Computer Science 425
Distributed Systems

CS 425 / ECE 428

Fall 2013

Indranil Gupta (Indy)

August 27, 2013

Lecture 1

Website: http://courses.engr.illinois.edu/cs425

Our Main Goal Today

To Define the Term Distributed System

Can you name some examples of Operating Systems?

Can you name some examples of Operating Systems?

…

Linux WinXP Vista Unix FreeBSD Mac OSX

2K Aegis Scout Hydra Mach SPIN

OS/2 Express Flux Hope Spring

AntaresOS EOS LOS SQOS LittleOS TINOS

PalmOS WinCE TinyOS iOS

…

What is an Operating System?

What is an Operating System?

- **User interface to hardware (device driver)**
- **Provides abstractions (processes, file system)**
- **Resource manager (scheduler)**
- **Means of communication (networking)**
- **…**

FOLDOC definition

(FOLDOC = Free On-Line Dictionary of Computing)

Operating System - The low-level software which handles the interface to peripheral hardware, schedules tasks, allocates storage, and presents a default interface to the user when no application program is running.

Can you name some examples of Distributed Systems?

Can you name some examples of Distributed Systems?

- Client-Server (NFS)
- The Web
- The Internet
- A wireless network
- DNS
- Gnutella or BitTorrent (peer to peer overlays)
- A “cloud”, e.g., Amazon EC2/S3, Microsoft Azure
- A datacenter, e.g., NCSA, a Google datacenter, The Planet

What is a Distributed System?

FOLDOC definition

A collection of (probably heterogeneous) automata whose distribution is transparent to the user so that the system appears as one local machine. This is in contrast to a network, where the user is aware that there are several machines, and their location, storage replication, load balancing and functionality is not transparent. Distributed systems usually use some kind of client-server organization.

Textbook definitions

- **A distributed system is a collection of independent computers that appear to the users of the system as a single computer.**

  [Andrew Tanenbaum]

- **A distributed system is several computers doing something together. Thus, a distributed system has three primary characteristics: multiple computers, interconnections, and shared state.**

  [Michael Schroeder]

Unsatisfactory

- Why are these definitions short?
- Why do these definitions look inadequate to us?
- Because we are interested in the insides of a distributed system
  - design and implementation
  - Maintenance
  - Algorithmics ("protocols")

“***I shall not today attempt further to define*** ***the kinds of material I understand to be embraced within that shorthand description; and perhaps I could never succeed in intelligibly doing so. But I know it when I see it, and the motion picture involved in this case is not that.***”

[Potter Stewart, Associate Justice, US Supreme Court (talking about his interpretation of a technical term laid down in the law, case Jacobellis versus Ohio 1964)]

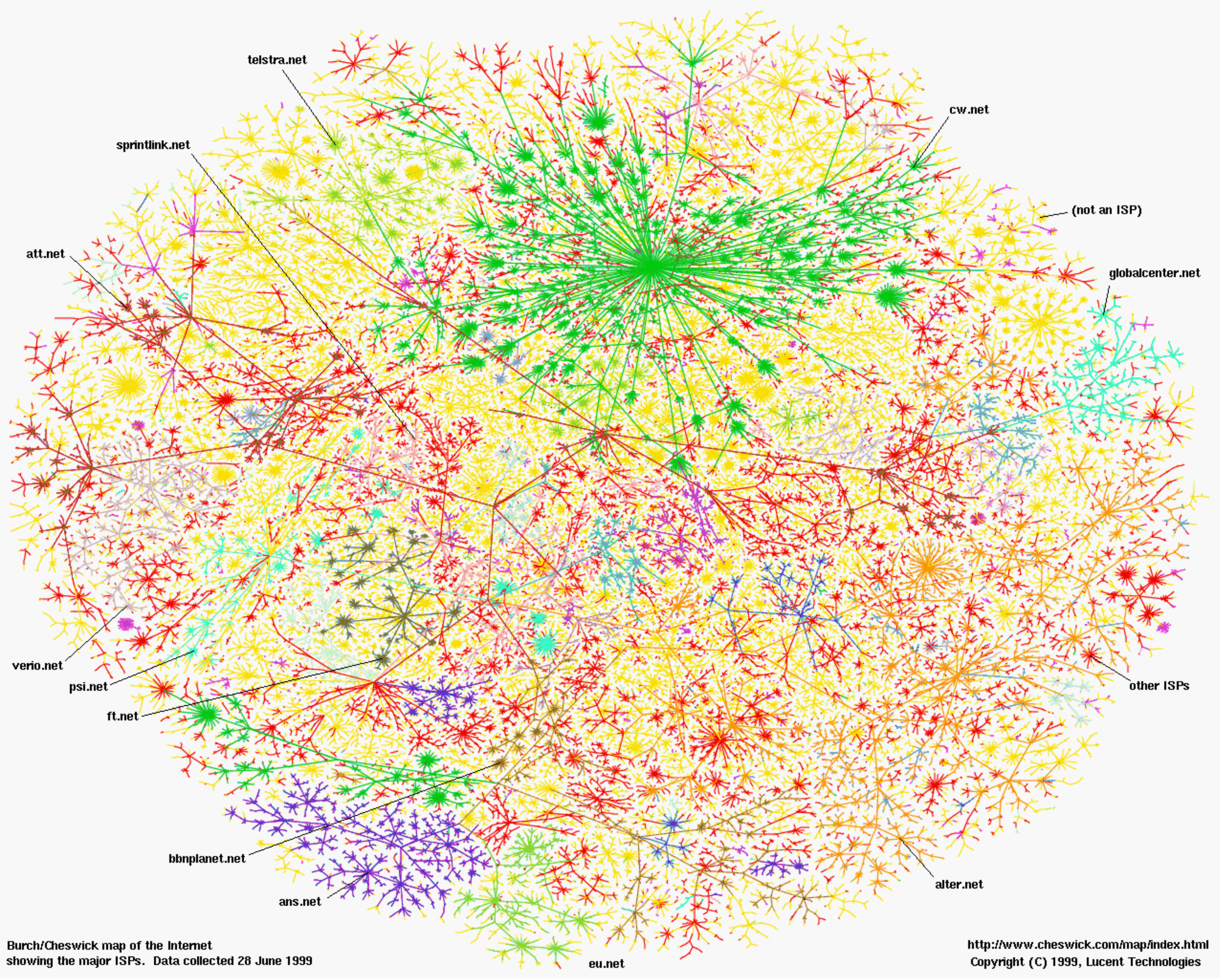
Which is a Distributed System – (A) *or* (B)?

(A)

(A) Facebook Social Network Graph among humans

Source: https://www.facebook.com/note.php?note_id=469716398919

(B)



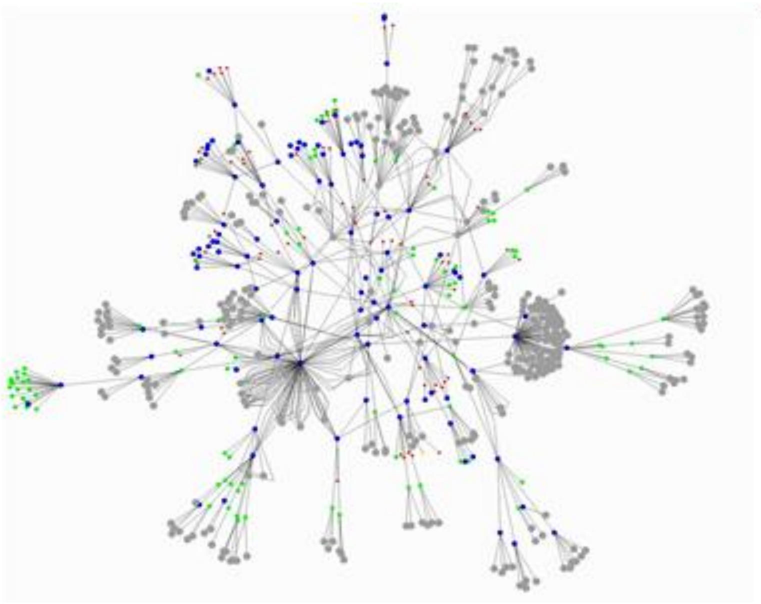
(B) The Internet (Internet Mapping Project, color coded by ISPs)

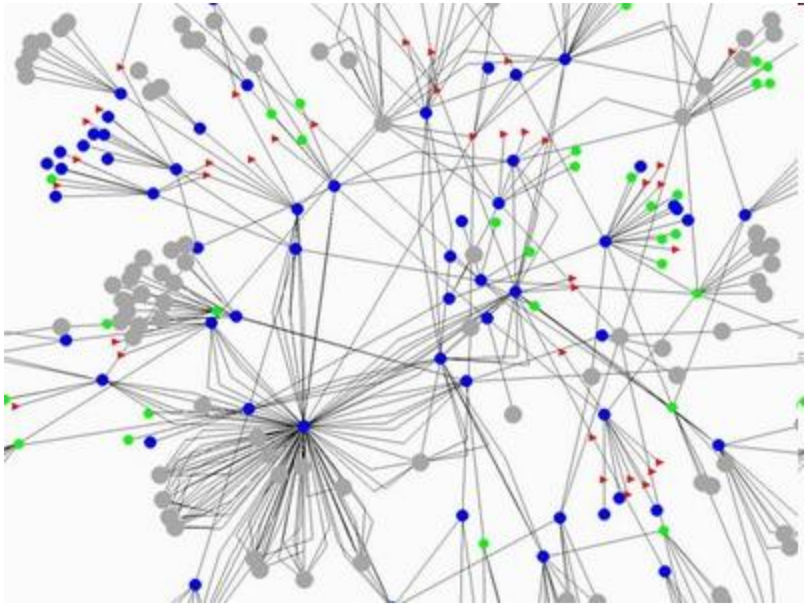
A working definition for us

A distributed system is a collection of entities, each of which is autonomous, programmable, asynchronous and failure-prone, and which communicate through an unreliable communication medium.

- **Entity=a process on a device (PC, PDA)**
- **Communication Medium=Wired or wireless network**
- **Our interest in distributed systems involves**
  - **design and implementation, maintenance, algorithmics**

Distributed System Example – Gnutella Peer to Peer System





What are the "entities" (nodes)?

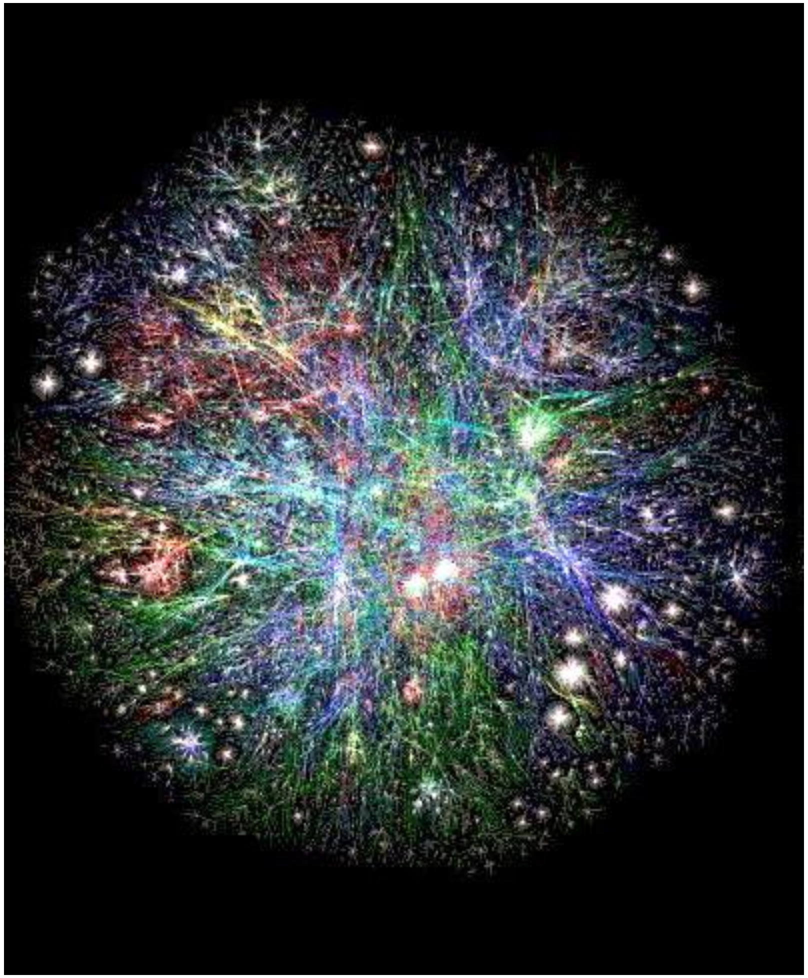
What is the communication medium (links)?

Source: GnuMap Project

Distributed System Example – Web domains

What are the "entities" (nodes)?

What is the communication medium (links)?



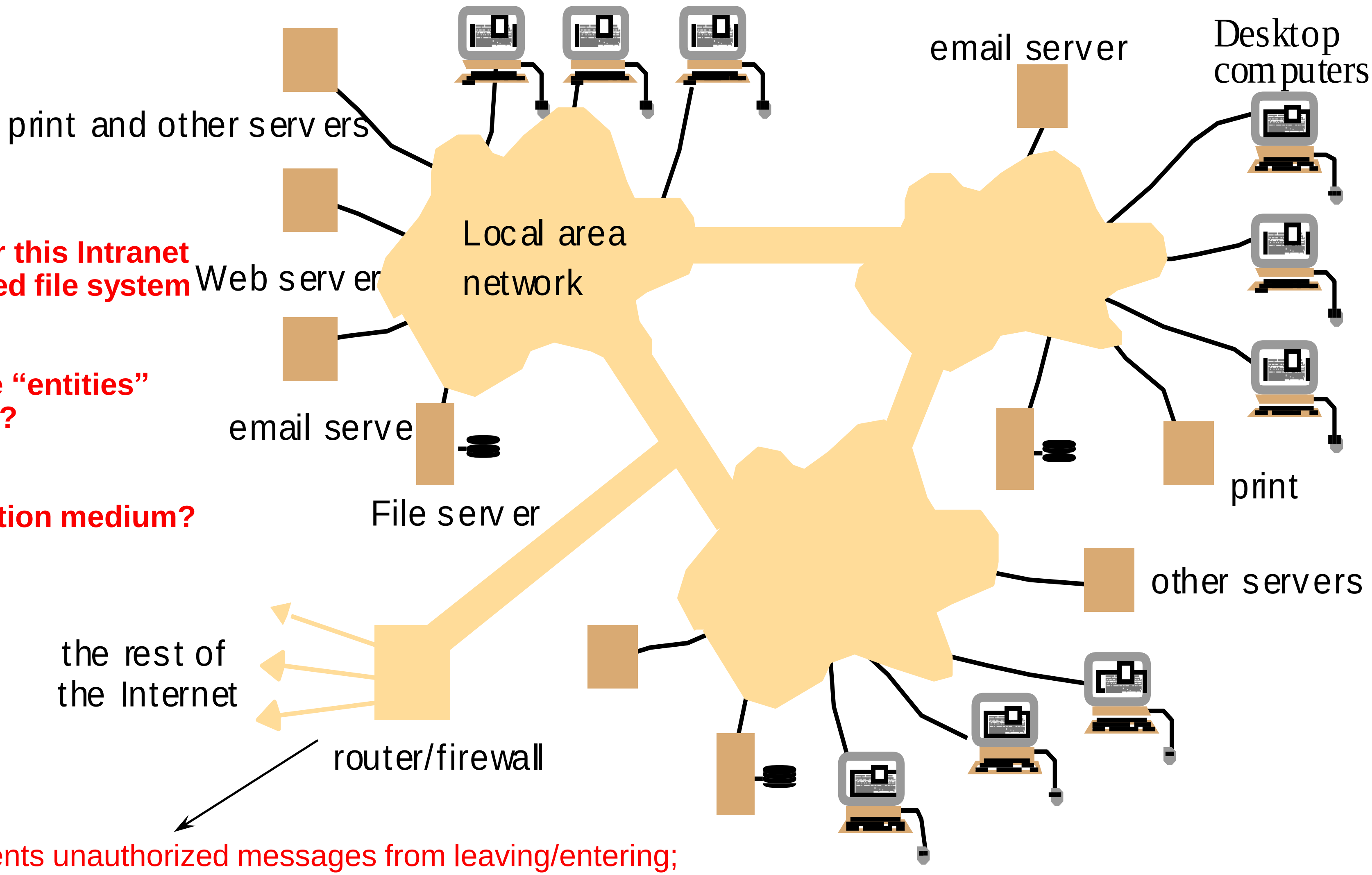
Source: http://www.vlib.us/web/worldwideweb3d.html

Distributed System Example – A Datacenter

The Internet – Quick Refresher

- **Underlies many distributed systems.**
- **A vast interconnected collection of computer networks of many types.**
- **<u>Intranets</u> – subnetworks operated by companies and organizations.**
- **Intranets contain subnets and LANs.**
- **WAN – wide area networks, consists of LANs**
- **ISPs – companies that provide modem links and other types of connections to users.**
- **Intranets (actually the ISPs' core routers) are linked by <u>backbones</u> – network links of large bandwidth, such as satellite connections, fiber optic cables, and other high-bandwidth circuits.**
- **UC2B? Google Fiber?**

An Intranet and a distributed system over it



Running over this Intranet is a distributed file system -

- **What are the "entities" (nodes) in it?**
- **What is the communication medium?**

prevents unauthorized messages from leaving/entering; implemented by filtering incoming and outgoing messages

Distributed Systems are layered over networks

Application	Application layer protocol	Underlying transport protocol
e-mail	smtp [RFC 821]	TCP
remote terminal access	telnet [RFC 854]	TCP
Web	http [RFC 2068]	TCP
file transfer	ftp [RFC 959]	TCP
streaming multimedia	proprietary (e.g. RealNetworks)	TCP or UDP
remote file server	NFS	TCP or UDP
Internet telephony	proprietary (e.g., Skype)	typically UDP

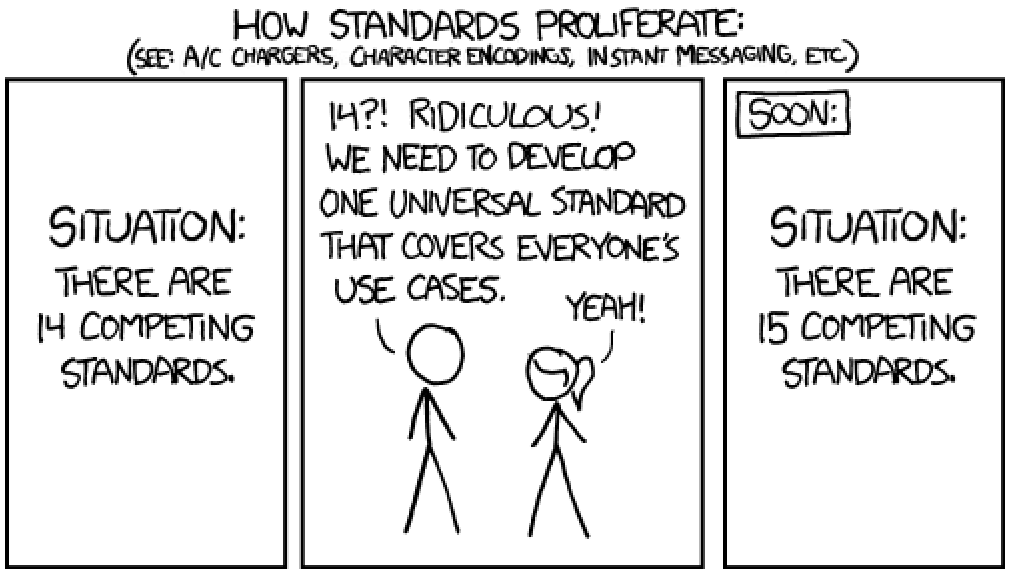
TCP=Transmission Control Protocol
UDP=User Datagram Protocol

Implemented via network "sockets". Basic primitive that allows machines to send messages to each other

Distributed System Protocols!

Networking Protocols

The History of Internet Standards

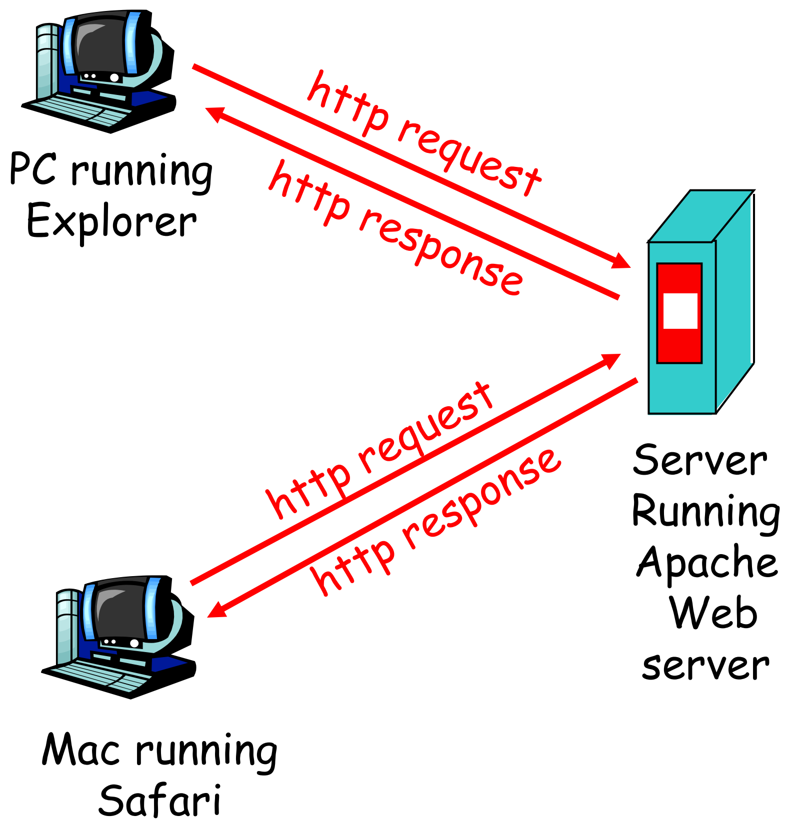


Source: http://xkcd.com/927/

The Secret of the World Wide Web: the HTTP Standard

HTTP: hypertext transfer protocol

- **WWW's application layer protocol**
- **client/server model**
  - ***client:*** **browser that requests, receives, and "displays" WWW objects**
  - ***server:*** **WWW server, which is storing the website, sends objects in response to requests**
- **http1.0: RFC 1945**
- **http1.1: RFC 2068**
  - **Leverages same connection to download images, scripts, etc.**



The HTTP Protocol: More

http: TCP transport service:

- client initiates a TCP connection (creates socket) to server, port 80
- server accepts the TCP connection from client
- http messages (application-layer protocol messages) exchanged between browser (http client) and WWW server (http server)
- TCP connection closed

http is "stateless"

- server maintains no information about past client requests

Why?

Protocols that maintain session "state" are complex!

- past history (state) must be maintained and updated.
- if server/client crashes, their views of "state" may be inconsistent, and hence must be reconciled.
- RESTful protocols are stateless.

HTTP Example

Suppose user enters URL www.cs.uiuc.edu/

(contains text, references to 10 jpeg images)

1a. **http client initiates a TCP connection to http server (process) at www.cs.uiuc.edu. Port 80 is default for http server.**

1b. **http server at host www.cs.uiuc.edu waiting for a TCP connection at port 80. "accepts" connection, notifying client**

2. **http client sends a http *request message* (containing URL) into TCP connection socket**

3. **http server receives request messages, forms a *response message* containing requested object (index.html), sends message into socket**

time

HTTP Example (cont.)

4. **http server closes the TCP connection (if necessary).**

5. **http client receives a response message containing html file, displays html, Parses html file, finds 10 referenced jpeg objects**

6. ***Steps 1-5 are then repeated for <u>each</u> of 10 jpeg objects***

time

For fetching referenced objects, have 2 options:

- **non-persistent connection:** **only one object fetched per TCP connection**
  - **some browsers create multiple TCP connections *simultaneously* - one per object**
- **persistent connection:** **multiple objects transferred within one TCP connection**

A human as a browser (Client Side)

1. Telnet to your favorite WWW server:

```
telnet www.google.com 80
```

Opens TCP connection to port 80 (default http server port) at www.google.com
Anything typed in sent
to port 80 at www.google.com

2. Type in a GET http request:

```
GET /index.html
Or
GET /index.html HTTP/1.0
```

By typing this in (may need to hit return twice), you send
this minimal (but complete)
GET request to http server

3. Look at response message sent by http server!

What do you think the response is?

Does our Working Definition work for the http Web?

A distributed system is a collection of entities, each of which is autonomous, programmable, asynchronous and failure-prone, and that communicate through an unreliable communication medium.

- **Entity=a process on a device (PC, PDA)**
- **Communication Medium=Wired or wireless network**
- **Our interest in distributed systems involves**
  - **design and implementation, maintenance, study, algorithmics**

"Important" Distributed Systems Issues

- No global clock; no single global notion of the correct time (asynchrony)
- Unpredictable failures of components: lack of response may be due to either failure of a network component, network path being down, or a computer crash (failure-prone, unreliable)
- Highly variable bandwidth: from 16Kbps (slow modems or Google Balloon) to Gbps (Internet2) to Tbps (in between DCs of same big company)
- Possibly large and variable latency: few ms to several seconds
- Large numbers of hosts: 2 to several million

There are a range of interesting problems for Distributed System designers

- Real distributed systems
  - Cloud Computing, Peer to peer systems, Hadoop, key-value stores/NoSQL, distributed file systems, sensor networks, measurements, graph processing, …
- Classical Problems
  - Failure detection, Asynchrony, Snapshots, Multicast, Consensus, Mutual Exclusion, Election, …
- Concurrency
  - RPCs, Concurrency Control, Replication Control, …
- Security
  - Byzantine Faults, …
- Others…

Typical Distributed Systems Design Goals

- **Common Goals:**
  - **Heterogeneity** – can the system handle a large variety of types of PCs and devices?
  - **Robustness** – is the system resilient to host crashes and failures, and to the network dropping messages?
  - **Availability** – are data+services always there for clients?
  - **Transparency** – can the system hide its internal workings from the users?
  - **Concurrency** – can the server handle multiple clients simultaneously?
  - **Efficiency** – is the service fast enough? Does it utilize 100% of all resources?
  - **Scalability** – can it handle 100 million nodes without degrading service? (nodes=clients and/or servers) How about 6 B? More?
  - **Security** – can the system withstand hacker attacks?
  - **Openness** – is the system extensible?

"Important" Issues

- **If you're already complaining that the list of topics we've discussed so far has been perplexing…**
  - **You're right!**
  - **It was meant to be (perplexing)**
- **The Goal for the Rest of the Course: see enough examples and learn enough concepts so these topics and issues will make sense**
  - **We will revisit many of these slides in the very last lecture of the course!**

“Concepts”?

- **Which of the following inventions do you think is the most important?**
  1. **Car**
  2. **Wheel**
  3. **Bicycle**

“What lies beneath?” Concepts!

How will We Learn?

Take a look at handout "Course Information and Schedule"

- **Text: Colouris, Dollimore, Kindberg and Blair (5th edition)**
- **Lectures**
- **Homeworks**
  - **Approx. one every two-three weeks**
  - **Solutions need to be typed, figures can be hand-drawn**
- **Programming assignments**
  - **Incremental, in 4 stages**
  - **We will build a real *cloud computing system*!**
    - » **A key-value store this semester!**
  - **MP groups (== 2 students)**
    - » **Pair with someone taking course for same number of hours (3 with 3, 4 with 4)**
  - **4 credit students have to complete "Extra Credit" portion on each MP**
- **Exams/quizzes**
  - **Midterm + Final**

On the Textbook

- **Text: Colouris, Dollimore, Kindberg and Blair (5th edition).**

- **We will refer to section, chapter, and problem numbers only in the 5th edition.**
  - **Older editions may have a different numbering for some HW problems (that we give from the textbook). Make sure you solve the right problem; the responsibility is yours (no points for solving the wrong problem!)**

What assistance is available to you?

Source: http://dilbert.com

- **Lectures**
  - **lecture slides will be placed online at course website**
    - » **"Tentative" version before lecture**
    - » **"Final" version after lecture**
- **Homeworks – office hours and discussion forum to help you (without giving you the solution).**
- **Programming Assignments (MPs) – You can use any language, but we recommend C/C++ and Java ("best projects" code will be released in these) – office hours and discussion forum to help you (without giving you the solution).**
  - **Last year, groups used C, C++, Java, and even Go.**
- **Course Prerequisite: CS 241 or ECE 391 or equivalent OS/networking course (latter need instructor permission)**

You can find us almost everywhere

In person (Office Hours):

- **Myself (Indy): 3112 Siebel Center**
  - **Every Tuesday and Thursday right after class – see website**
- **TAs:**
  - **Hilfi Alkaff: office hours on website**
  - **Hongwei Wang: office hours on website (online students should contact this TA; on-campus can contact too)**
  - **Anshuman Tripathi: office hours TBD**

(Check course website)

Virtually:

- **Discussion Forum: Piazza (most preferable, monitored daily) – 24 hour turnaround time for questions!**
- **Email (turnaround time may be longer than Piazza) – use cs425-ece428@mx.uillinois.edu**
- **Email individuals (instructor, TA) only if absolutely necessary (e.g., private matter)**

Readings

- **For today's lecture: Relevant parts of Chapter 1**
- **Typically, before a lecture you will be expected to read the relevant chapter**
- **For next lecture**
  - **No readings! Topic: "Introduction to Cloud Computing"**
  - **Fill out and return Student InfoSheet**