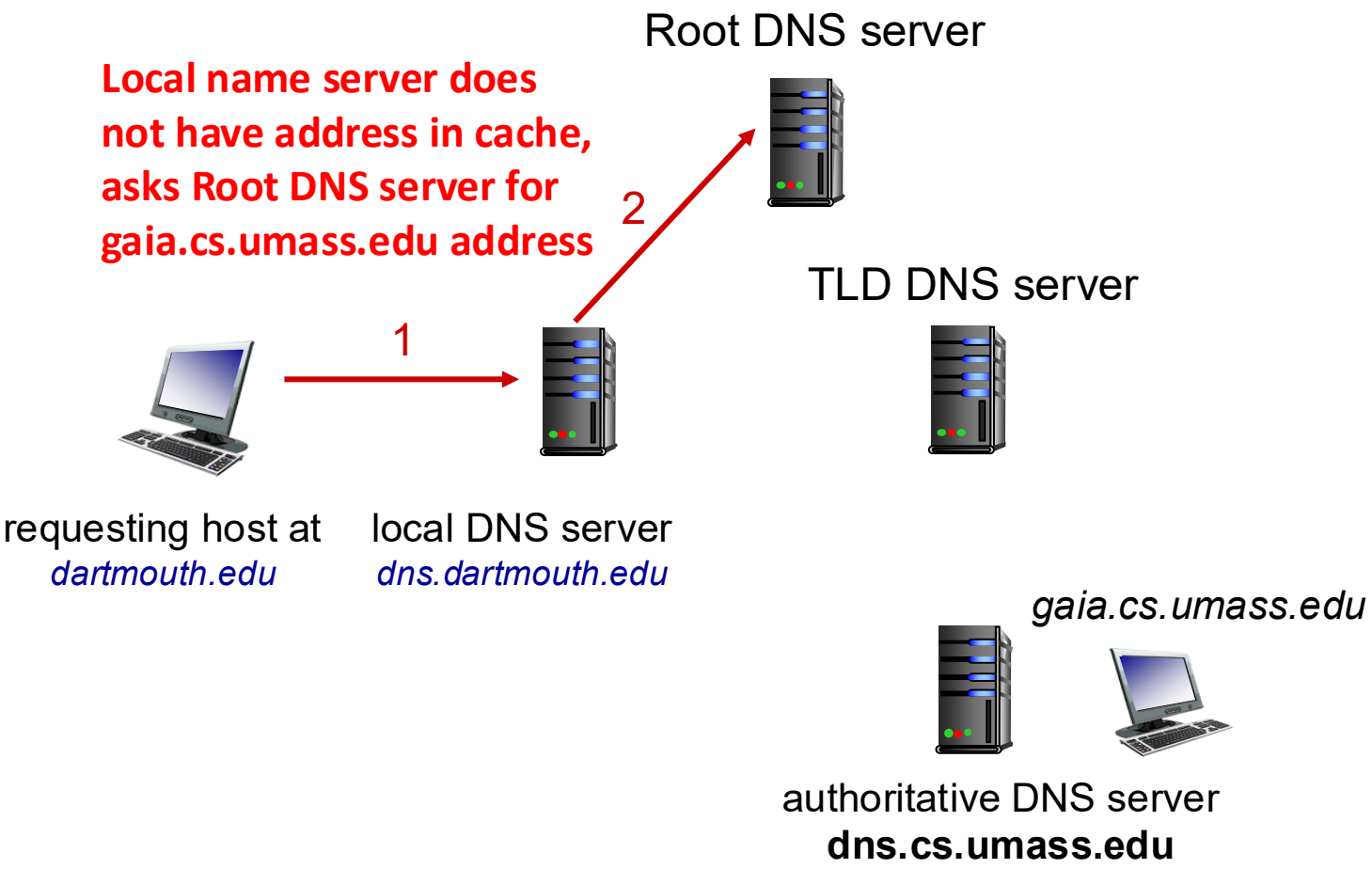
Example: host at *dartmouth.edu* wants IP address for *gaia.cs.umass.edu*



Adapted from Kurose and Ross: Computer Networking: A Top-Down Approach

DNS name resolution: iterated query

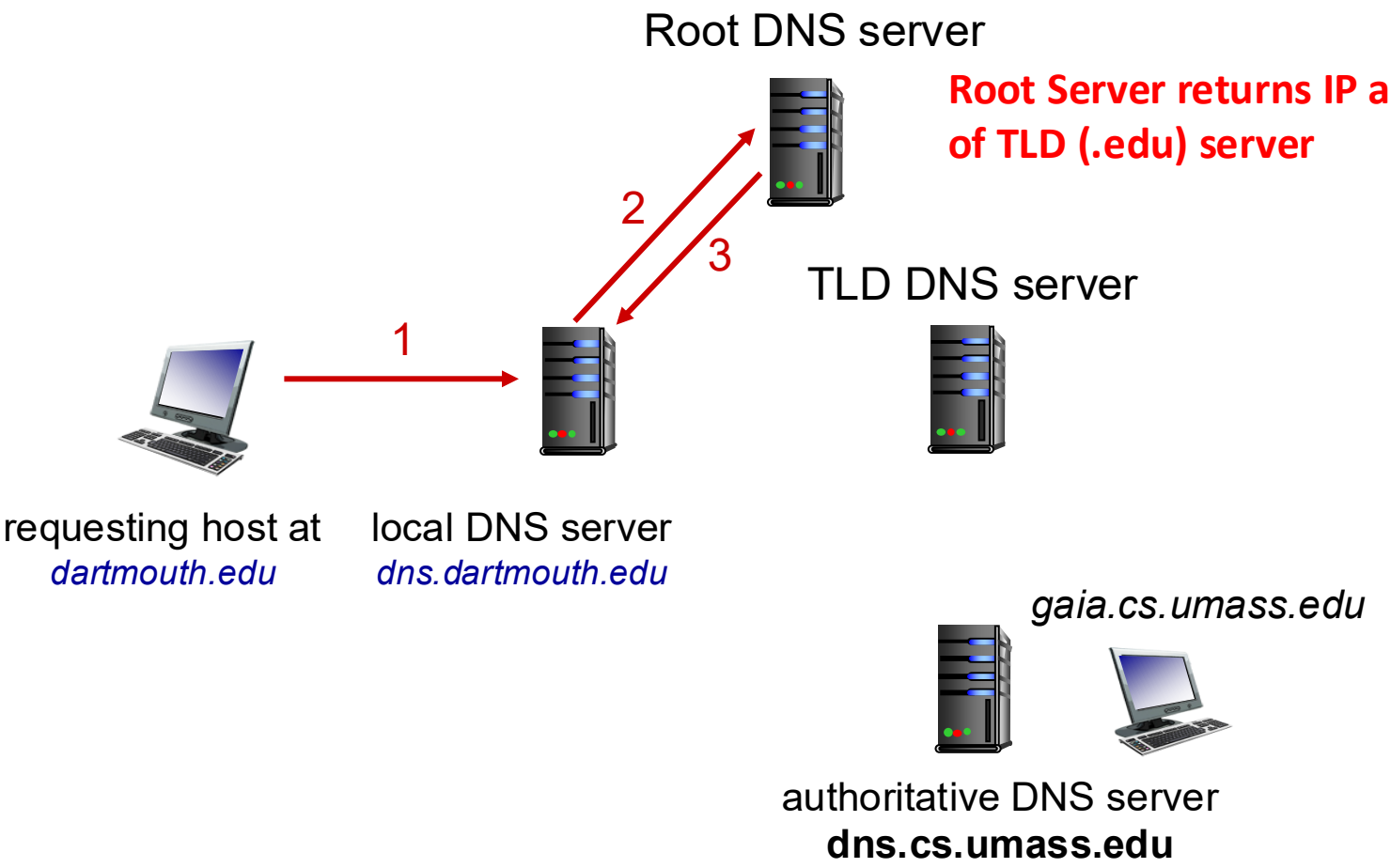
Example: host at *dartmouth.edu* wants IP address for *gaia.cs.umass.edu*



Adapted from Kurose and Ross: Computer Networking: A Top-Down Approach

DNS name resolution: iterated query

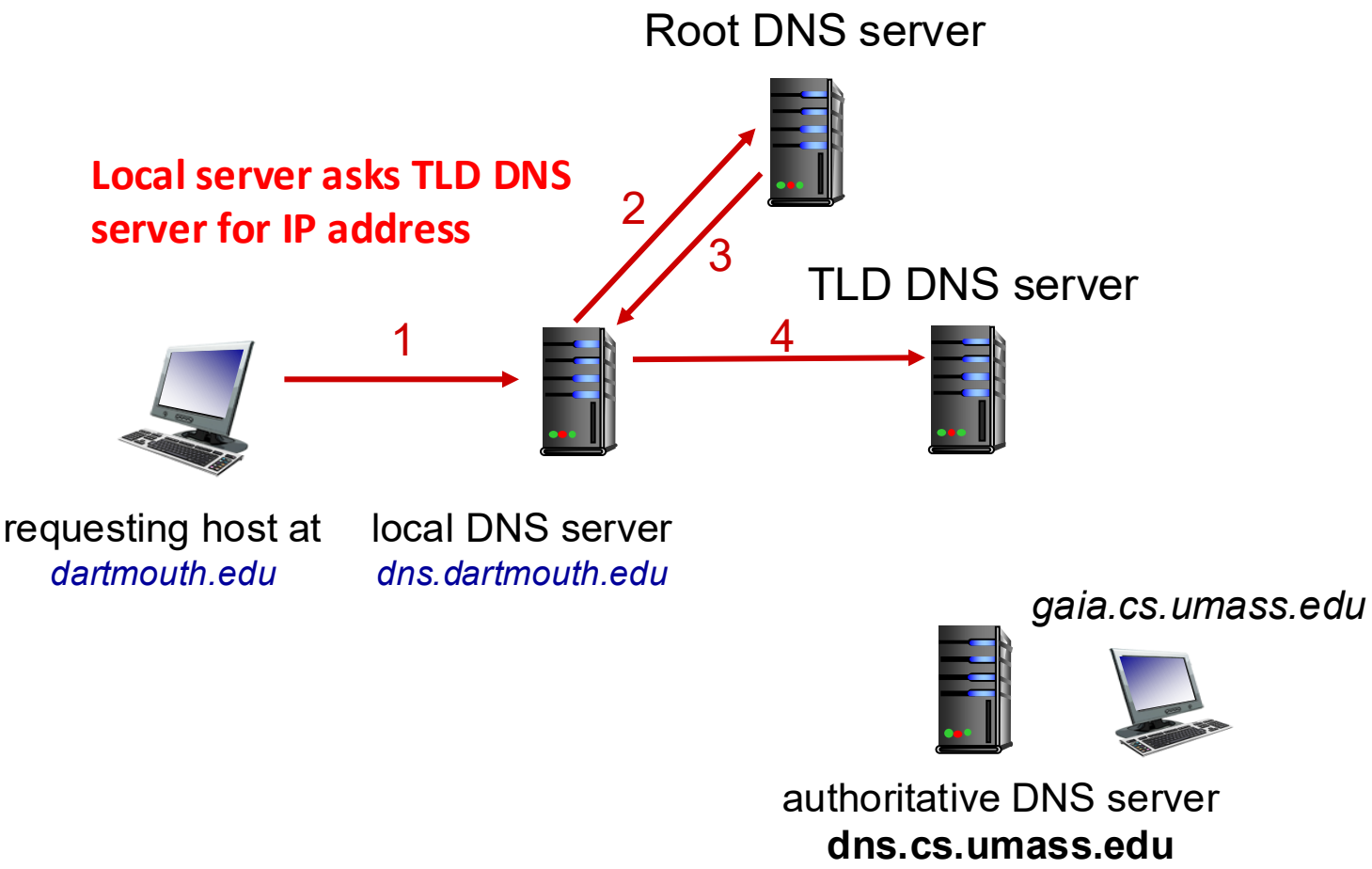
Example: host at *dartmouth.edu* wants IP address for *gaia.cs.umass.edu*



Adapted from Kurose and Ross: Computer Networking: A Top-Down Approach

DNS name resolution: iterated query

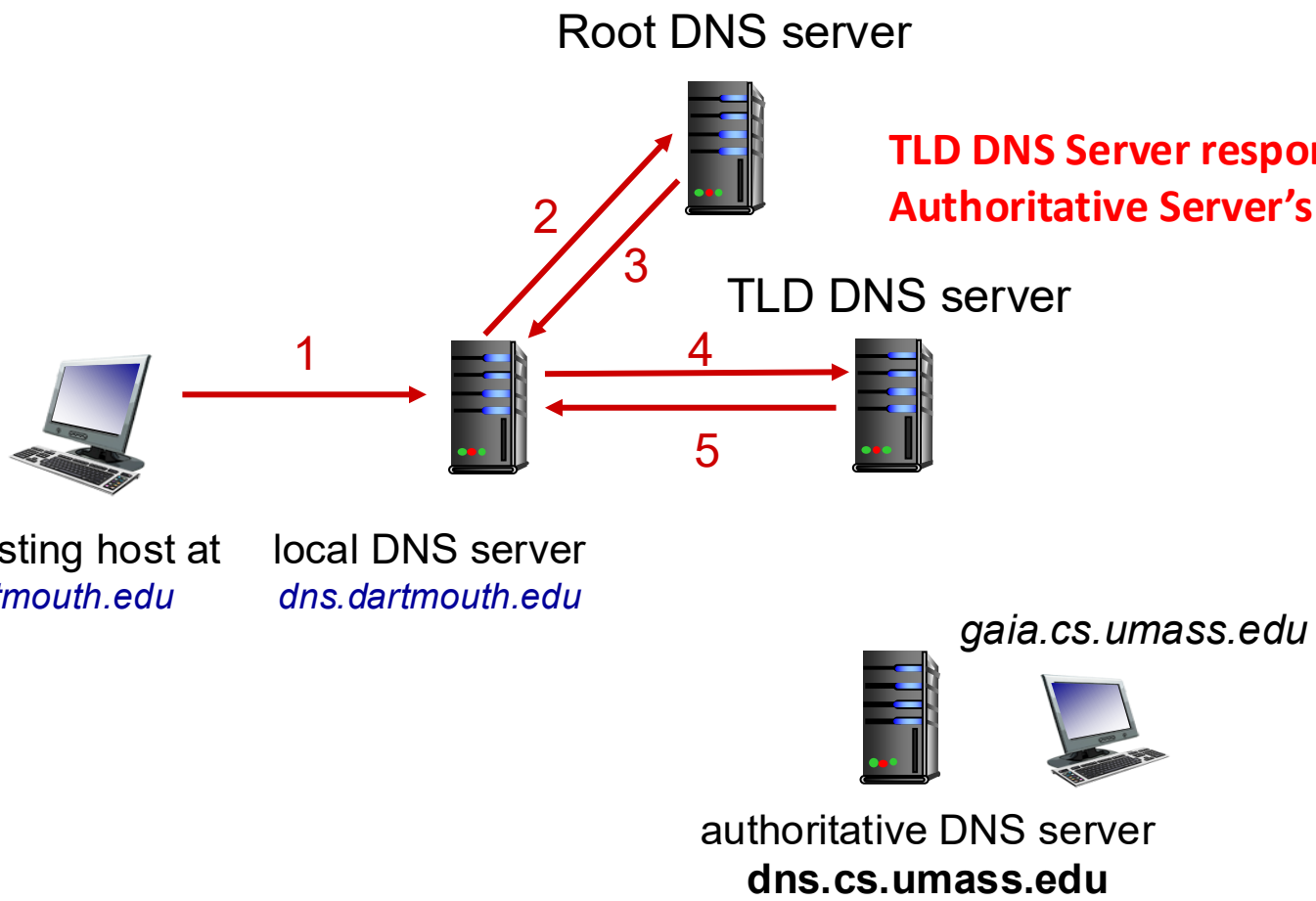
Example: host at *dartmouth.edu* wants IP address for *gaia.cs.umass.edu*



Adapted from Kurose and Ross: Computer Networking: A Top-Down Approach

DNS name resolution: iterated query

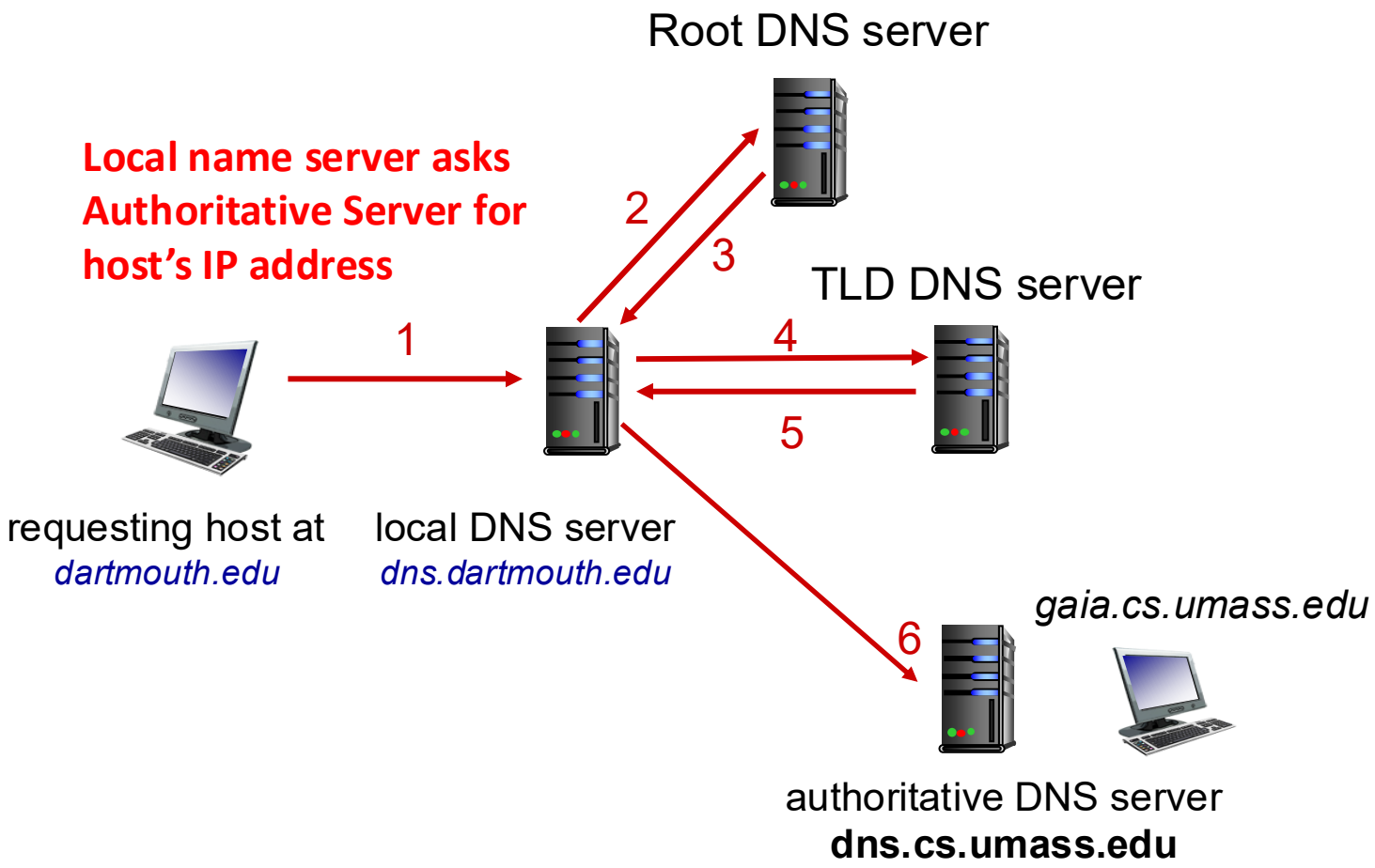
Example: host at *dartmouth.edu* wants IP address for *gaia.cs.umass.edu*



Adapted from Kurose and Ross: Computer Networking: A Top-Down Approach

DNS name resolution: iterated query

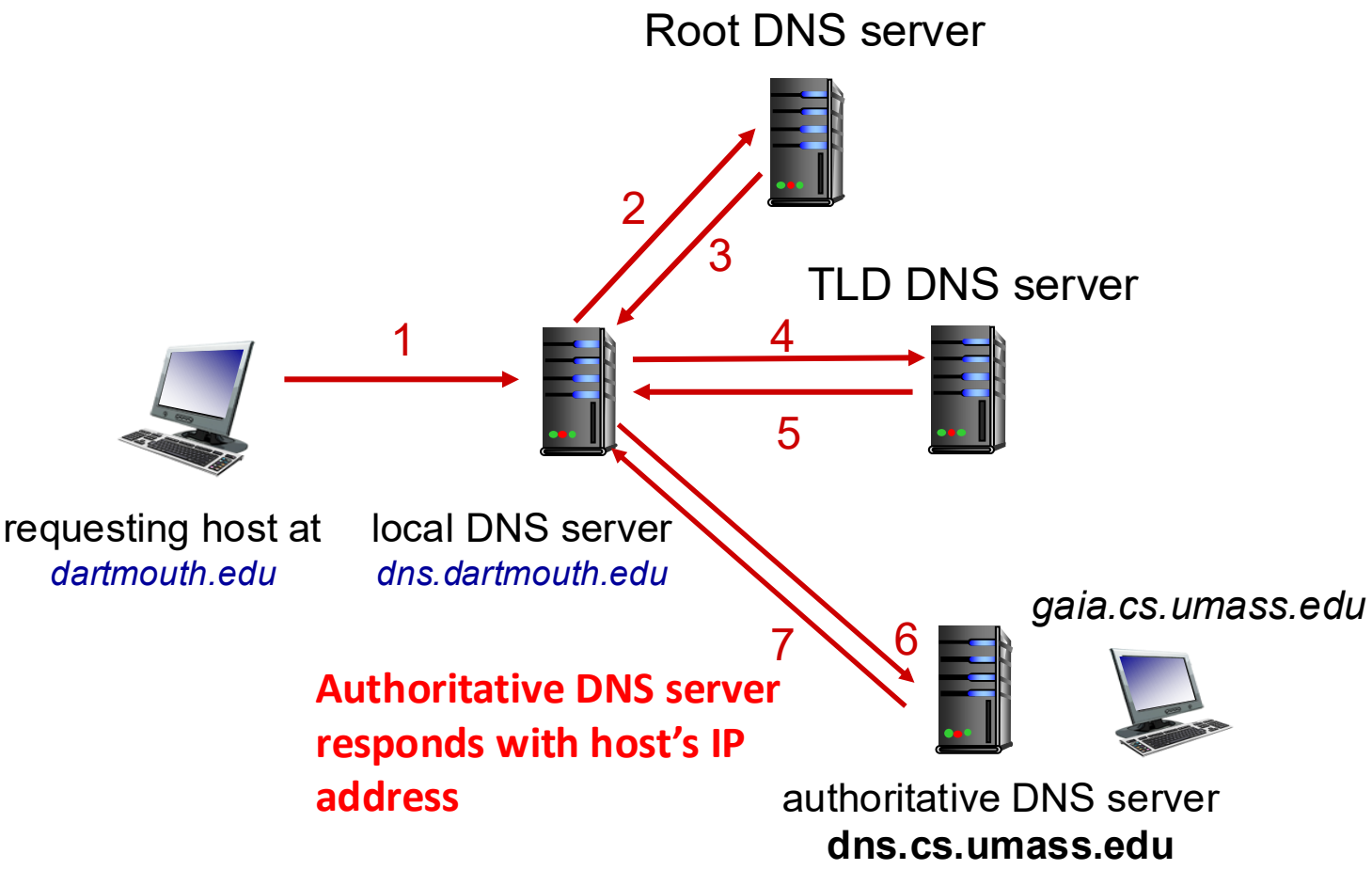
Example: host at *dartmouth.edu* wants IP address for *gaia.cs.umass.edu*



Adapted from Kurose and Ross: Computer Networking: A Top-Down Approach

DNS name resolution: iterated query

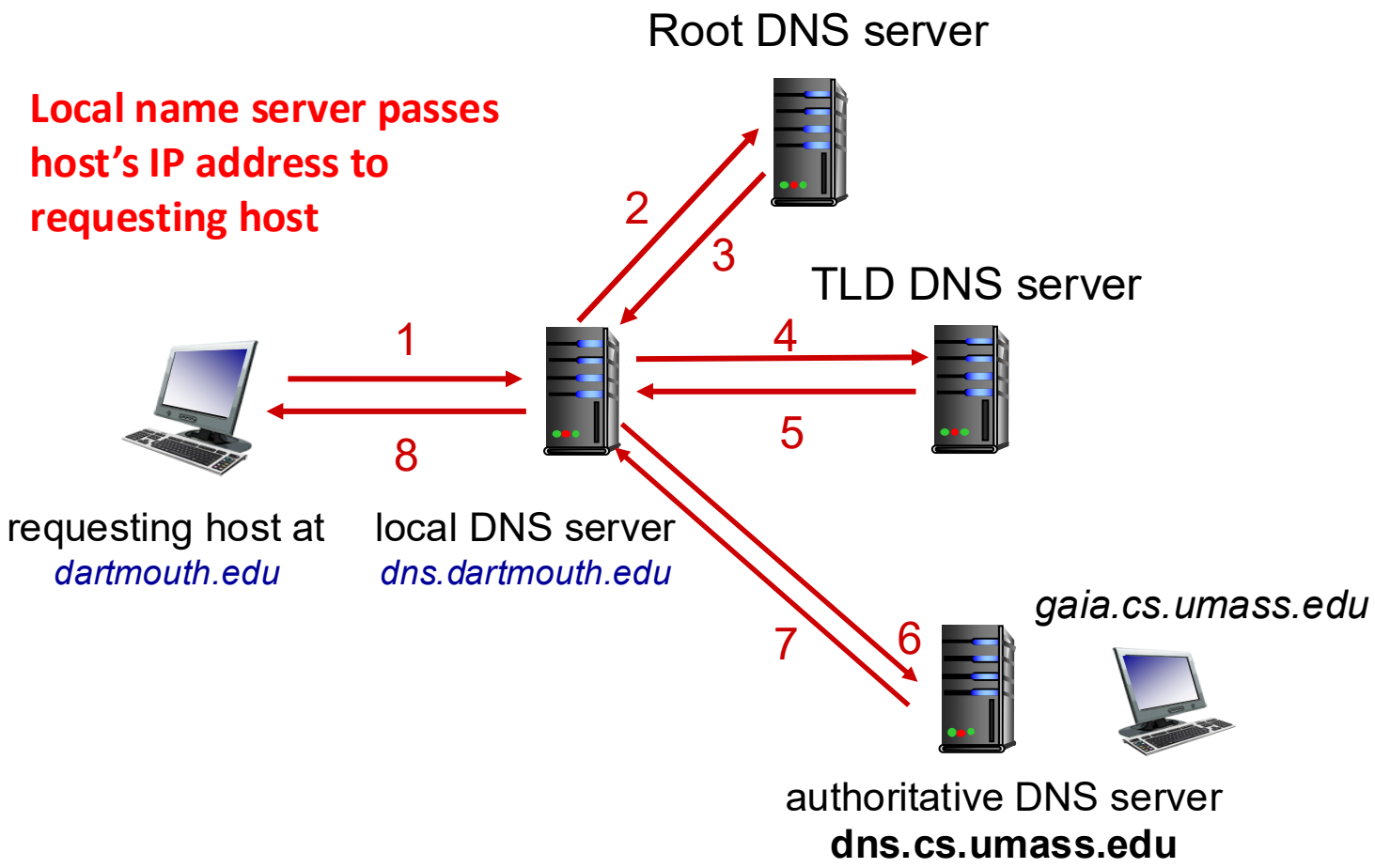
Example: host at *dartmouth.edu* wants IP address for *gaia.cs.umass.edu*



Adapted from Kurose and Ross: Computer Networking: A Top-Down Approach

DNS name resolution: iterated query

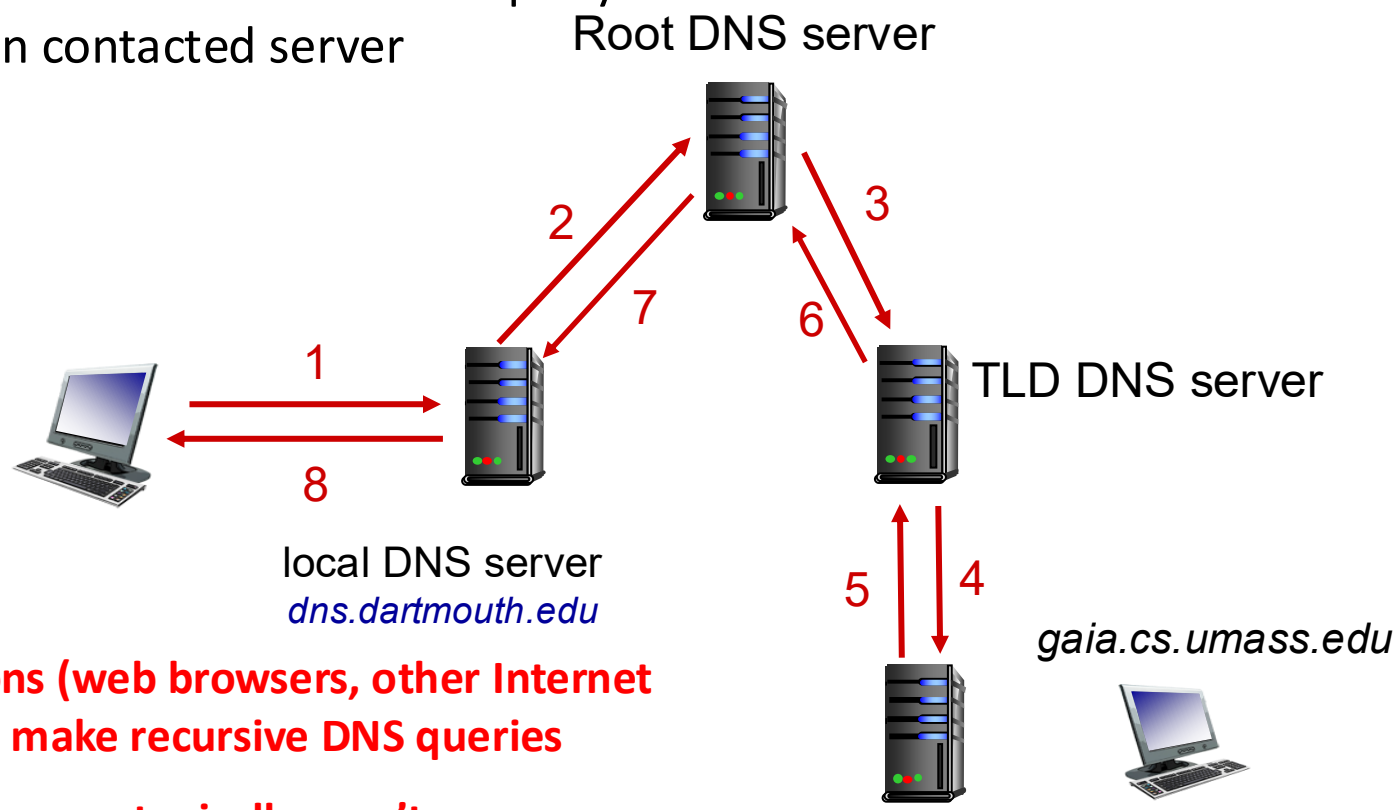
Example: host at *dartmouth.edu* wants IP address for *gaia.cs.umass.edu*



Adapted from Kurose and Ross: Computer Networking: A Top-Down Approach

DNS name resolution: recursive query

Example: host at dartmouth.edu wants IP address for gaia.cs.umass.edu
Local name server asks for ***recursive*** query
Puts burden on contacted server



User applications (web browsers, other Internet apps) normally make recursive DNS queries

Root and TLD servers typically won't answer recursive queries!

CloudFlare (1.1.1.1) or Google (8.8.8.8) or ISP DNS servers will

Adapted from Kurose and Ross: Computer Networking: A Top-Down Approach

DNS servers cache name/IP address pairs to reduce the number of looks up

- Once any name server learns mapping, it *caches* the mapping
  - Cache entries timeout (disappear) after some time (TTL = Time To Live)
  - TTL set by server admin, typically minutes to hours
  - A common value is 86,400 seconds, which is 24 hours

- Local DNS name server might give answer right away if it has unexpired cached mapping (even if the local DNS server is not the authoritative DNS server)
  - TLD servers typically cached in local name servers
  - Thus, Root name servers not often visited

- Cached entries may be *out-of-date* (best-effort name-to-address translation!)
  - If name host changes IP address, may not be known Internet-wide until all TTLs expire!

- Update/notify mechanisms proposed IETF standard
  - RFC 2136

Adapted from Kurose and Ross: Computer Networking: A Top-Down Approach

DNS record types



A Record

Maps hostname → IPv4 address

example.com → 93.184.216.34

Normally for web server

Use AAAA record for IPv6



MX Record

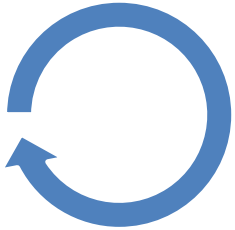
Specifies mail servers for domain

example.com → 10 mail1.example.com

→ 20 mail2.example.com

Try mail1 first (priority 10), then mail2 (priority 20)

Lookup IP address with A record for mail1.example.com



CNAME Record

Canonical (real) name

Alias: one name points to another

www.example.com → example.com

or

www.ibm.com → servereast.backup2.ibm.com



NS Record

Authoritative name server for domain

Delegates domain to name servers

example.com → ns1.example.net

Use nslookup to get DNS information about a domain

```
$ nslookup -type=A dartmouth.edu
Server: 129.170.17.4
Address: 129.170.17.4#53


Name: dartmouth.edu
Address: 129.170.224.138
```



Dartmouth authoritative name server



Web site address (A record)

Use nslookup to get DNS information about a domain

```
$ nslookup -type=A dartmouth.edu
Server: 129.170.17.4
Address: 129.170.17.4#53


Name: dartmouth.edu
Address: 129.170.224.138

$ ping 129.170.224.138
PING 129.170.224.138 (129.170.224.138): 56 data bytes
64 bytes from 129.170.224.138: icmp_seq=0 ttl=59 time=39.202 ms
64 bytes from 129.170.224.138: icmp_seq=1 ttl=59 time=37.357 ms
64 bytes from 129.170.224.138: icmp_seq=2 ttl=59 time=42.621 ms
^C
```



Dartmouth authoritative name server



Web site address (A record)

Confirm server is up

Use nslookup to get DNS information about a domain

```
$ nslookup -type=A dartmouth.edu
Server: 129.170.17.4
Address: 129.170.17.4#53


Name: dartmouth.edu
Address: 129.170.224.138

$ ping 129.170.224.138
PING 129.170.224.138 (129.170.224.138): 56 data bytes
64 bytes from 129.170.224.138: icmp_seq=0 ttl=59 time=39.202 ms
64 bytes from 129.170.224.138: icmp_seq=1 ttl=59 time=37.357 ms
64 bytes from 129.170.224.138: icmp_seq=2 ttl=59 time=42.621 ms
^C
```

Dartmouth authoritative name server

Web site address (A record)

Confirm server is up

Enter address into browser



Use nslookup to get DNS information about a domain

```
$ nslookup -type=A dartmouth.edu
Server: 129.170.17.4
Address: 129.170.17.4#53


Name: dartmouth.edu
Address: 129.170.224.138

$ ping 129.170.224.138
PING 129.170.224.138 (129.170.224.138): 56 data bytes
64 bytes from 129.170.224.138: icmp_seq=0 ttl=59 time
64 bytes from 129.170.224.138: icmp_seq=1 ttl=59 time
64 bytes from 129.170.224.138: icmp_seq=2 ttl=59 time
^C
```

Your web browser does this to find the IP address when you visit a web page using something like dartmouth.edu in the URL

Dartmouth authoritative name server

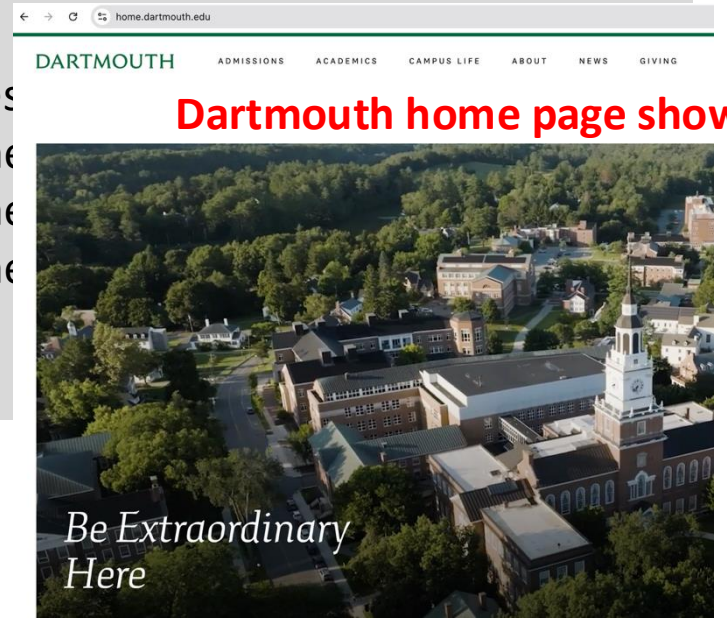
Web site address (A record)

Confirm server is up

Enter address into browser



Dartmouth home page shown



Use nslookup to get DNS information about a domain

```
$ nslookup -type=MX dartmouth.edu
Server: 129.170.17.4
Address: 129.170.17.4#53

dartmouth.edu mail exchanger = 5 dartmouth-edu.mail.protection.outlook.com.
```

Dartmouth authoritative name server

Mail server (MX record)

- **Priority (only one mail server listed, priority = 5)**
- **Mail server name (dartmouth-edu.mail.protection.outlook.com)**

Get A record to get mail server's IP address given server name

Can you get the name of Dartmouth's name server?

Use nslookup

Give it the type of record for a name server

Use nslookup to get DNS information about a domain

```
$ nslookup -type=NS dartmouth.edu
Server: 129.170.17.4
Address: 129.170.17.4#53


dartmouth.edu nameserver = dns2.dartmouth.edu.
dartmouth.edu nameserver = dns1.dartmouth.edu.
dartmouth.edu nameserver = west-campus-dns.dartmouth.edu.
```



Name server (NS record)

- **Dartmouth maintains three name servers**
- **Why?**
- **In case one goes down, DNS is critical for Internet**
- **Call nslookup again and get a round robin response where name servers listed in a different order**

Dig command functions similarly to nslookup

```
$ dig dartmouth.edu
<snip>

;; ANSWER SECTION:
dartmouth.edu. 300 IN A 129.170.224.138


$ dig dartmouth.edu MX
<snip>

;; ANSWER SECTION:
dartmouth.edu. 3600 IN MX 5 dartmouth-edu.mail.protection.outlook.com.
```

Dig* (Domain Information Groper) is more powerful than *nslookup

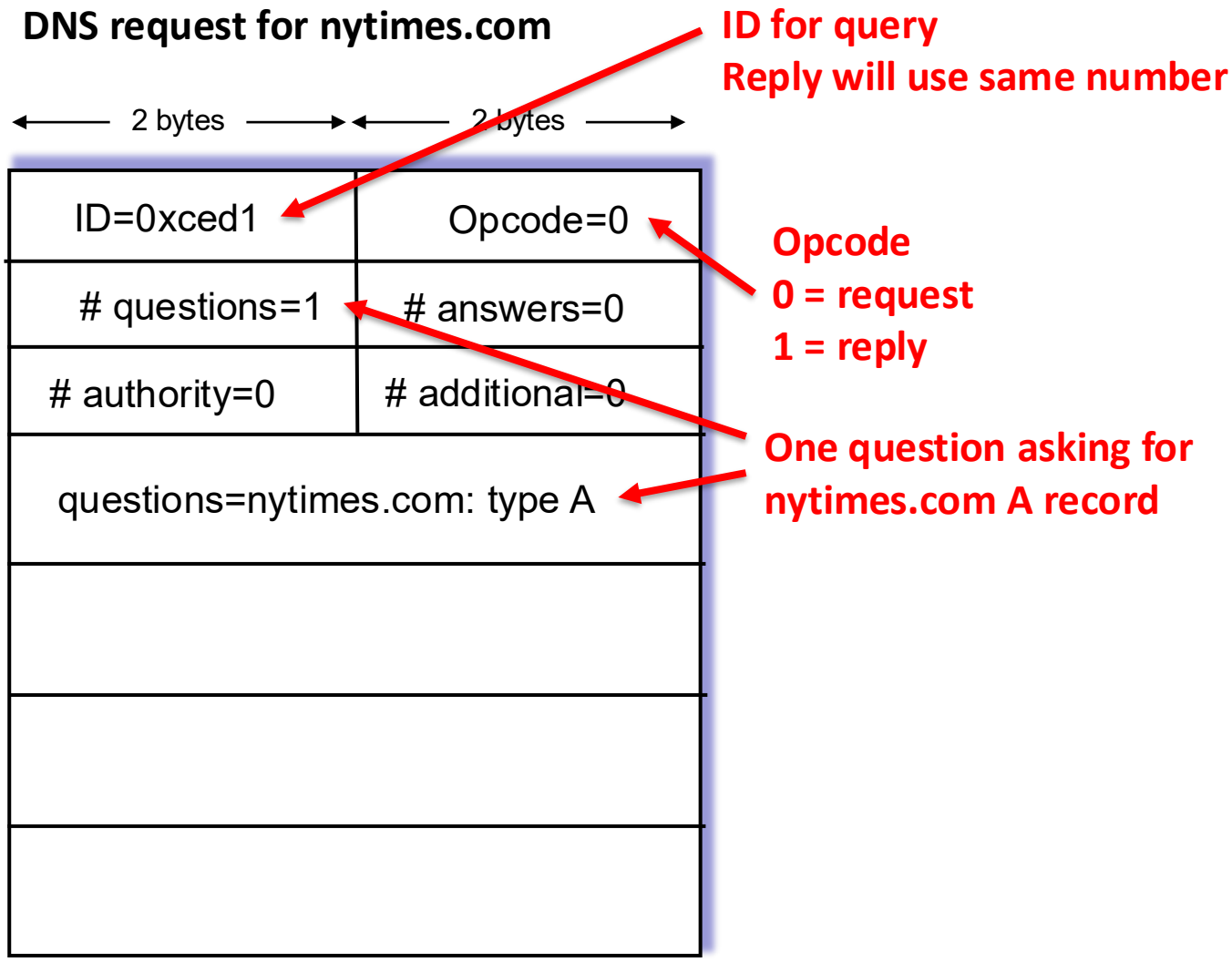
whois can also be helpful

$ **whois dartmouth.edu**
% IANA WHOIS server
% for more information on IANA, visit http://www.iana.org
% This query returned 1 object
refer: whois.educause.edu
domain: EDU
<snip>
Domain Name: DARTMOUTH.EDU

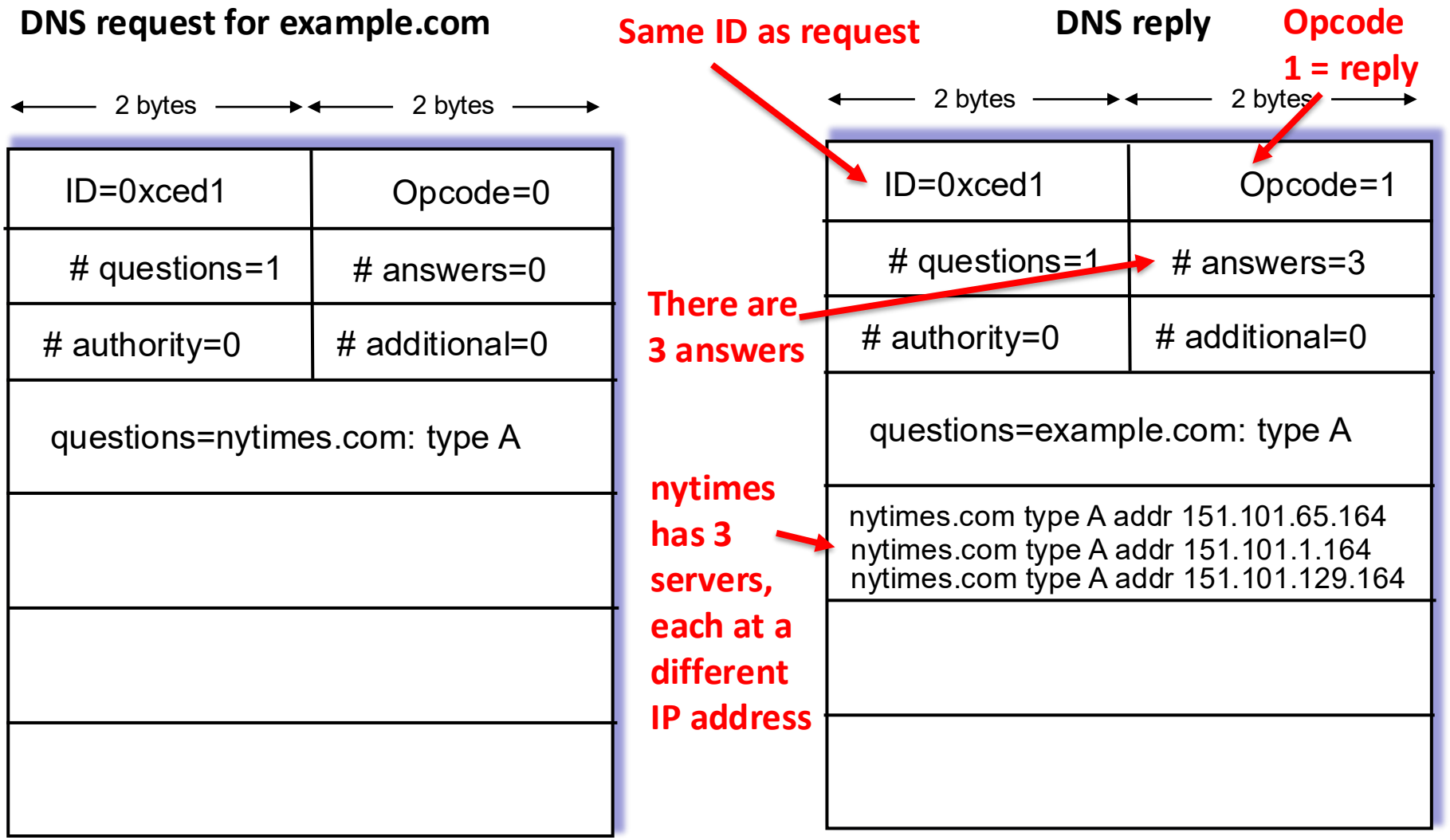
Registrant:
Dartmouth College
Dartmouth Network Services
4 Currier Place, Suite 101
Hanover, NH 03755, USA

Administrative Contact:
<a real person's name appears here, but redacted on this slide>
Dartmouth Information Technology & Consulting
HB 6028
Hanover, NH 03755, USA
+1.6036460999
dartmouth.network.services@dartmouth.edu
<snip>
Domain record activated: 03-Mar-1986
Domain record last updated: 04-Jun-2025
Domain expires: 31-Jul-2028

DNS Query and Reply messages have the same format

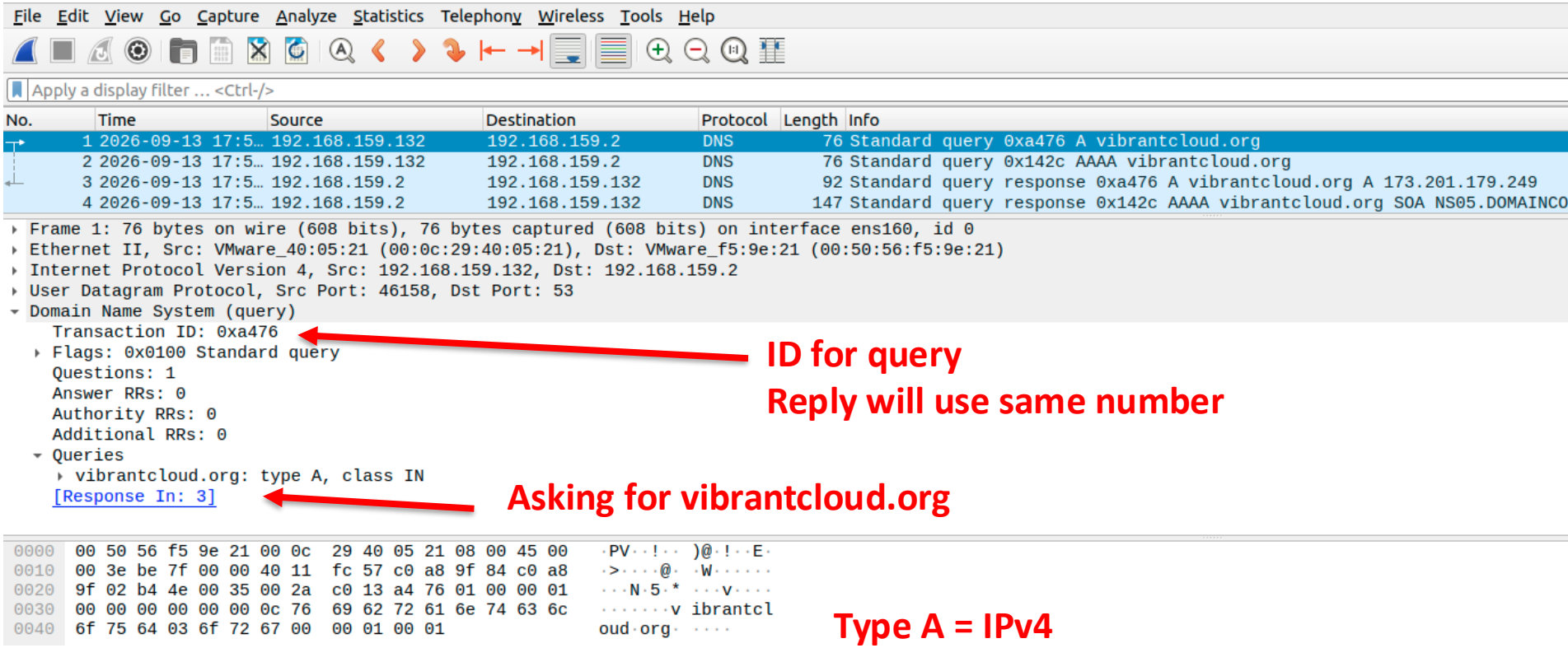


DNS Query and Reply messages have the same format



Wireshark shows DNS in action for a request for vibrantcloud.org

vibrantcloud_dns_tcp_http.pcapng



ID for query

Reply will use same number

Asking for vibrantcloud.org

Type A = IPv4

Type AAAA = IPv6

Inserting records into DNS normally involves registering web and email servers

Example: new startup “Network Utopia”

- Register name *networkuptopia.com* at *DNS registrar*
  - Formerly only registrar was Network Solutions, now there are many registrars
  - Provide names, IP addresses of authoritative name server (primary and secondary)
  - Registrar inserts NS, A records into .com TLD server (.com here):
    - name server: networkutopia.com, dns1.networkutopia.com, NS
    - authoritative name server ip: dns1.networkutopia.com, 212.212.212.1, A
- Create authoritative server locally with IP address 212.212.212.1
  - Type A record for www.networkuptopia.com with web server IP address
  - Type MX record for networkutopia.com with mail server IP address

Adapted from Kurose and Ross: Computer Networking: A Top-Down Approach

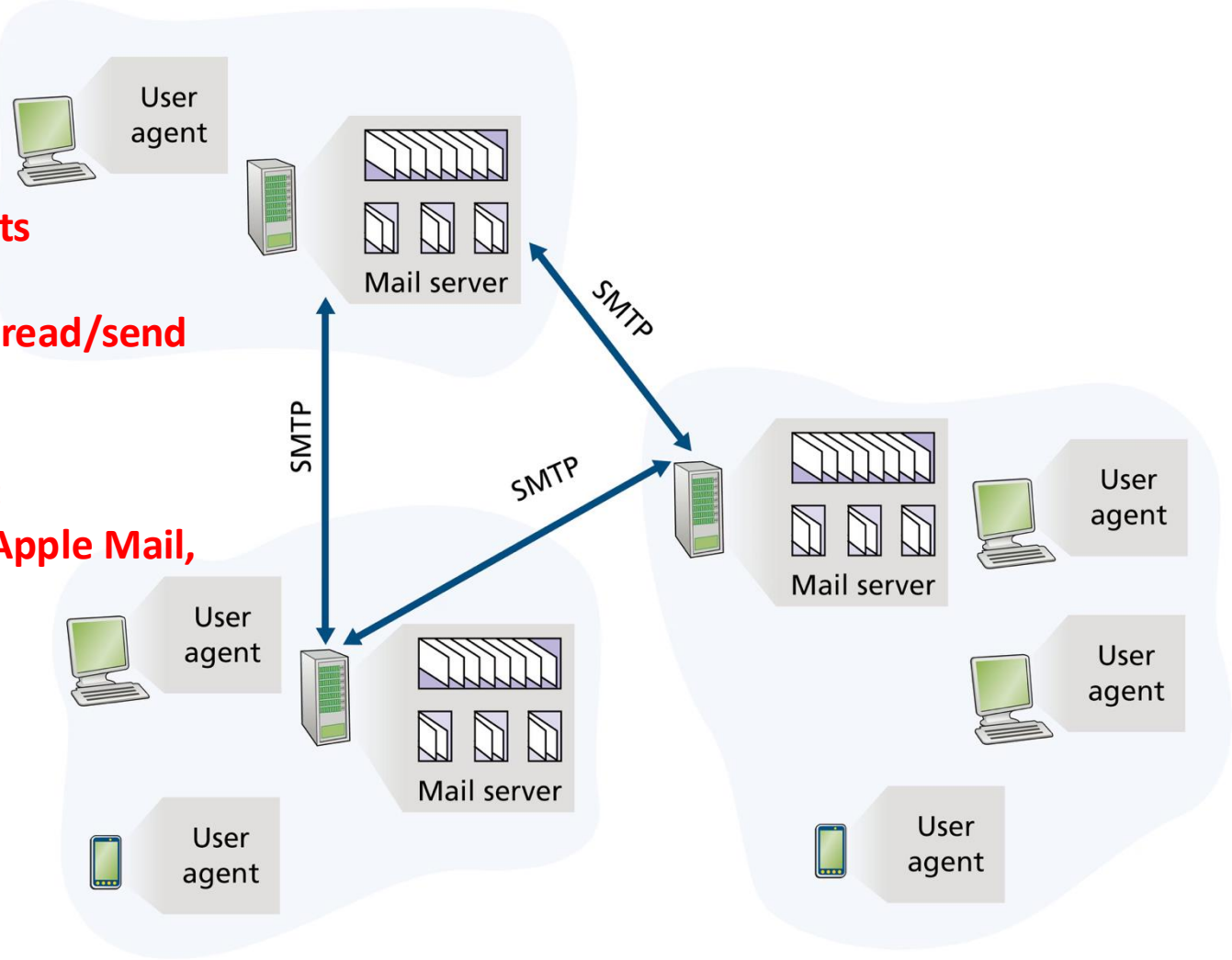
Agenda

1. IP addresses: NAT
2. Messages
3. DNS
4. Email
5. Web
6. Exercises

Email has three main components: user agents, mail servers, and SMTP

User agents
Users can compose/read/send mail

Examples:
Outlook, Apple Mail, Gmail



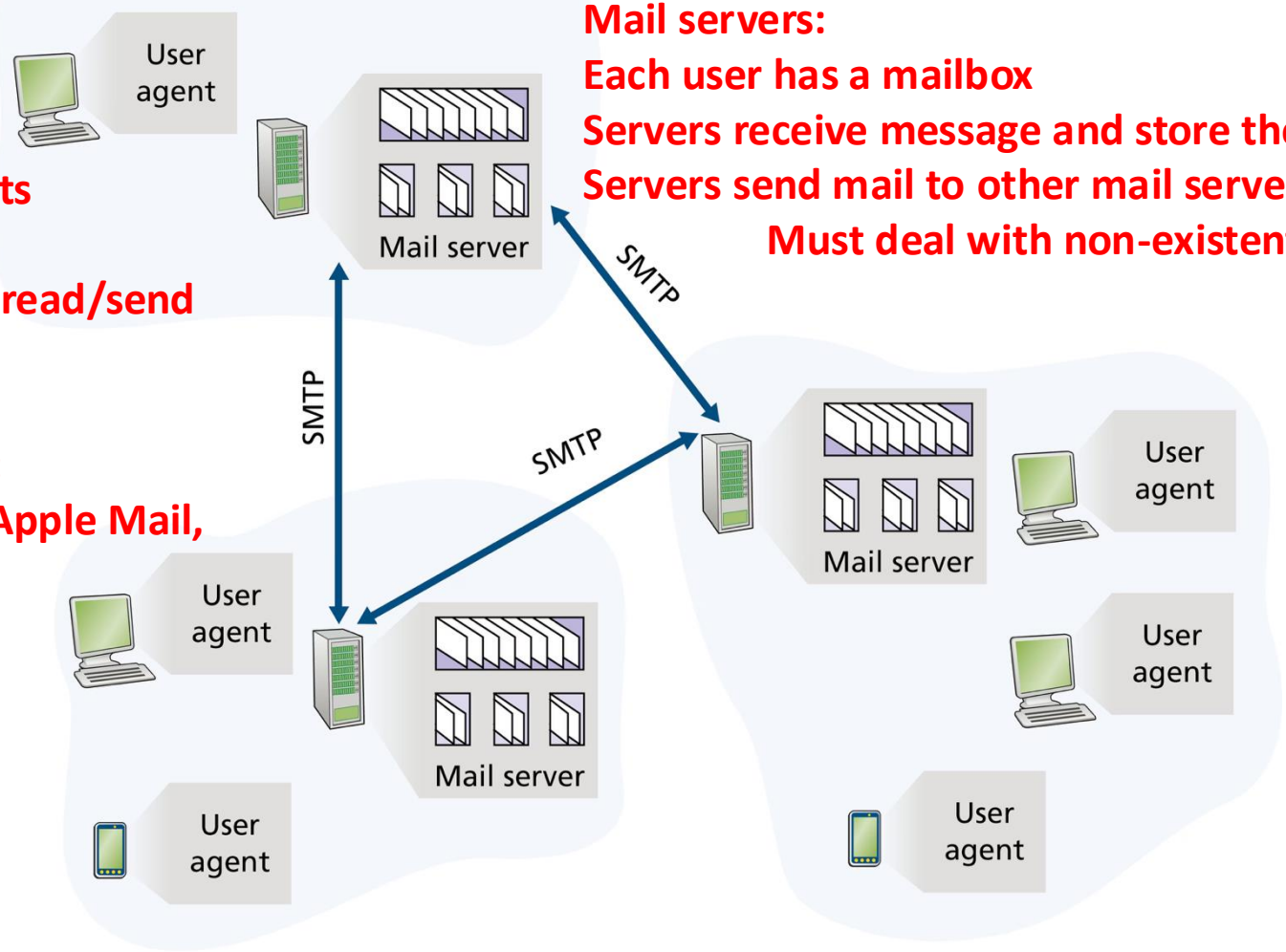
Adapted from Kurose and Ross: Computer Networking: A Top-Down Approach

Email has three main components: user agents, mail servers, and SMTP

User agents
Users can compose/read/send mail

Examples:
Outlook, Apple Mail, Gmail

Mail servers:
Each user has a mailbox
Servers receive message and store them for agents
Servers send mail to other mail servers
Must deal with non-existent mailboxes



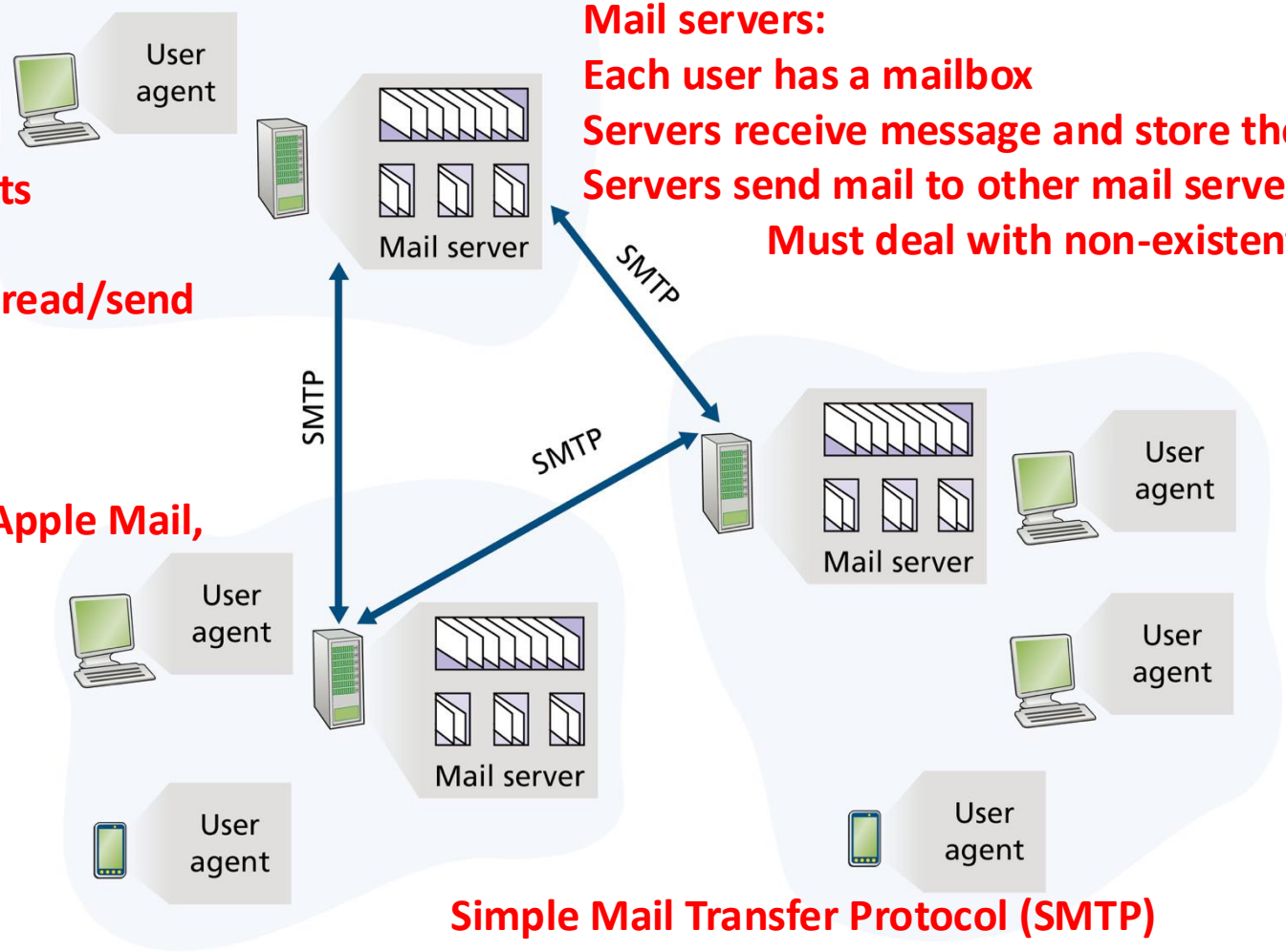
Adapted from Kurose and Ross: Computer Networking: A Top-Down Approach

Email has three main components: user agents, mail servers, and SMTP

User agents
Users can compose/read/send mail

Examples:
Outlook, Apple Mail, Gmail

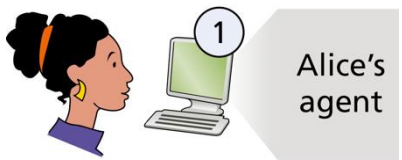
Mail servers:
Each user has a mailbox
Servers receive message and store them for agents
Servers send mail to other mail servers
Must deal with non-existent mailboxes



Simple Mail Transfer Protocol (SMTP)
Sends mail between servers

Adapted from Kurose and Ross: Computer Networking: A Top-Down Approach

Email from Alice to Bob starts with Alice's User Agent (UA)



1) Alice uses UA to compose e-mail message "to" bob@someschool.edu

UA sends message to Alice's mail server

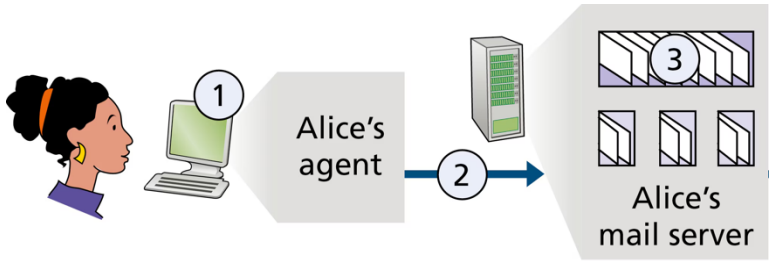


1) Alice uses UA to compose e-mail message "to" bob@someschool.edu

2) Alice's UA sends message to her mail server; message placed in message queue

Adapted from Kurose and Ross: Computer Networking: A Top-Down Approach

Alice's mail server stores message, opens TCP connection with Bob's mail server



Key:

Message queue

User mailbox

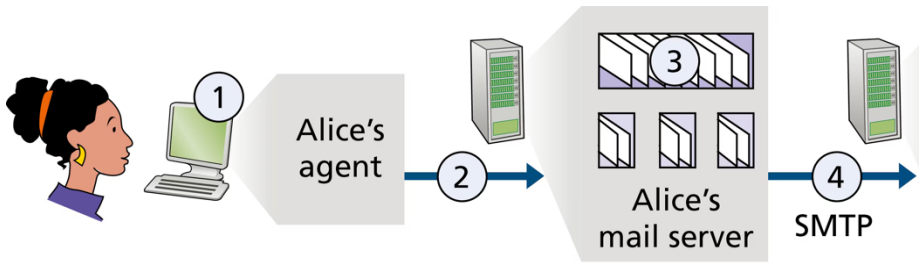
1) Alice uses UA to compose e-mail message "to" bob@someschool.edu

2) Alice's UA sends message to her mail server; message placed in message queue

3) Client side of SMTP opens TCP connection with Bob's mail server

Adapted from Kurose and Ross: Computer Networking: A Top-Down Approach

Alice's mail server sends message to Bob's mail server using SMTP



Key:

Message queue

User mailbox

1) Alice uses UA to compose e-mail message "to" bob@someschool.edu

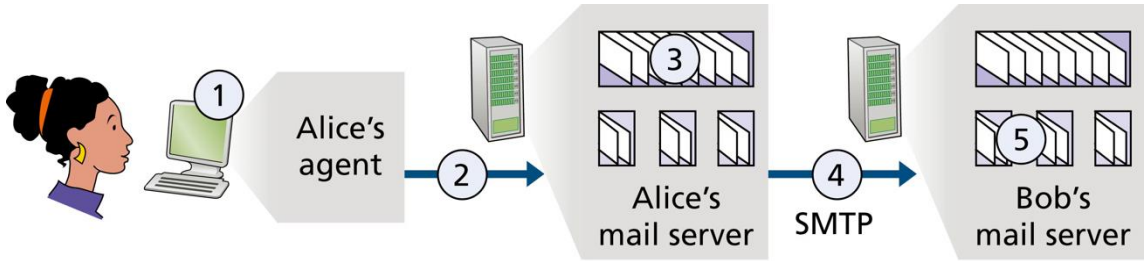
2) Alice's UA sends message to her mail server; message placed in message queue

3) Client side of SMTP opens TCP connection with Bob's mail server

4) SMTP client sends Alice's message over the TCP connection

Adapted from Kurose and Ross: Computer Networking: A Top-Down Approach

Bob's mail server stores message in Bob's mailbox



Key:

Message queue

User mailbox

1) Alice uses UA to compose e-mail message "to" bob@someschool.edu

2) Alice's UA sends message to her mail server; message placed in message queue

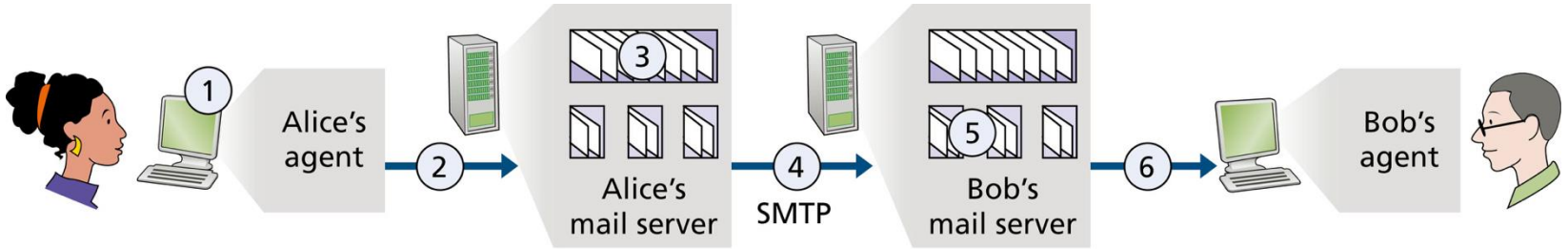
3) Client side of SMTP opens TCP connection with Bob's mail server

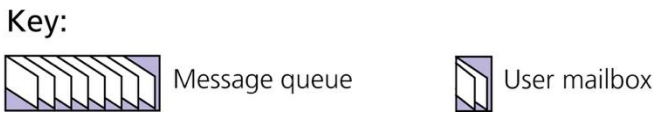
4) SMTP client sends Alice's message over the TCP connection

5) Bob's mail server places the message in Bob's mailbox

Adapted from Kurose and Ross: Computer Networking: A Top-Down Approach

Bob reads Alice's message using his User Agent





1) Alice uses UA to compose e-mail message "to" bob@someschool.edu

2) Alice's UA sends message to her mail server; message placed in message queue

3) Client side of SMTP opens TCP connection with Bob's mail server

4) SMTP client sends Alice's message over the TCP connection

5) Bob's mail server places the message in Bob's mailbox

6) Bob invokes his user agent to read message

Adapted from Kurose and Ross: Computer Networking: A Top-Down Approach

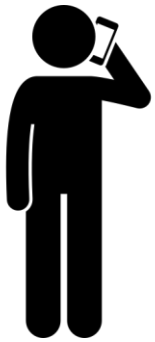
Email notes

- Mail uses TCP to reliably transfer email message from client (mail server initiating connection) to server on port 25
- Direct transfer between servers:
  - Sending server (acting like client) to receiving server
  - No intermediate servers that store and forward
  - If server down, “client” server tries again later
- Three phases of transfer
  - Handshaking (greeting)
  - Transfer of messages
  - Closure
- Command/response interaction (like HTTP)
  - commands: ASCII text
  - response: status code and phrase
- Messages must be in 7-bit ASCI (must convert non-text to text characters!)

Sending a SMTP Message

- How to format message?
- What does the server expect?
- Request for Comments (RFC)
  - SMTP defined in RFC 5321 & RFC 5322
  - DNS, HTTP etc.

Communicating with Server



Dialing 1-800-123-4567

How can I help you?

I'm Bob

I want to send a message

Mm-Hmm

It is going to .....

.

.

.

Bye





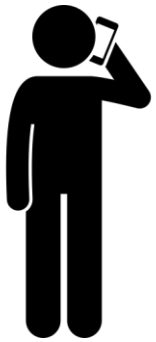
Communicating with Server



Dialing 1-800-123-4567

How can I help you?





I'm Bob

I want to send a message

Mm-Hmm



It is going to …..

.

.

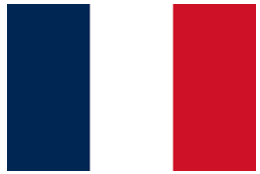
.

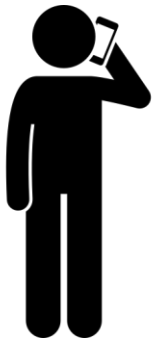
Bye

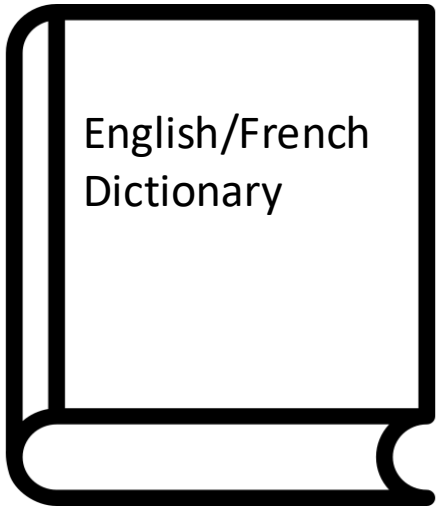


Communicating with Server









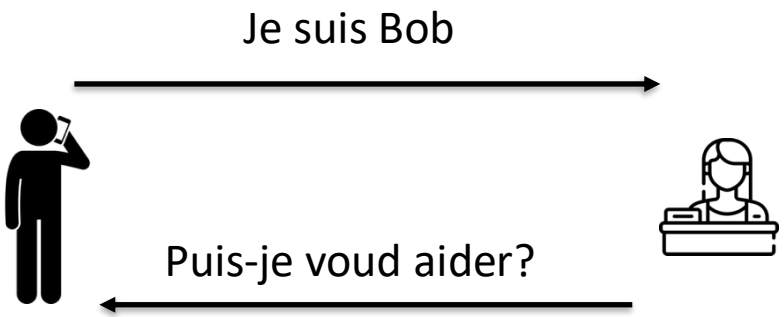


Communicating with Server

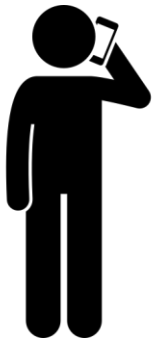




English	French
I am Bob	Je suis Bob
How can I help you?	Puis-je vous aider?
Bye	Au revoir

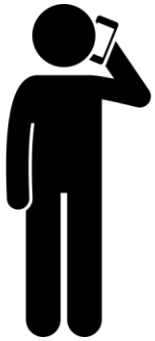


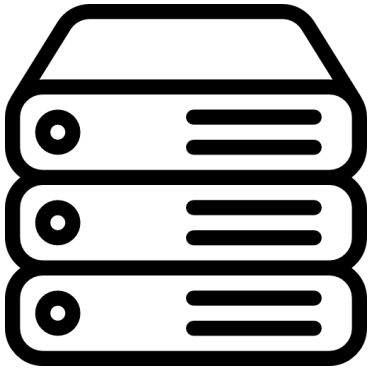
Communicating with Server





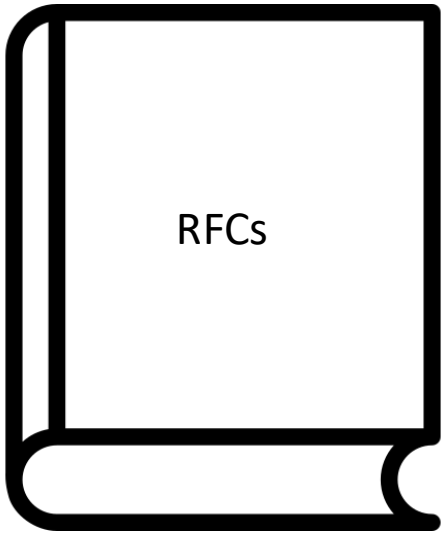
Communicating with Server





SMTP Server

What is the client's dictionary?



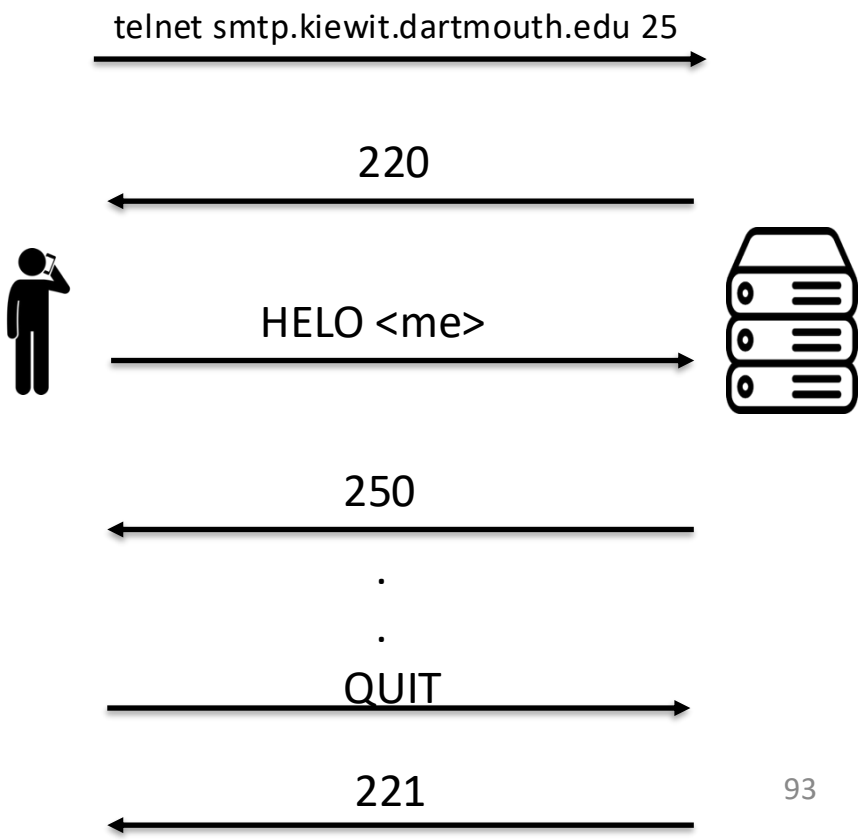
Sending a SMTP Message

What I want to do	RFC 5321 & RFC 5322
Connect	Port 25
Introduce myself	HELO
Inform of sender	MAIL FROM
Inform of recipient	RCPT TO
Send content of message	DATA
End connection	QUIT

Communicating with Server

What I want to do	RFC 5321 & RFC 5322
Connect	Port 25
Introduce myself	HELO
Inform of sender	MAIL FROM
Inform of recipient	RCPT TO
Send content of message	DATA
End connection	QUIT

Reply Codes	Meaning
220	Service Ready
250	Ok
552/451	Error



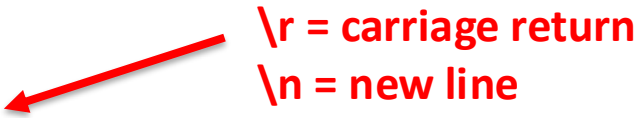
Demo with CS60/160 SMTP Server

```
$ telnet smtp.kiewit.dartmouth.edu 25
Trying 10.28.55.168...
Connected to smtp.kiewit.dartmouth.edu.
Escape character is '^]'.
220 mailhog.example ESMTP MailHog
HELO test.com
250 Hello test.com
MAIL FROM: <tjp@test.com>
250 Sender tjp@test.com ok
RCPT TO: <you@test.com>
250 Recipient you@test.com ok
DATA
354 End data with <CR><LF>.<CR><LF>
This is line 1
this is another line
I'll end with just a period
.
250 Ok: queued as sQ8bP0_qeuegsw-GG6rDYPmOjcGd6cjtkMhqwUUA228=@mailhog.example
QUIT
221 Bye
Connection closed by foreign host.
```

smtp.kiewit.dartmouth.edu is not a real email server

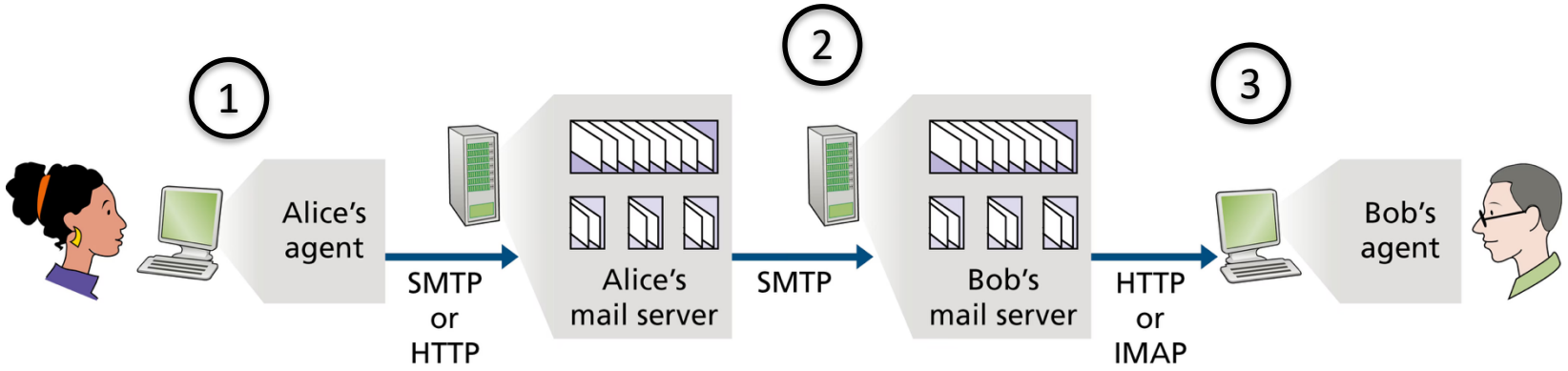
Your plans for using it as a base for your spam empire are dashed ☺

Sending as one message




```
"HELO localhost\r\n\
MAIL FROM:<alice@cs60.com>\r\n\
RCPT TO:<bob@cs60.com>\r\n\
DATA\r\n\
From: Alice <alice@cs60.com>\r\n\
To: Bob <bob@cs60.com>\r\n\
Subject: Test email using netcat\r\n\
\r\n\
Hello cs60 student, this is a test email.\r\n\
.\r\n\
QUIT\r\n"
```

SMTP sends messages, IMAP or HTTP retrieve messages from server



1. Alice's User Agent sends messages to email server using SMTP or HTTP (gmail)

2. Alice's mail server sends message to Bob's mail server using SMTP

3. Bob's User Agent retrieves messages from his mail server using either
   - IMAP (Outlook)
   - HTTP on top of SMTP
   - POP3 (post office protocol, less common now)

IMAP/HTTP leave messages on server, download copy

SMTP = send
IMAP = receive

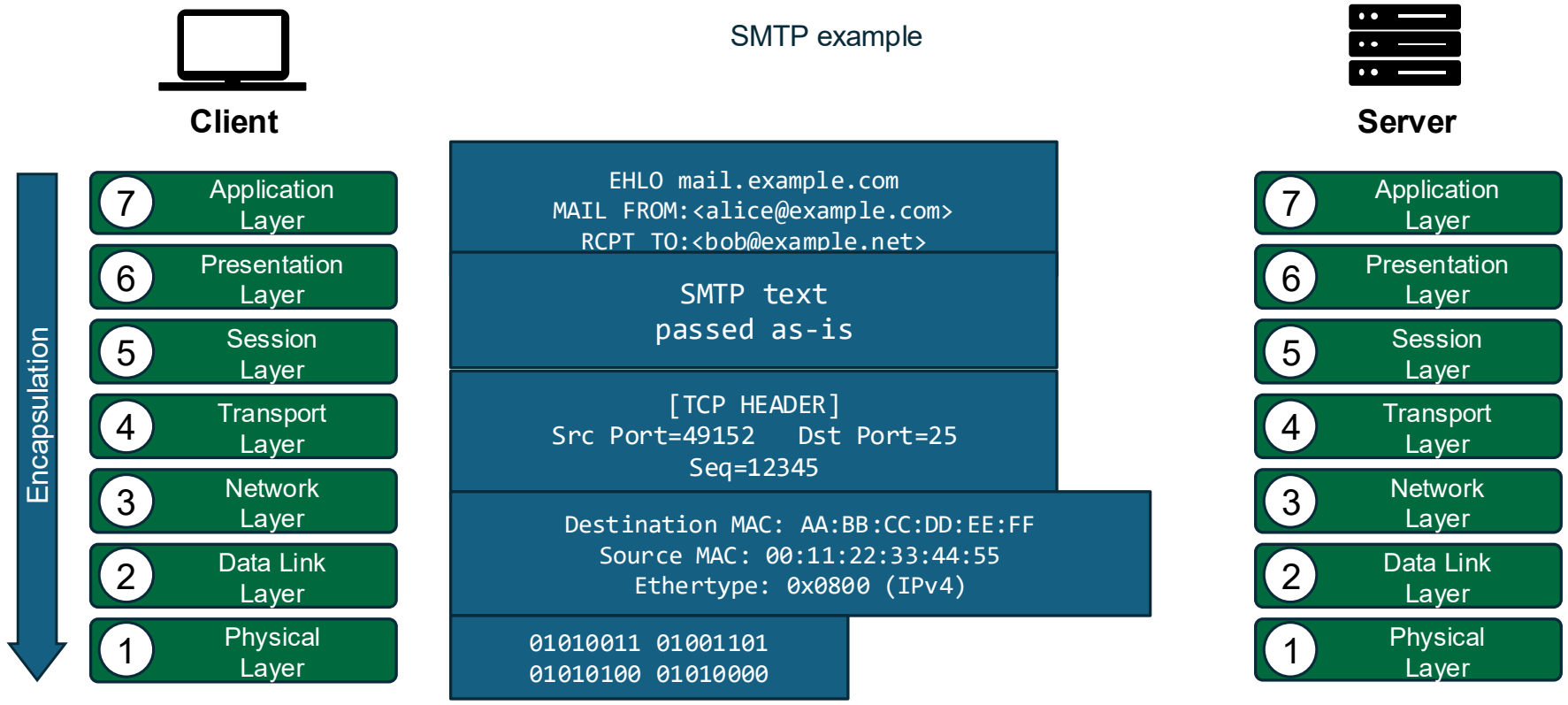
Erases messages on server after read

IMAP/HTTP better for multi-device users because messages stay on server so other devices can access them
POP3 only downloads message to one device!

Could Alice send directly to Bob's UA?

Adapted from Kurose and Ross: Computer Networking: A Top-Down Approach

OSI Model


Client
SMTP example
Server
Encapsulation
7 Application Layer
6 Presentation Layer
5 Session Layer
4 Transport Layer
3 Network Layer
2 Data Link Layer
1 Physical Layer
EHLO mail.example.com
MAIL FROM:<alice@example.com>
RCPT TO:<bob@example.net>
SMTP text
passed as-is
[TCP HEADER]
Src Port=49152 Dst Port=25
Seq=12345
Destination MAC: AA:BB:CC:DD:EE:FF
Source MAC: 00:11:22:33:44:55
Ethertype: 0x0800 (IPv4)
01010011 01001101
01010100 01010000
7 Application Layer
6 Presentation Layer
5 Session Layer
4 Transport Layer
3 Network Layer
2 Data Link Layer
1 Physical Layer

Agenda

1. IP addresses: NAT
2. Messages
3. DNS
4. Email
5. Web
6. Exercises

CS10 Review: Web pages are just text documents on a server

- Uniform Resource Locators (URLs) identify objects on the Internet such as a webpage, image, or file

- Web pages are just text files that may consists of *objects,* each of which can be stored on different Web servers, addressed by a URL

http://tjp.cs.dartmouth.edu/cs60/index.html

Layer 7 protocol: HyperText Transfer Protocol (http)

Host computer DNS provides IP address for the host name

Directory

File

The file may contain URLs to other objects
For example, a web page written in HTML may have 10 images
Each image is a separate file on the web

HTTP follows a client/server model for serving web pages

HTTP: HyperText Transfer Protocol

- Web's application layer protocol
- Client/server model:
  - *Client:* browser that requests, receives, (using HTTP protocol) and "displays" Web objects
  - *Server:* Web server sends (using HTTP protocol) objects in response to requests



Server has fixed IP address

Find IP address with DNS A record

Popular web servers include:

- **Apache**
- **IIS**
- **Nginx**

Adapted from Kurose and Ross: Computer Networking: A Top-Down Approach

HTTP uses TCP and is "stateless"

HTTP uses TCP:

- Client initiates TCP connection (creates socket) to server, port 80
- Server accepts TCP connection from client
- HTTP messages (application-layer protocol messages) exchanged between browser (HTTP client) and Web server (HTTP server)
- TCP connection closed when data returned to client

HTTP is "stateless"

- Server maintains *no* information about past client requests

aside

Protocols that maintain "state" are complex!

- Past history (state) must be maintained
- If server/client crashes, their views of "state" may be inconsistent, must be reconciled

TCP (Layer 4) ensures packets are delivered in order HTTP (Layer 7) does not have to worry about missing or out of order packets! Layer 4 handles it

Adapted from Kurose and Ross: Computer Networking: A Top-Down Approach

HTTP has two varieties, non-persistent and persistent

Non-persistent HTTP

1. TCP connection opened
2. At most one object sent over TCP connection
3. TCP connection closed

Downloading multiple objects (say a web page with multiple embedded images) required multiple connections

Persistent HTTP

- TCP connection opened to a server
- Multiple objects can be sent over *single* TCP connection between client, and that server
- TCP connection closed

Adapted from Kurose and Ross: Computer Networking: A Top-Down Approach

Non-persistent HTTP opens a connection for each object in a page

User enters http://www.cs.dartmouth.edu/~tjp/cs60/index.html
(say page contains text and references to 10 jpeg images)





1a. HTTP client initiates TCP connection to HTTP server (process) at www.cs.dartmouth.edu on port 80

1b. HTTP server at host www.cs.dartmouth.edu waiting for TCP connection at port 80 "accepts" connection, notifying client

2. HTTP client sends HTTP *request message* (containing URL) into TCP connection socket. Message indicates that client wants object index.html in directory ~tjp/cs60

3. HTTP server receives request message, forms *response message* containing requested object, and sends message into its socket

4. HTTP server closes TCP connection.

5. HTTP client receives response message containing html file, displays html. Parsing html file, finds 10 referenced jpeg objects

time

Repeat steps 1 through 4 for each of the objects on the web page!

Adapted from Kurose and Ross: Computer Networking: A Top-Down Approach

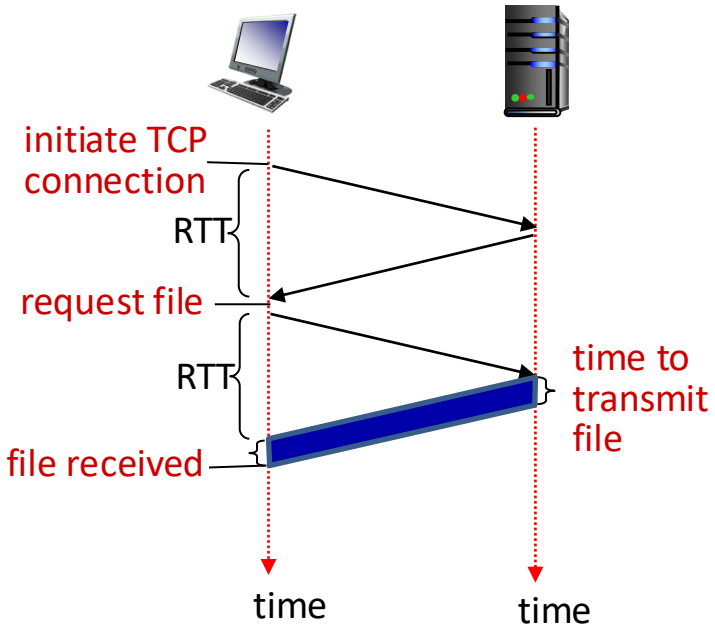
Round Trip Time (RTT): time from making a request to receiving a response

Round Trip Time (RTT)

Time for a small packet to travel from client to server and back

HTTP response time (per object):

- One RTT to initiate TCP connection
- One RTT for HTTP request and first few bytes of HTTP response to return
- Plus, time to transmit object/file



Non-persistent HTTP response time = 2RTT+ file transmission time

Adapted from Kurose and Ross: Computer Networking: A Top-Down Approach

HTTP version 1.1 introduced persistent connections

Non-persistent HTTP issues*:*

- Requires 2 RTTs per object
  - One connection
  - One object request
- OS overhead for *each* TCP connection
- Browsers often open multiple parallel TCP connections to fetch referenced objects in parallel

Persistent HTTP (HTTP1.1):

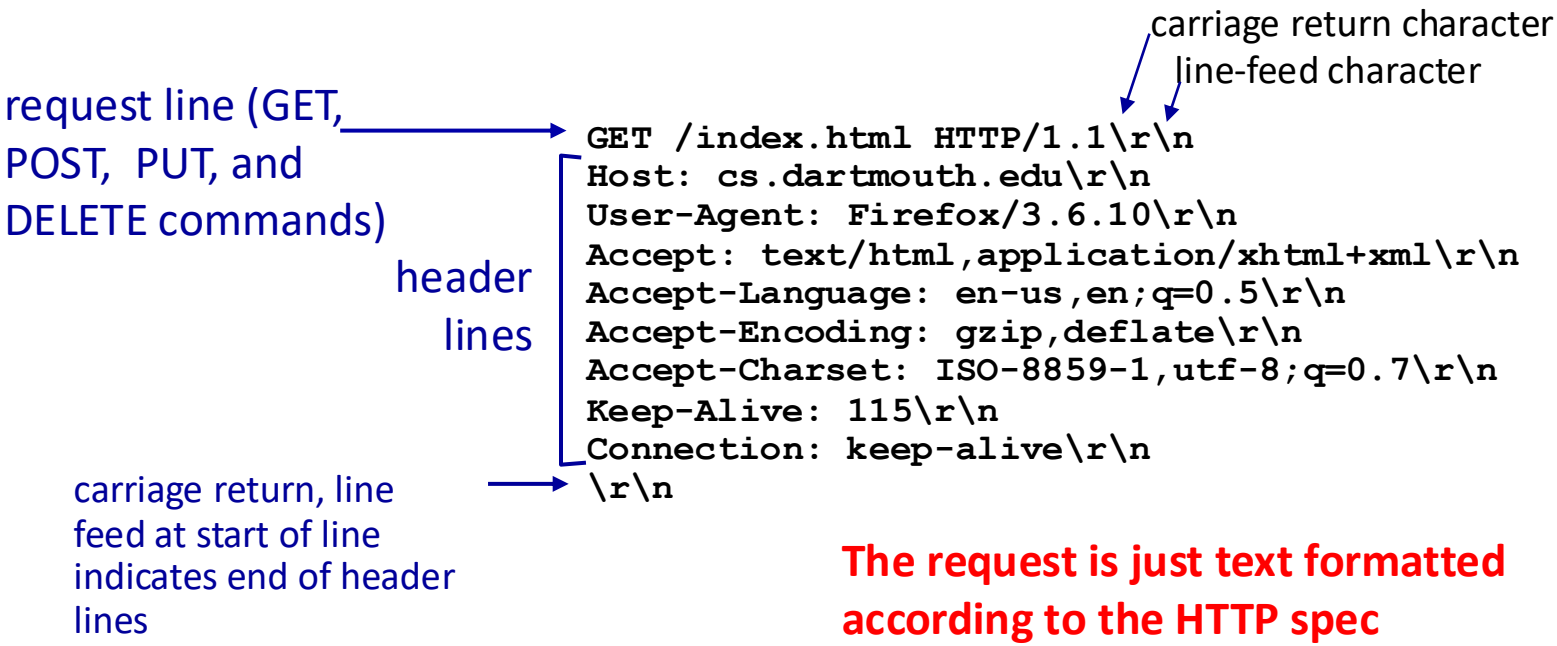
- Server leaves connection open after sending response
- Subsequent HTTP messages between same client/server sent over open connection
- Client sends requests as soon as it encounters a referenced object
- As little as one RTT for all the referenced objects (cutting response time in half)

Adapted from Kurose and Ross: Computer Networking: A Top-Down Approach

HTTP has two types of messages: requests and responses

HTTP request message:

- ASCII (human-readable format)

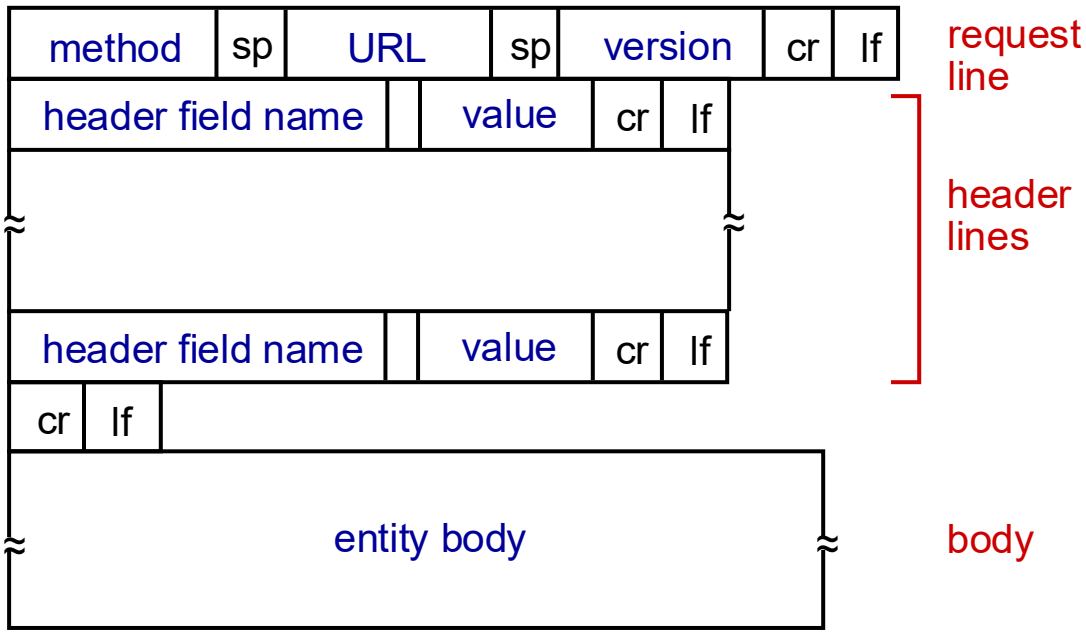


The request is just text formatted according to the HTTP spec

First two lines are required, others are optional

Adapted from Kurose and Ross: Computer Networking: A Top-Down Approach

HTTP request message: general format



Adapted from Kurose and Ross: Computer Networking: A Top-Down Approach

HTTP has several "verbs": GET, POST, PUT, and DELETE

GET method :

- Used to get data from a server
- Include parameters in URL query string of HTTP GET request message (following a '?', separated by &):

```
www.somesite.com/animalsearch?animal=rabbit&
food=carrot
```

PUT method:

- Uploads new file (object) to server
- Completely replaces file that exists at specified URL with content in entity body of POST HTTP request message

POST method:

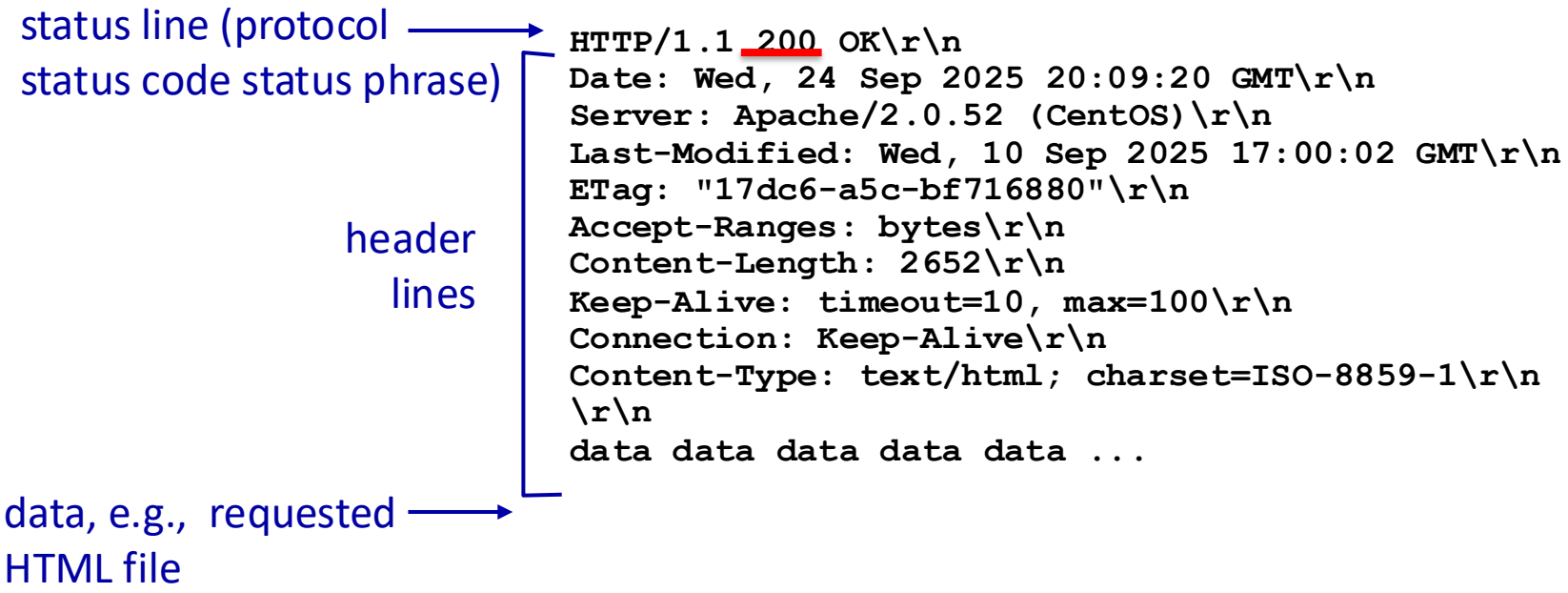
- Web page often includes form input
- User input sent from client to server in entity body of HTTP POST request message

DELETE method:

- Deletes a resources on the server

Adapted from Kurose and Ross: Computer Networking: A Top-Down Approach

Servers send back an HTTP response



Common status codes

200 OK

Request succeeded, requested object later in this message

301 Moved Permanently

Requested object moved, new location specified later in this message (in Location: field)

400 Bad Request

Request msg not understood by server

404 Not Found

Requested document not found on this server

505 HTTP Version Not Supported

Adapted from Kurose and Ross: Computer Networking: A Top-Down Approach

DEMO

1. Telnet to Web server:

```
telnet vibrantcloud.org 80
```

- Opens TCP connection to port 80 (default HTTP server port) at a server I control
- Anything typed in will be sent to port 80 at vibrantcloud.org

2. Type in a GET HTTP request:

```
GET /index.html HTTP/1.1
Host: vibrantcloud.org
```

- By typing this in (hit carriage return twice), you send this minimal (but complete) GET request to HTTP server
- Do this on the Linux VM (Mac sends first line only)

3. Look at response message sent by HTTP server
(or use Wireshark to look at captured

```
<!doctype html>

<html lang="en">
<head>
  <meta charset="utf-8">
  <title>VibrantCloud</title>
  <meta name="description" content="VibrantCloud">
  <meta name="author" content="Anon">
</head>

<body>
  <h1>VibrantCloud</h1>
  <p>This is the famous VibrantCloud you've heard so much about.</p>
</body>
</html>
```

VibrantCloud

This is the famous VibrantCloud you've heard so much about.

Browser caching can use a conditional GET

Goal: don't send object if browser has up-to-date cached version

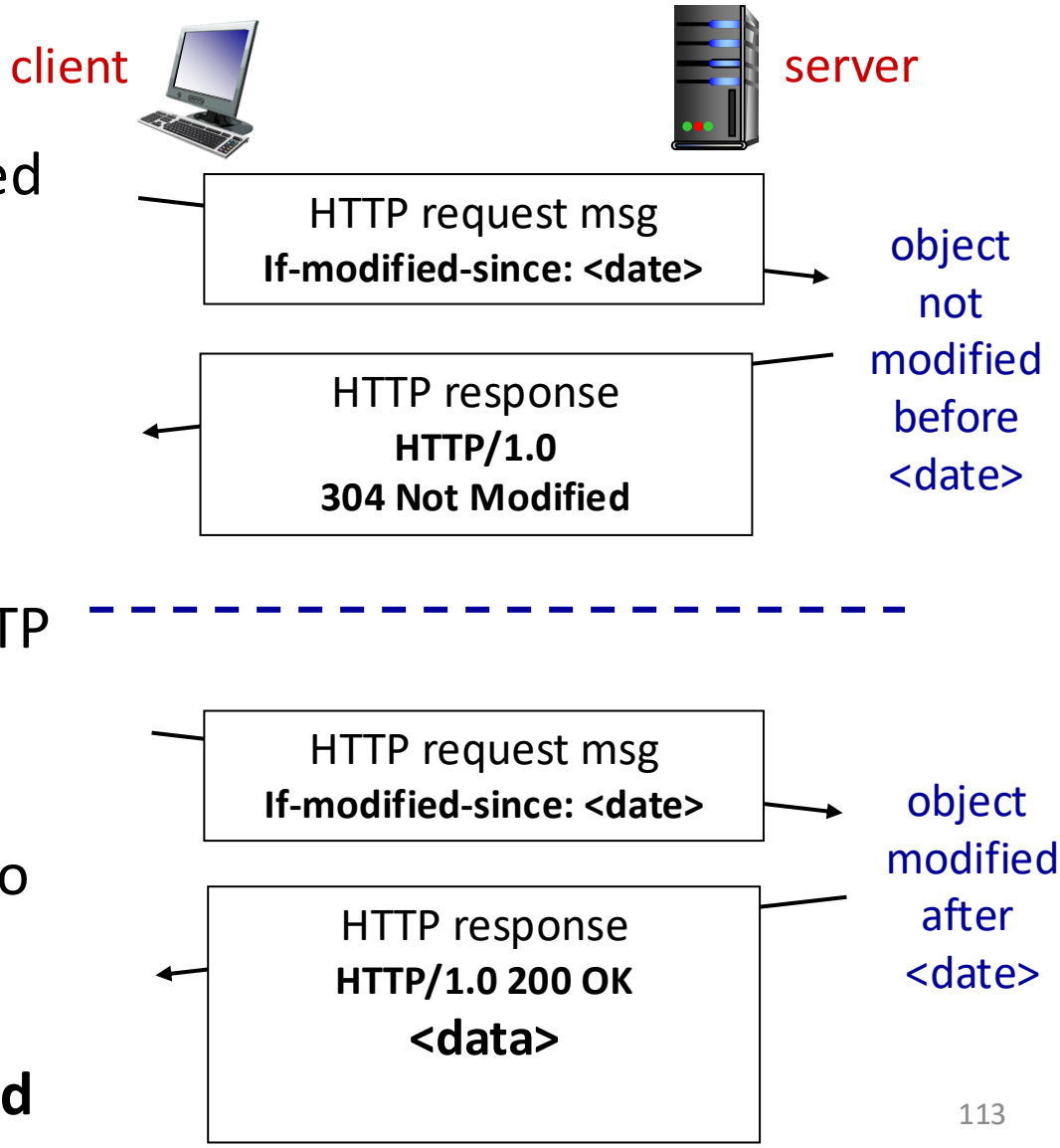
- No object transmission delay (or use of network resources)

- *Client:* specify date of browser-cached copy in HTTP request

  If-modified-since: <date>

- *Server:* response contains no object if browser-cached copy is up-to-date:

  HTTP/1.0 304 Not Modified



HTTP/1.1

Key goal: decreased delay in multi-object HTTP requests

HTTP1.1: introduced multiple, pipelined GETs over single TCP connection

- Server responds *in-order* (FCFS: first-come-first-served scheduling) to GET requests
- With FCFS, small object may have to wait for transmission (head-of-line (HOL) blocking) behind large object(s)
- Loss recovery (retransmitting lost TCP segments) stalls object transmission

Adapted from Kurose and Ross: Computer Networking: A Top-Down Approach

HTTP/2

Key goal: decreased delay in multi-object HTTP requests

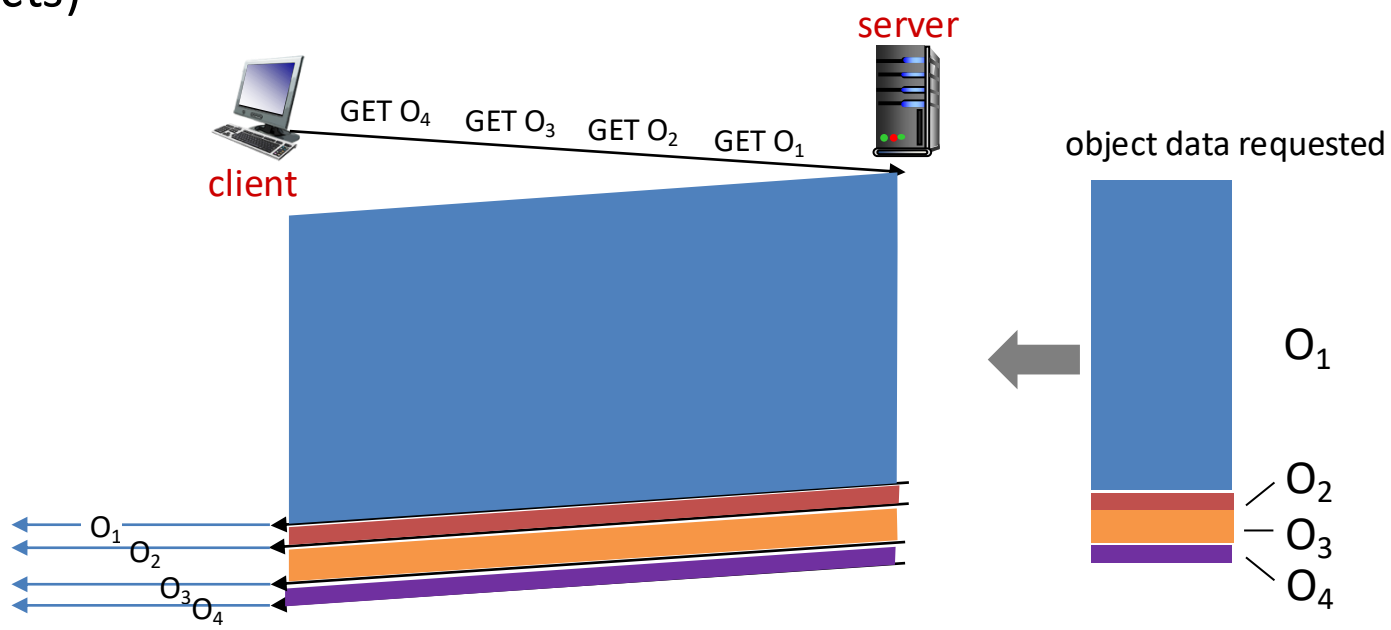
HTTP/2: [RFC 7540, 2015] increased flexibility at *server* in sending objects to client:

- Methods, status codes, most header fields unchanged from HTTP 1.1
- Transmission order of requested objects based on client-specified object priority (not necessarily FCFS)
- *Push* unrequested objects to client
- Divide objects into frames, schedule frames to mitigate HOL blocking

Adapted from Kurose and Ross: Computer Networking: A Top-Down Approach

Mitigating Head Of Line (HOL) problems

HTTP 1.1: client requests 1 large object (e.g., video file, and 3 smaller objects)

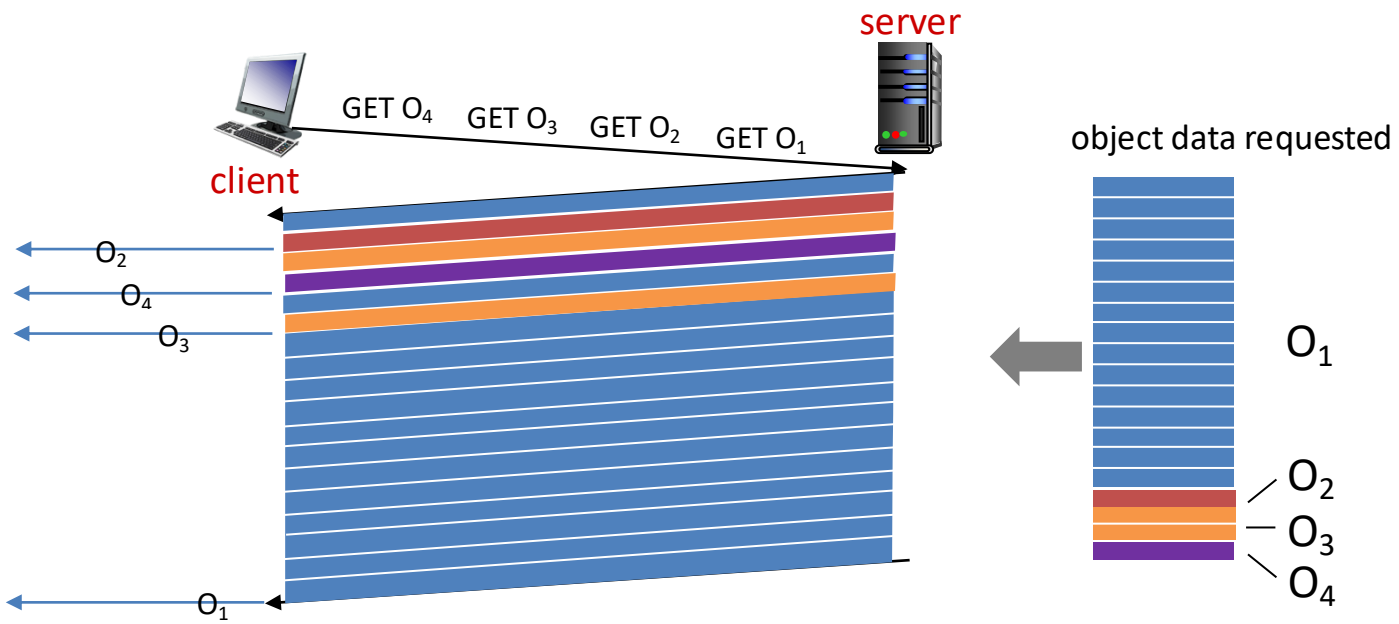


Objects delivered in order requested: O_2, O_3, O_4 wait behind O_1

Adapted from Kurose and Ross: Computer Networking: A Top-Down Approach

Mitigating HOL problems

HTTP/2: objects divided into frames, frame transmission interleaved



O_2, O_3, O_4 delivered quickly, O_1 slightly delayed

Adapted from Kurose and Ross: Computer Networking: A Top-Down Approach

HTTP/2 to HTTP/3

Key goal: decreased delay in multi-object HTTP requests

HTTP/2 over single TCP connection means:

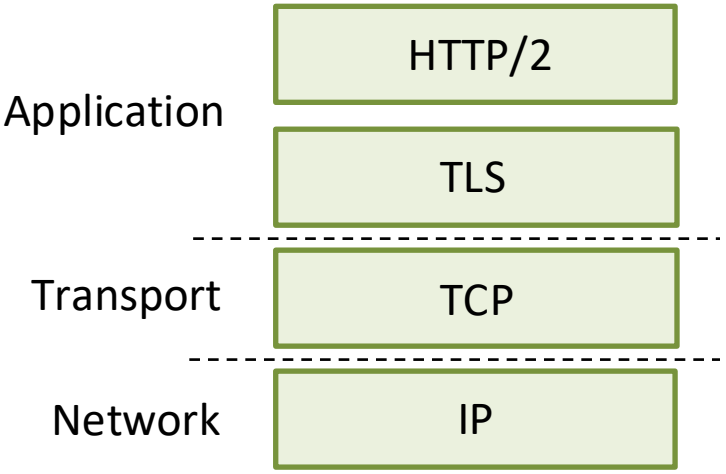
- Recovery from packet loss still stalls all object transmissions
  - As in HTTP 1.1, browsers have incentive to open multiple parallel TCP connections to reduce stalling, increase overall throughput
- No security over vanilla TCP connection
- HTTP/3: adds security , per object error- and congestion-control (more pipelining) over UDP
  - more on HTTP/3 in transport layer

Adapted from Kurose and Ross: Computer Networking: A Top-Down Approach

Google has developed a protocol called QUIC on top of UDP

Quick UDP Internet Connections (QUIC)

- Application-layer protocol, on top of UDP
  - Increase performance of HTTP
  - Deployed on many Google servers, apps (Chrome, mobile YouTube app)



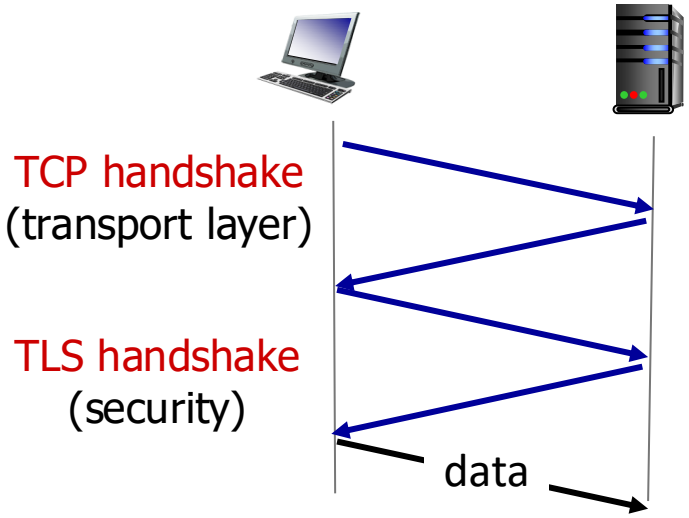
HTTP/2 over TCP

Google has developed a protocol called QUIC on top of UDP

QUIC adopts approaches we'll study in chapter for connection establishment, error control, congestion control

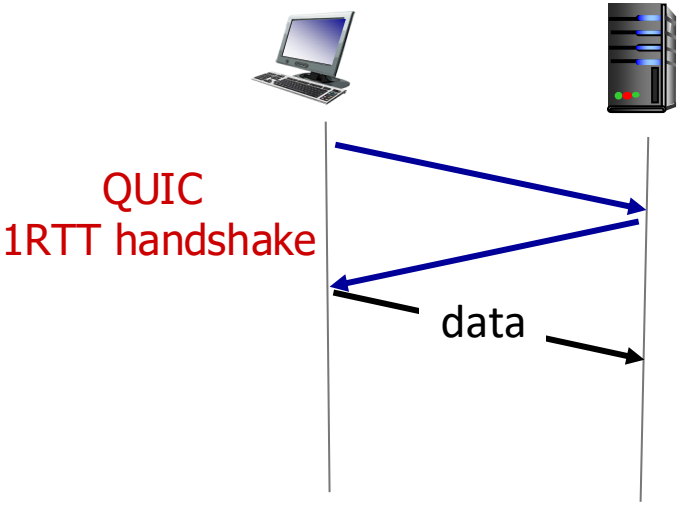
- **Error and congestion control:** "Readers familiar with TCP's loss detection and congestion control will find algorithms here that parallel well-known TCP ones." [from QUIC specification]
- **Connection establishment:** reliability, congestion control, authentication, encryption, state established in one RTT

QUIC has one connection handshake (vs two for TCP with TLS)



TCP (reliability, congestion control state) + TLS (authentication, crypto state)

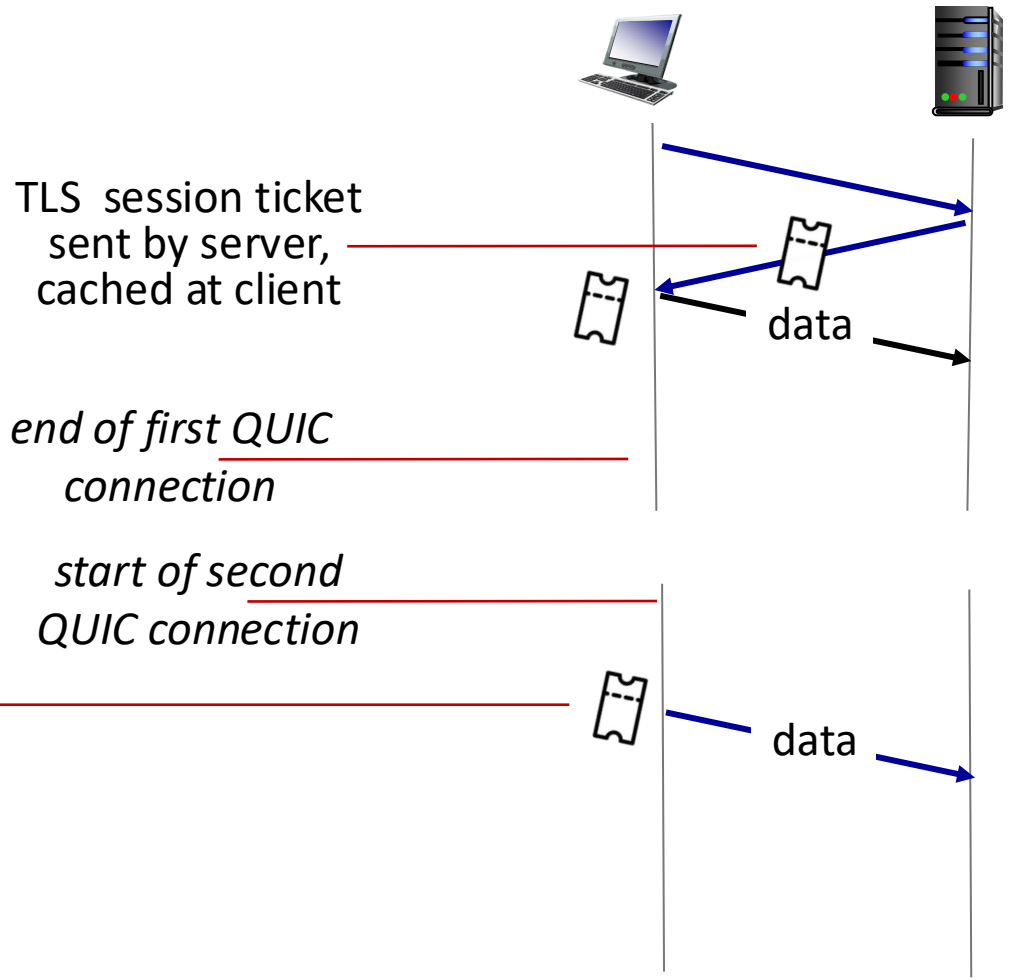
- 2 serial handshakes, one for TCP, for for TLS



QUIC: reliability, congestion control, authentication, crypto state

- 1 handshake

QUIC: 0-RTT Connection establishment



Client uses last session ticket for encryption, authentication for new connection

- 0 RTT handshake delay

Agenda

1. Messages
2. IP addresses: NAT
3. DNS
4. Email
5. Web
6. Exercises



Exercises

Write a program to send email to simulated server

- Use smtp.kiewet.dartmouth.edu on port 1025
- Use starter code on course web page for today
- Send yourself an email message

Write a program to query Google DNS server at 8.8.8.8

- Take the target domain as a parameter (e.g., dartmouth.edu, cnn.com)
- Take an optional parameter for the record type (A or MX, use A if not provided)
- Print all replies (look at what comes back from cnn.com, there are more than one A records returned)
- Ping the domain and confirm replies are from one of the responses
- Why does CNN have multiple DNS entries for cnn.com?