TCIPG Reading Group

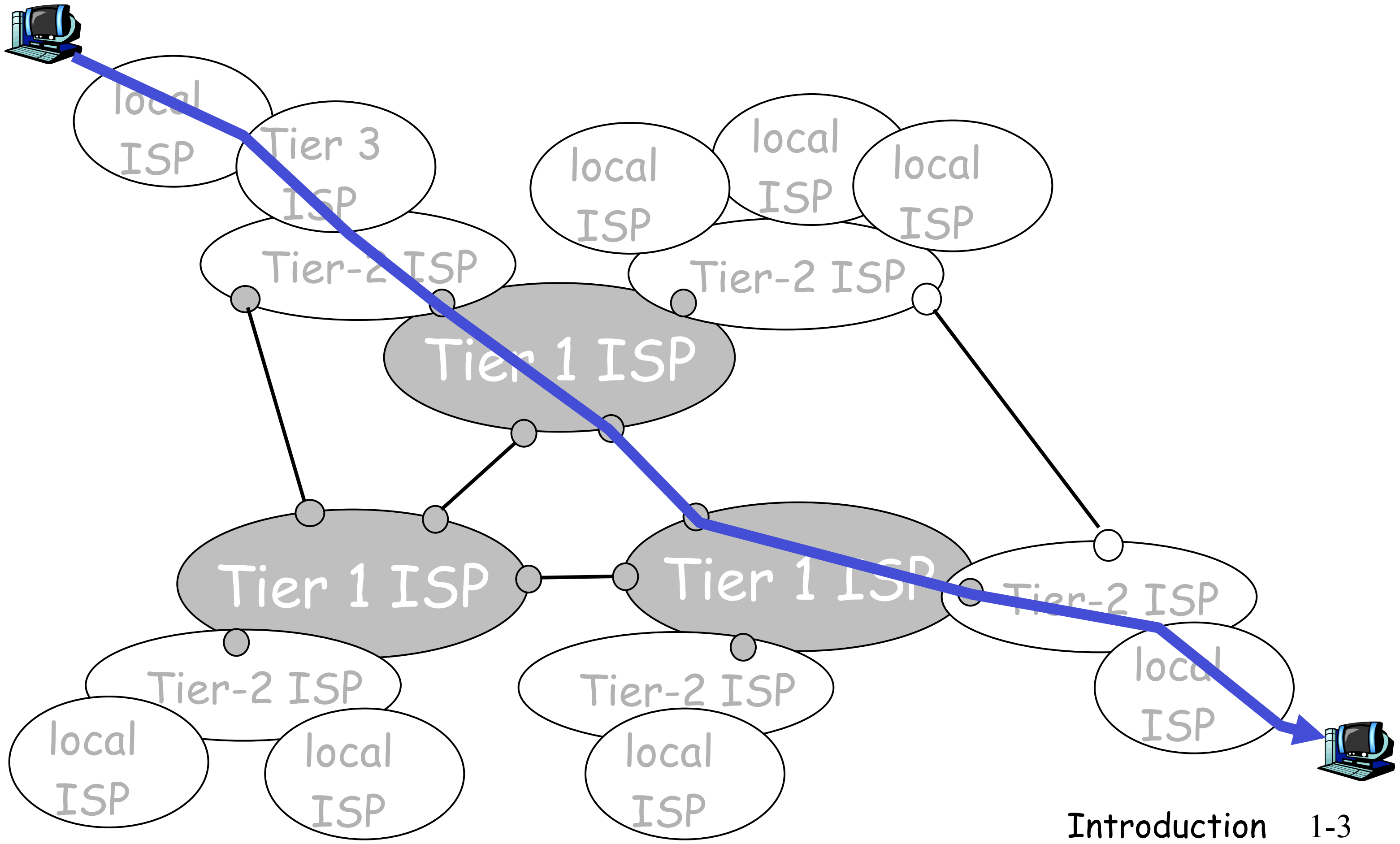
Application Layer

Based on: Computer Networking: A Top Down Approach, 4th edition. Jim Kurose, Keith Ross Addison-Wesley, July 2007.

OVERFLOW

Internet structure: network of networks

- a packet passes through many networks!



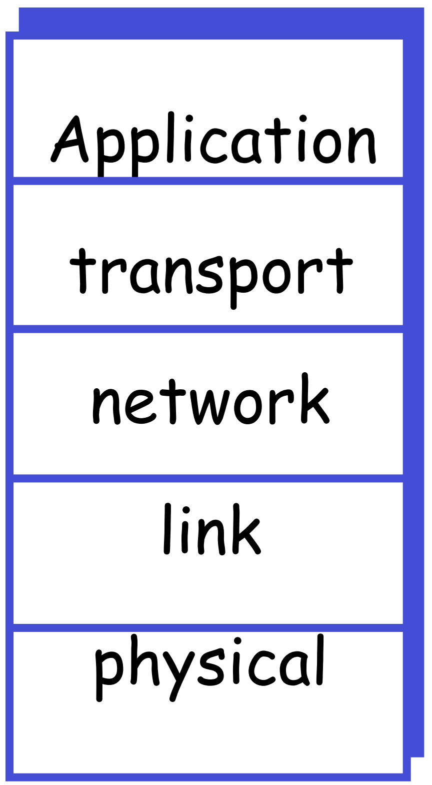
Why layering?

Dealing with complex systems:

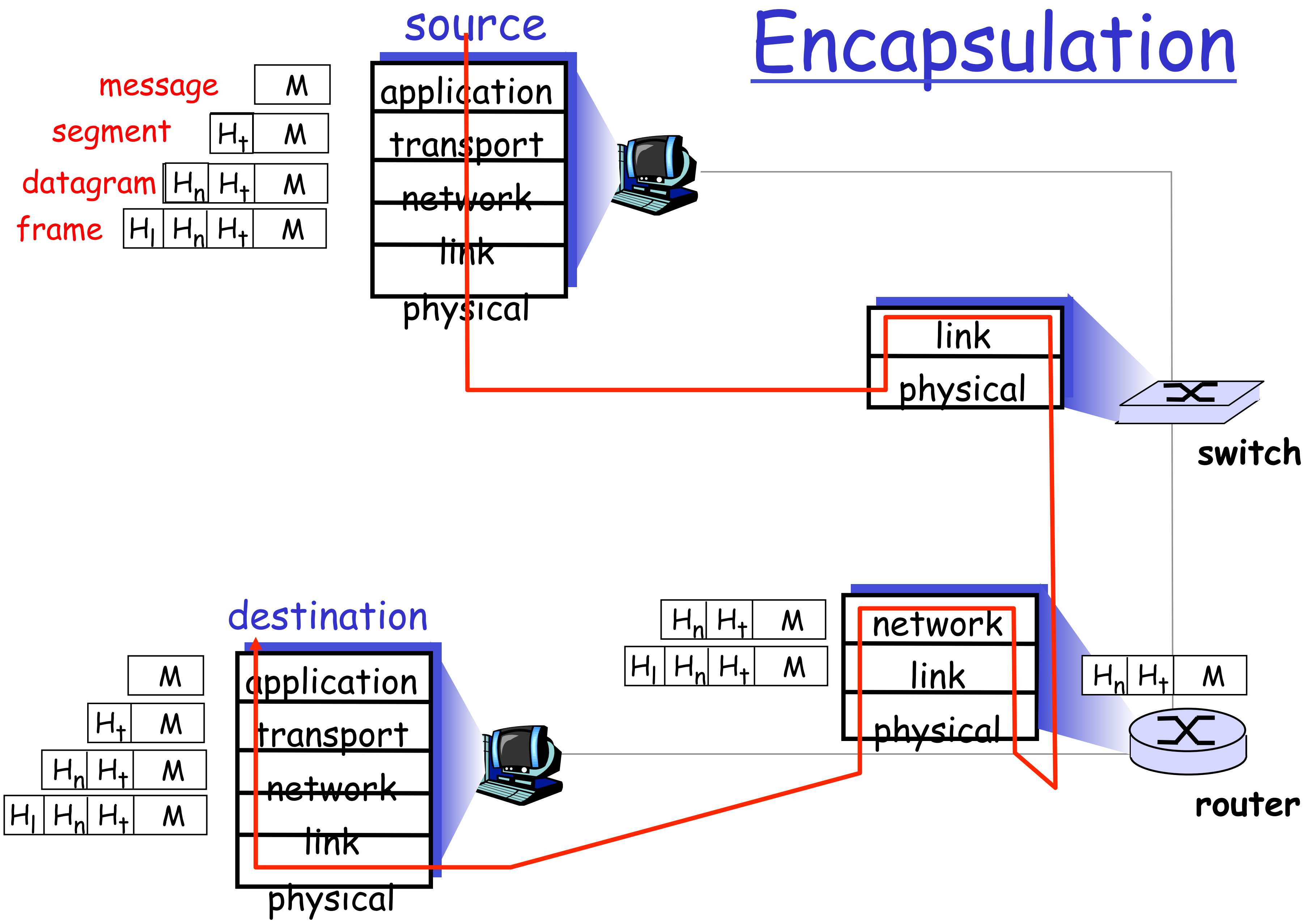
- explicit structure allows identification, relationship of complex system's pieces
  - layered reference model for discussion
- modularization eases maintenance, updating of system
  - change of implementation of layer's service transparent to rest of system
  - e.g., change in gate procedure doesn't affect rest of system
- layering considered harmful?

Internet protocol stack

- application: supporting network applications
  - FTP, SMTP, HTTP
- transport: process-process data transfer
  - TCP, UDP
- network: routing of datagrams from source to destination
  - IP, routing protocols
- link: data transfer between neighboring network elements
  - PPP, Ethernet
- physical: bits "on the wire"



Encapsulation



Application layer

- 2.1 Principles of network applications
- 2.2 Web and HTTP
- 2.3 FTP
- 2.4 ICCP
- 2.5 DNS

Some network apps

- e-mail
- Web
  - EMS
- instant messaging
- remote login
- P2P file sharing
- multi-user network games
- streaming stored phasor data
- voice over IP
- real-time video conferencing
- grid computing
- Databases
  - Historians

Processes communicating

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

Client process: process that initiates communication

Server process: process that waits to be contacted

- Note: applications with P2P architectures have client processes & server processes

Addressing processes

- to receive messages, process must have *identifier*
- host device has unique 32-bit IP address
- *Q:* does IP address of host suffice for identifying the process?

Addressing processes

- 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 ece.illinois.edu web server:
  - IP address: 130.126.112.22
  - Port number: 80
- more shortly…

Check your IP Address

- Windows
  - Start CMD
  - Run: **ipconfig**
  - Locate your network interface

- OS X/Linux
  - Start terminal
  - Run: **ifconfig**
  - Locate your network interface

App-layer protocol defines

- Types of messages exchanged,
  - e.g., request, response
- Message syntax:
  - what fields in messages & how fields are delineated
- Message semantics
  - meaning of information in fields
- Rules for when and how processes send & respond to messages

Public-domain protocols:

- defined in RFCs
- allows for interoperability
- e.g., HTTP, SMTP

Power systems protocols:

- DNP3, ICCP
- ANSI C12.22

Proprietary protocols:

- e.g., Skype

What transport service does an app need?

Data loss

- some apps (e.g., audio) can tolerate some loss
- other apps (e.g., file transfer, telnet) require 100% reliable data transfer

Timing

- some apps (e.g., VoIP, online games) require low delay to be "effective"

Throughput

- some apps (e.g., multimedia) require minimum amount of throughput to be "effective"
- other apps ("elastic apps") make use of whatever throughput they get

Security

- Encryption, data integrity, …

Transport service requirements of common apps

Application	Data loss	Throughput	Time Sensitive
file transfer	no loss	elastic	no
e-mail	no loss	elastic	no
Web documents	no loss	elastic	no
real-time audio/video	loss-tolerant	audio: 5kbps-1Mbps video:10kbps-5Mbps	yes, 100's msec
stored audio/video	loss-tolerant	same as above	
interactive games	loss-tolerant	few kbps up	yes, few secs
instant messaging	no loss	elastic	yes, 100's msec
			yes and no

Application layer

- 2.1 Principles of network applications
- 2.2 Web and HTTP
- 2.3 FTP
- 2.4 ICCP
- 2.5 DNS

Web and HTTP

First some jargon

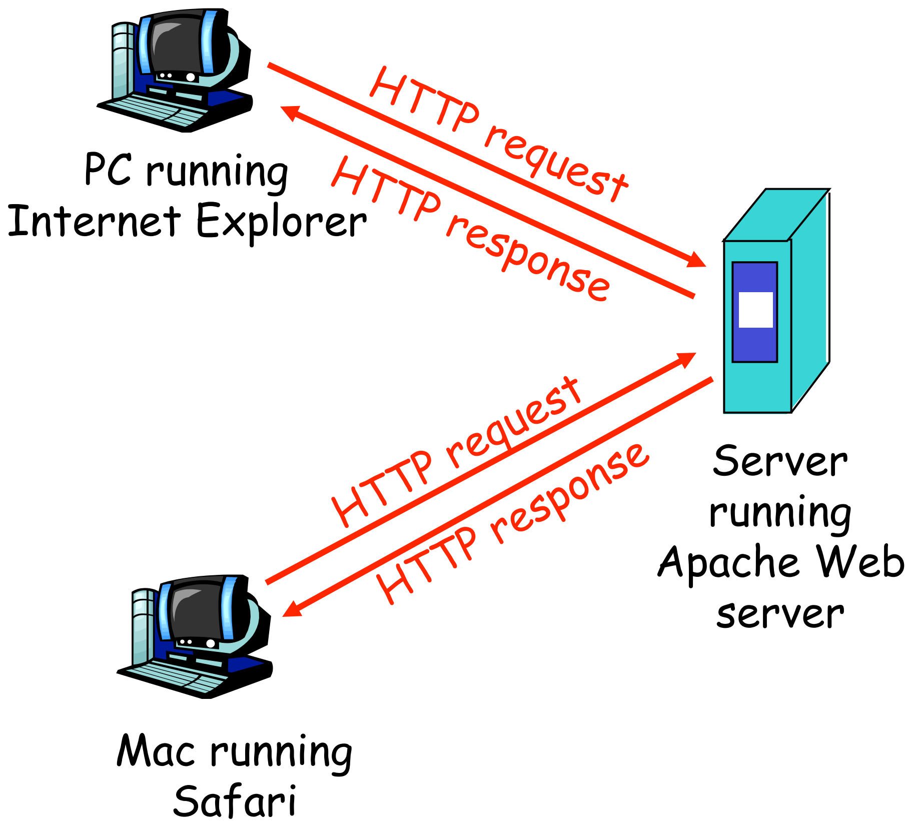
- Web page consists of objects
- Object can be HTML file, JPEG image, Video file, flash objects, audio file,…
- Web page consists of base HTML-file which includes several referenced objects
- Each object is addressable by a URL
- Example URL:

$$\underbrace{\texttt{www.someschool.edu}}_{\text{host name}}\texttt{/}\underbrace{\texttt{someDept/pic.gif}}_{\text{path name}}$$

HTTP overview

HTTP: hypertext transfer protocol

- 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



HTTP

Suppose user enters URL `www.someSchool.edu/someDepartment/home.index`

(contains text, references to 10 jpeg images)

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

1b. HTTP server at host www.someSchool.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 someDepartment/home.index

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

time

HTTP (cont.)

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

6. Steps 1-5 repeated for each of 10 jpeg objects

HTTP request message

- two types of HTTP messages: *request, response*
- HTTP request message:
  - ASCII (human-readable format)

request line (GET, POST, HEAD commands)

header lines

Carriage return, line feed indicates end of message

```
GET /somedir/page.html HTTP/1.1
Host: www.someschool.edu
User-agent: Mozilla/4.0
Connection: close
Accept-language:fr

(extra carriage return, line feed)
```

Uploading form input

Post method:

- Web page often includes form input
- Input is uploaded to server in entity body

URL method:

- Uses GET method
- Input is uploaded in URL field of request line:

```
www.somesite.com/animalsearch?monkeys&banana
```

HTTP response message

status line
(protocol
status code
status phrase)

header
lines

data, e.g.,
requested
HTML file

```
HTTP/1.1 200 OK
Connection close
Date: Thu, 06 Aug 1998 12:00:15 GMT
Server: Apache/1.3.0 (Unix)
Last-Modified: Mon, 22 Jun 1998 …...
Content-Length: 6821
Content-Type: text/html

data data data data data ...
```

Test HTTP

- Wireshark
  - Start wireshark and start capturing packets
  - Open any browser and go to
    http://tcipg.org/tcipg-reading-group-fall-2012
  - Stop capture
- 1. Filter packets by HTTP
  - Notice the HTTP request sent and response which contains the HTML page content
- 2. Filter packets by TCP
  - Notice the established TCP connection
  - Notice the port number used

Trying out HTTP (client side) for yourself

1. Telnet to your favorite Web server:

```
telnet www.crhc.illinois.edu 80
```

Opens TCP connection to port 80
www.crhc.illinois.edu
Anything typed in sent
to port 80 at www.crhc.illinois.edu

2. Type in a GET HTTP request:

```
GET /Faculty/whs.html HTTP/1.1
Host: www.crhc.illinois.edu
```

By typing this in (hit carriage return twice), you send this minimal (but complete) GET request to HTTP server

3. Look at response message sent by HTTP server!

Exercise

- HTTPS
  - HTTPS stands for
    - Hypertext Transfer Protocol Secure
  - Same process but open https://www.google.com
- Set Filter to HTTPS
  - Try to find the request and response packets

User-server state: cookies

HTTP is stateless

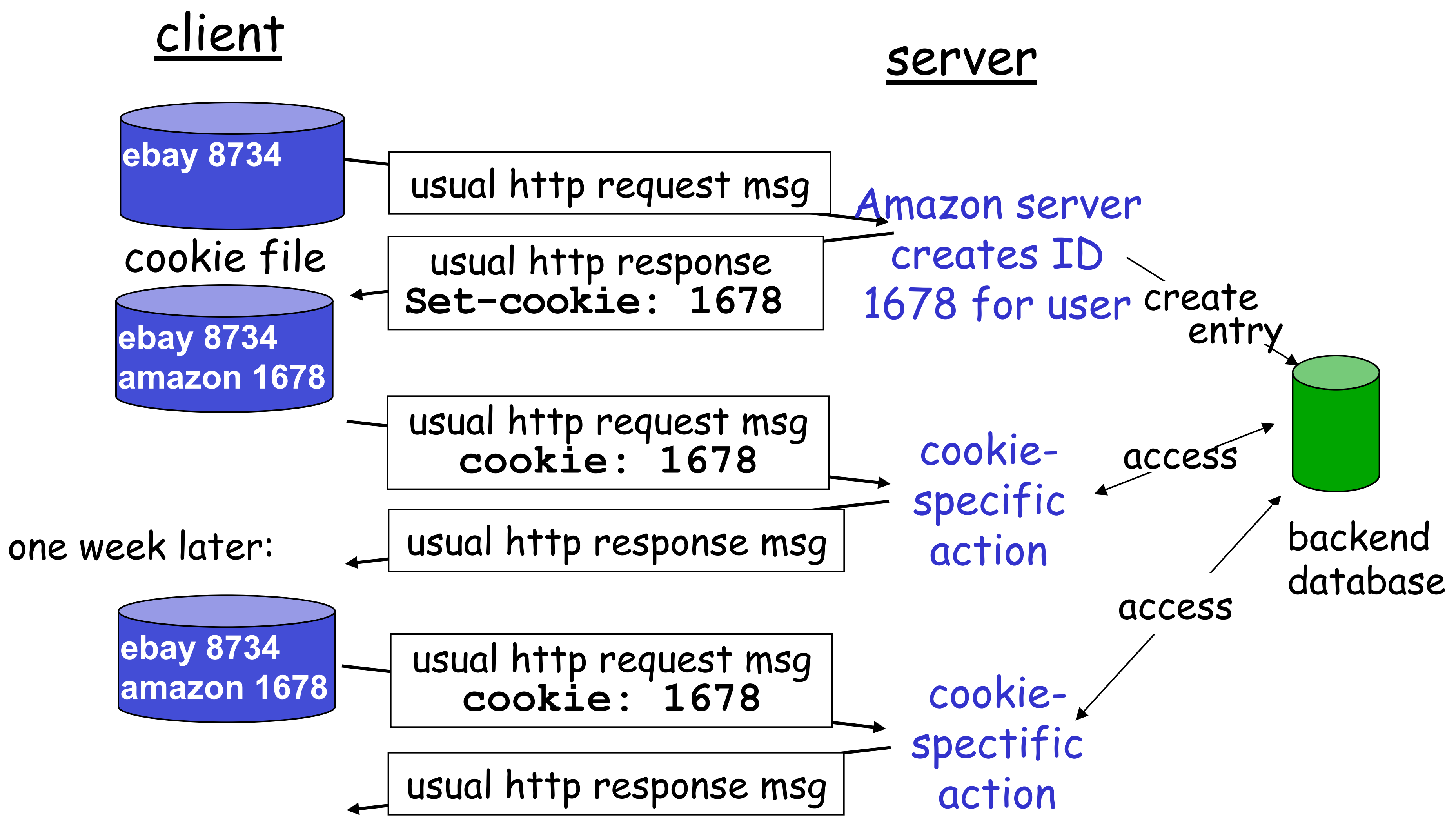
Cookies are the answer:

- protocol endpoints: maintain state at sender/receiver over multiple transactions
- cookies: http messages carry state

What cookies can bring:

- authorization
- shopping carts
- recommendations
- user session state (Web e-mail)

Cookies: keeping “state” (cont.)



Check your cookies

- Chrome
  - Open:

    chrome://chrome/settings/cookies
- Firefox:
  - Preferences -> Privacy -> Show Cookies
- Safari:
  - Preferences -> Privacy -> Cookies and other website data -> details

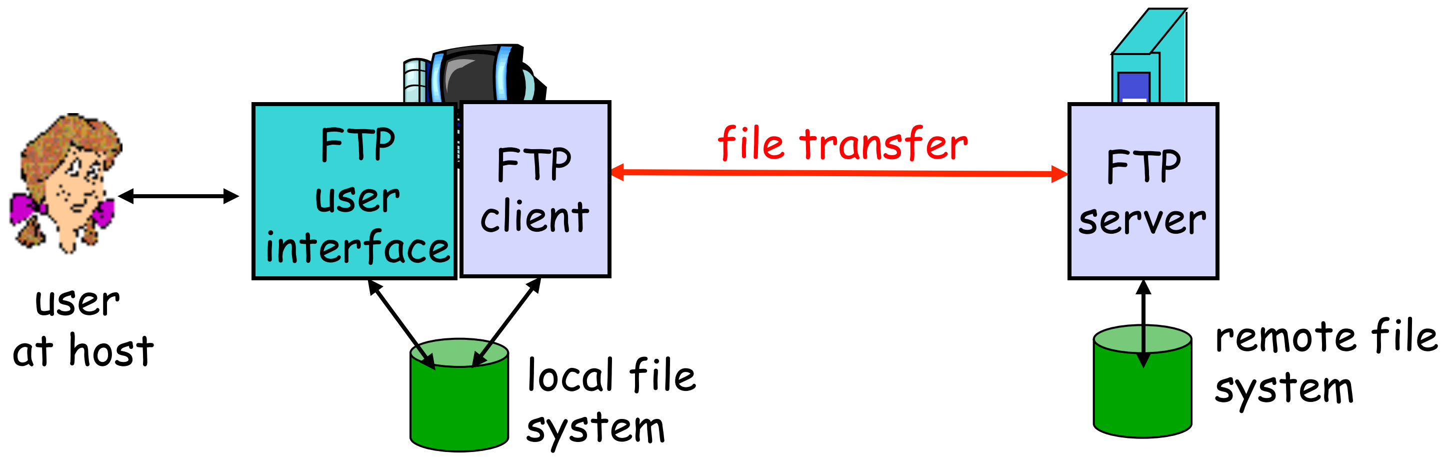
Security and Privacy Issues

- Storing Personal Information
- Tracking User Behavior
  - Google ads
  - ...
- Session hijacking
  - HTTP based authentication/session cookies

Application layer

- 2.1 Principles of network applications
- 2.2 Web and HTTP
- 2.3 FTP
- 2.4 ICCP
- 2.5 DNS

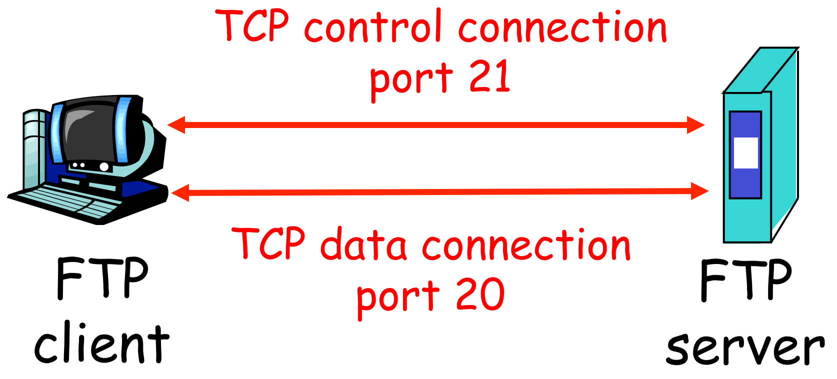
FTP: the file transfer protocol



- transfer file to/from remote host
- client/server model
  - *client:* side that initiates transfer (either to/from remote)
  - *server:* remote host
- ftp: RFC 959
- ftp server: port 21

FTP: separate control, data connections

- FTP client contacts FTP server at port 21
- TCP is transport protocol
- client authorized over control connection
- when server receives file transfer command, server opens *2nd* TCP connection (for file) to client



Application layer

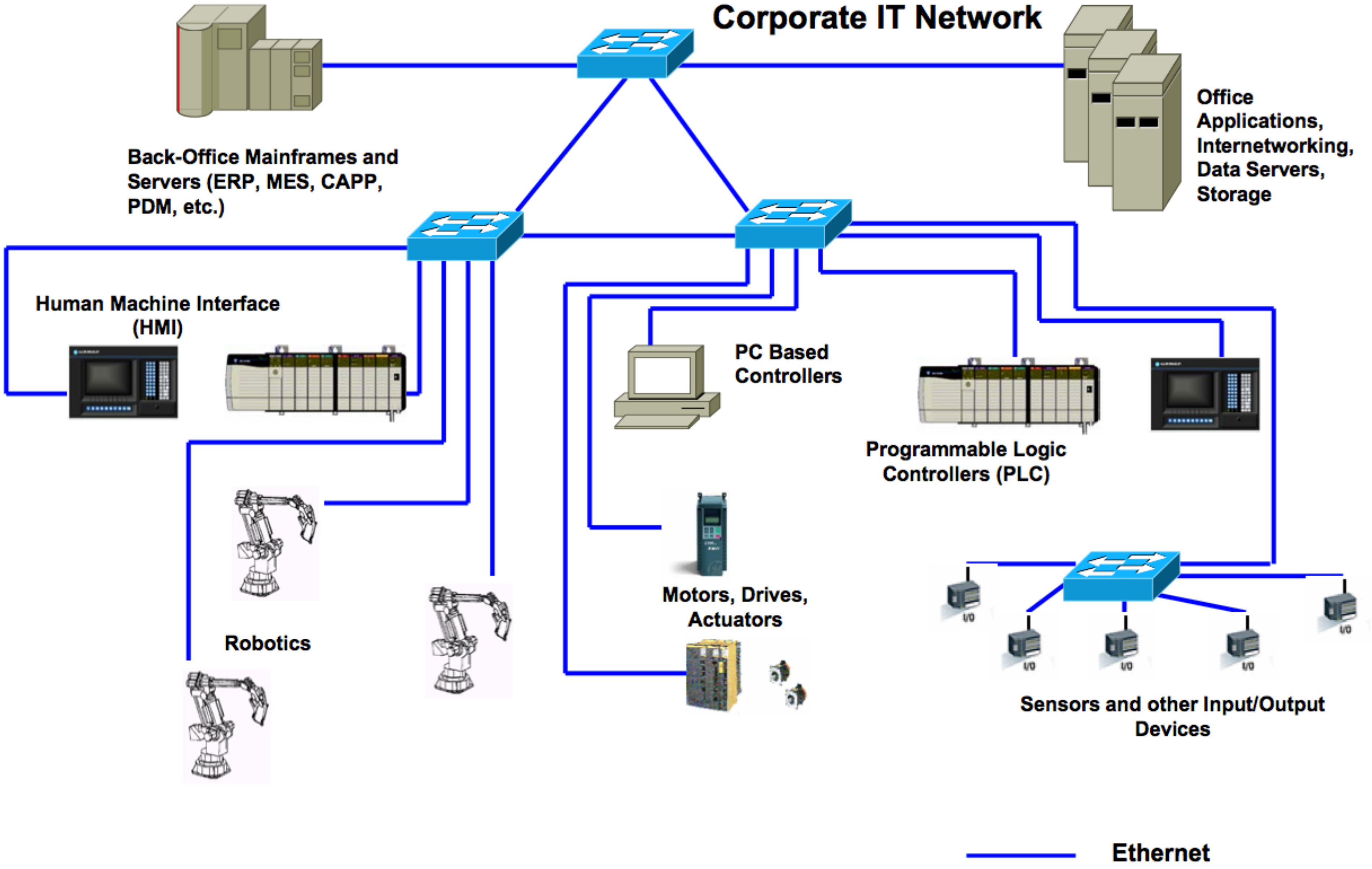
- 2.1 Principles of network applications
- 2.2 Web and HTTP
- 2.3 FTP
- 2.4 ICCP
- 2.5 DNS

ICCP

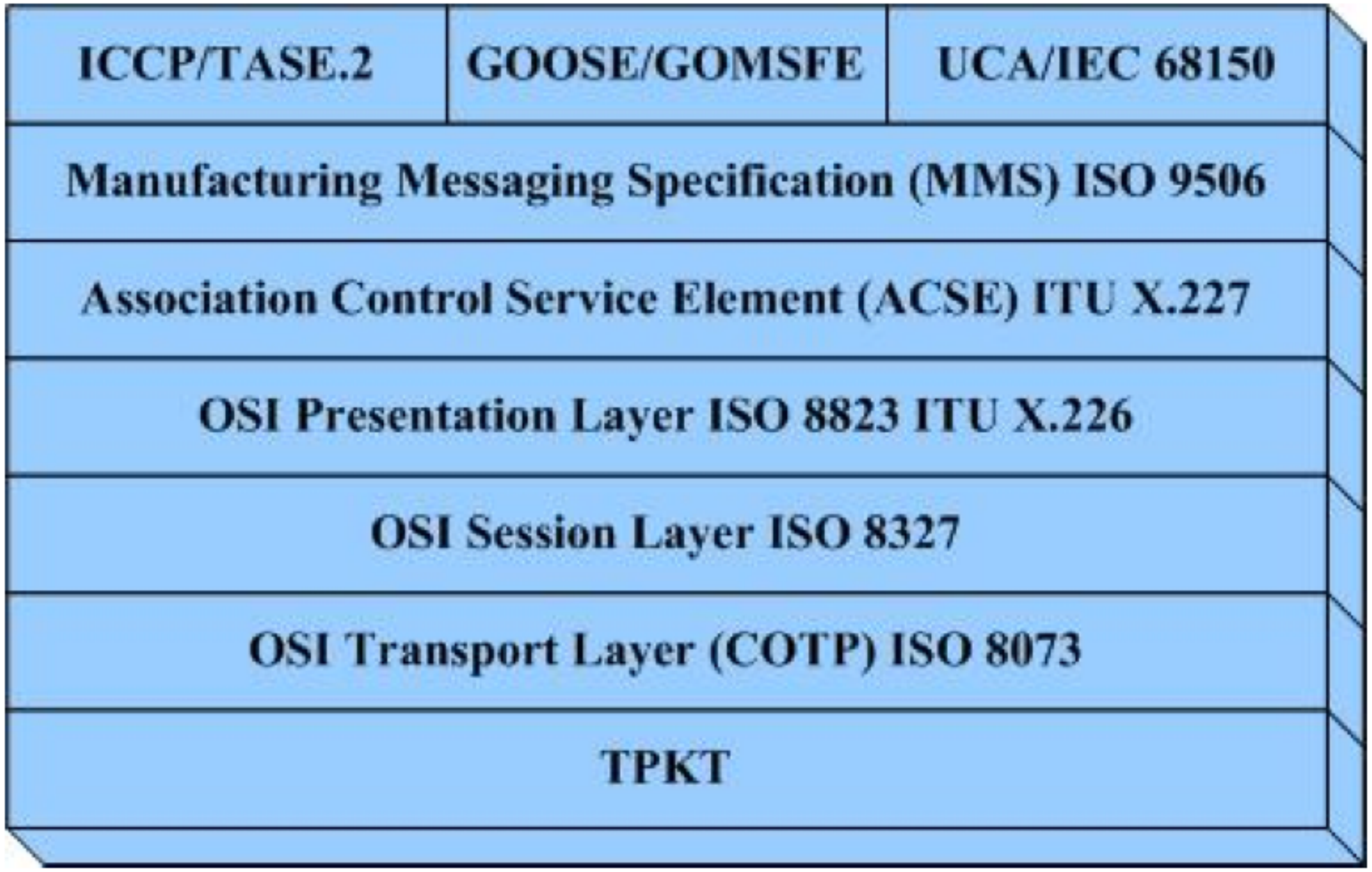
- Inter-Control Center Communications Protocol
- Developed by Electric Power Research Institute(EPRI) and Northern States Power (NSP)
- A real-time data exchange standard for transferring the information required to the interconnected electrical systems operation

- The information consists of real-time power system monitoring and control data
  - measured values, scheduling data, energy,
  - accounting data and operator messages.
- Supports the control of programs at a remote control center.

SCADA Network


Corporate IT Network
Back-Office Mainframes and Servers (ERP, MES, CAPP, PDM, etc.)
Office Applications, Internetworking, Data Servers, Storage
Human Machine Interface (HMI)
PC Based Controllers
Programmable Logic Controllers (PLC)
Robotics
Motors, Drives, Actuators
I/O
I/O
I/O
I/O
I/O
Sensors and other Input/Output Devices
Ethernet

ICCP Protocol Stack


ICCP/TASE.2
GOOSE/GOMSFE
UCA/IEC 68150
Manufacturing Messaging Specification (MMS) ISO 9506
Association Control Service Element (ACSE) ITU X.227
OSI Presentation Layer ISO 8823 ITU X.226
OSI Session Layer ISO 8327
OSI Transport Layer (COTP) ISO 8073
TPKT

ICCP Access Control

- Associations – Connections are relations between control centers
- ICCP does not provide authentication or encryption
- A Bilateral Table represents the agreement between two control centers connected with an ICCP link.
    - identifies data elements and objects that can be accessed

Application layer

- 2.1 Principles of network applications
- 2.2 Web and HTTP
- 2.3 FTP
- 2.4 ICCP
- 2.5 DNS

DNS: Domain Name System

People: many identifiers:
- SSN, name, passport #

Internet hosts, routers:
- IP address (32 bit) - used for addressing datagrams
- "name", e.g., ww.yahoo.com - used by humans

Q: map between IP addresses and name ?

Domain Name System:
- *distributed database* implemented in hierarchy of many *name servers*
- *application-layer protocol* host, routers, name servers to communicate to *resolve* names (address/name translation)
  - note: core Internet function, implemented as application-layer protocol
  - complexity at network's "edge"

DNS

DNS services

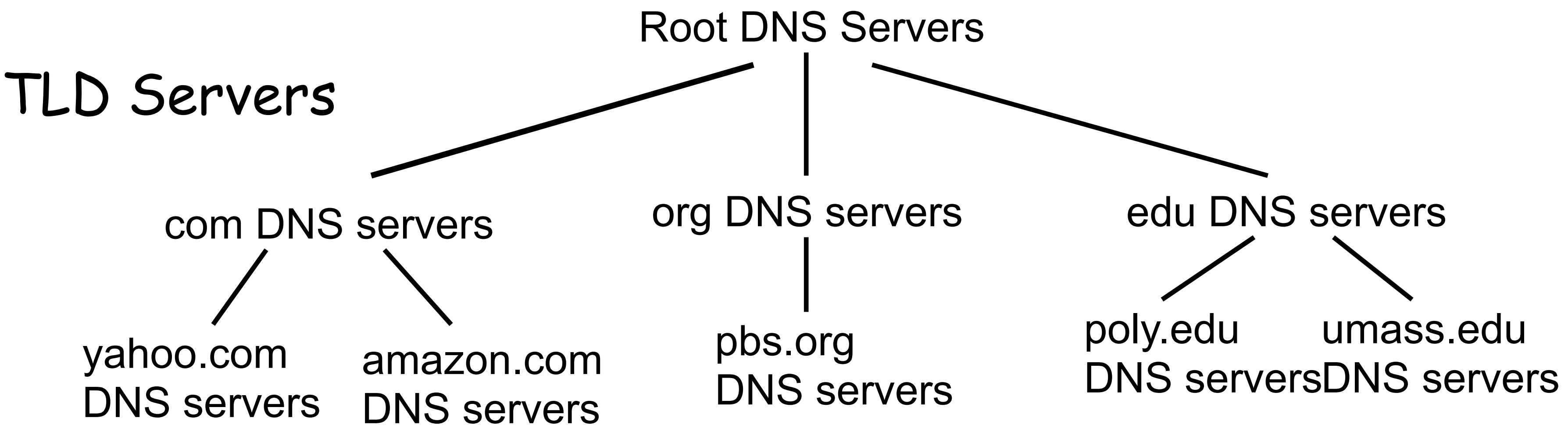
- hostname to IP address translation
- host aliasing
  - Canonical, alias names
- mail server aliasing
- load distribution
  - replicated Web servers: set of IP addresses for one canonical name

Why not centralize DNS?

- single point of failure
- traffic volume
- distant centralized database
- maintenance

doesn't *scale!*

Distributed, Hierarchical Database



Client wants IP for www.amazon.com; 1st approx:

- client queries a root server to find com DNS server
- client queries com DNS server to get amazon.com DNS server
- client queries amazon.com DNS server to get IP address for www.amazon.com

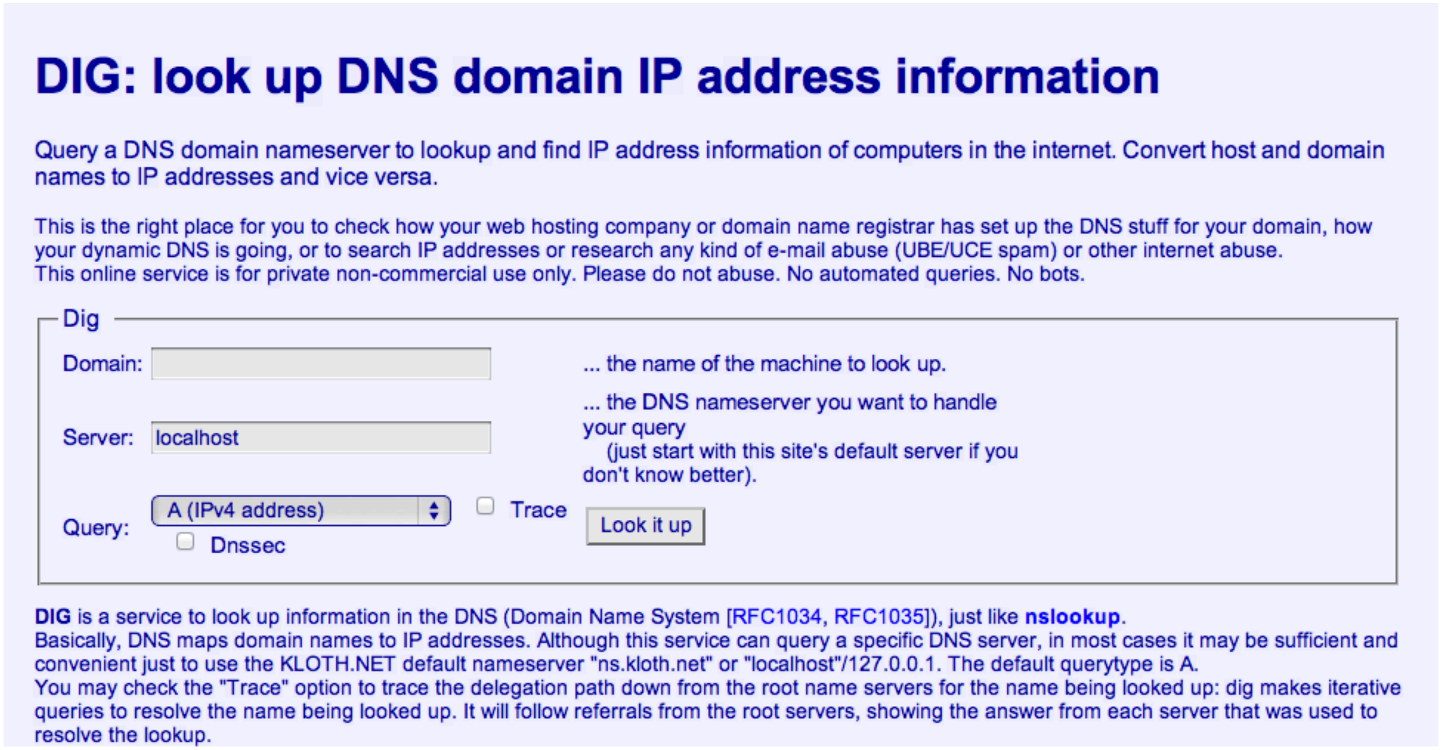
Local Name Server

- does not strictly belong to hierarchy
- each ISP (residential ISP, company, university) has one.
  - also called “default name server”
- when host makes DNS query, query is sent to its local DNS server
  - acts as proxy, forwards query into hierarchy
- Windows: ipconfig \all
- Linux/Mac: cat /etc/resolv.conf

Exercise

Online DNS resolution tool

- http://www.kloth.net/services/dig.php



DIG: look up DNS domain IP address information

Query a DNS domain nameserver to lookup and find IP address information of computers in the internet. Convert host and domain names to IP addresses and vice versa.

This is the right place for you to check how your web hosting company or domain name registrar has set up the DNS stuff for your domain, how your dynamic DNS is going, or to search IP addresses or research any kind of e-mail abuse (UBE/UCE spam) or other internet abuse.
This online service is for private non-commercial use only. Please do not abuse. No automated queries. No bots.

Dig

Domain: ... the name of the machine to look up.

Server: localhost ... the DNS nameserver you want to handle your query (just start with this site's default server if you don't know better).

Query: A (IPv4 address) ☐ Trace ☐ Dnssec

DIG is a service to look up information in the DNS (Domain Name System [RFC1034, RFC1035]), just like **nslookup**.
Basically, DNS maps domain names to IP addresses. Although this service can query a specific DNS server, in most cases it may be sufficient and convenient just to use the KLOTH.NET default nameserver "ns.kloth.net" or "localhost"/127.0.0.1. The default querytype is A.
You may check the "Trace" option to trace the delegation path down from the root name servers for the name being looked up: dig makes iterative queries to resolve the name being looked up. It will follow referrals from the root servers, showing the answer from each server that was used to resolve the lookup.

Exercise

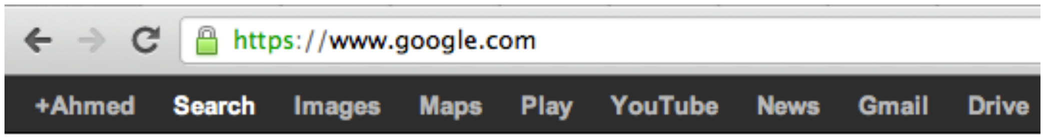
- Test with different domains, servers, and record types (while selecting trace)
  - google.com, …
  - Illinois.edu, …
  - Doe.gov, …
  - Apple.co.uk, …
- Observe the different TLD servers reached
- Try different server (8.8.8.8)

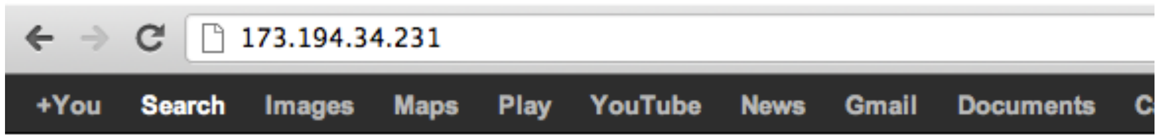
Linux/Windows Tools

- Command line tools to test on your own
  - nslookup
  - dig
  - host

DNS continued

- 1. Get the IP address of google.com
- 2. Use the IP to access the website through the browser
- The browser is essentially doing the same process


https://www.google.com


173.194.34.231

DNS Cache

- Your DNS cache stores the locations (IP addresses) of pages you have recently viewed.
- We will flush the cache to ensure we get a response
- OS X:
  - dscacheutil –flushcache
  - sudo killall -HUP mDNSResponder (Lion+)
- Windows:
  - ipconfig /flushdns
- Linux:
  - /etc/rc.d/init.d/nscd restart

Wireshark and DNS

- Flush dns cache
- Start packet capture using wireshark
- Set wireshark's display filter to DNS
- Use nslookup to find an address of illinois.edu

  ```
  >nslookup
  >server 8.8.8.8
  ```

  - (configures dns server to 8.8.8.8)

  ```
  >Illinois.edu
  ```

- Stop wireshark capture
- Inspect query and response

Questions

- Locate the DNS query and response messages. Are they sent over UDP or TCP?
- What is the destination port for the DNS query message? What is the source port of DNS response message?
- To what IP address is the DNS query message sent?
- Examine the DNS query message. What "Type" of DNS query is it? Does the query message contain any "answers"?
- Examine the DNS response message. How many "answers" are provided? What does each of these answers contain?

Wireshark and Browsing

- Start capture and open your browser
- Go to: www.google.com
- Stop capture
- Questions?
  - Consider the subsequent TCP SYN packet sent by your host. Does the destination IP address of the SYN packet correspond to any of the IP addresses provided in the DNS response message?
  - This web page contains images. Before retrieving each image, does your host issue new DNS queries?