Announcements

- Email me the survey: See the Announcements page on the course web for instructions
- Today
  - Conceptual overview of distributed systems
  - Characterization of distributed systems
- Reading
  - Today: Chapter 1 of Coulouris
  - Next time: Chapter 2 of Coulouris
- Take a break around 10:15am
- Ack: Some slides are from Coulouris or Steve Ko

Networks of computers are everywhere!

- The Internet
- Mobile phone networks
- Corporate networks
- Factory networks
- Campus networks
- Home networks
- In-car networks
- . . .

Main motivation for distributed systems

- Sharing of resources
  - Hardware components, e.g., disks, printers, etc.
  - Software entities, e.g., files, databases, search engines, etc.

- Resources managed by servers and accessed by clients

A distribution system is . . .

- one in which components located at networked computers communicate and coordinate their actions only by passing messages
- (another definition) a collection of entities with a common goal, each of which is autonomous, programmable, asynchronous and failure-prone, and which communicates through an unreliable communication medium

- These definitions lead to the following characteristics of distributed systems:
  - Concurrency of components
  - No global clock
  - Independent failures of components

Concurrency

- In a network of computers, concurrent program execution is the norm, sharing resources
- The capacity of the system to handle shared resources can be increased by adding more resources to the network

No global clock

- When programs need to cooperate, they coordinate their actions by exchanging messages
- Close coordination depends on shared time
- There are limits to the accuracy with which the computers in a network can synchronize their clocks – there is no single global notion of the correct time [Section 6.1.1 of Tanenbaum]
- This is a consequence of the fact that the only communication is by sending messages through the network

Independent failures

- All computer systems can fail and the system designer is responsible for planning for the consequences of possible failures
- Each component of a distributed system can fail independently, leaving the others still running
- The failure may be due to a crash or a slow response

Examples of distributed systems

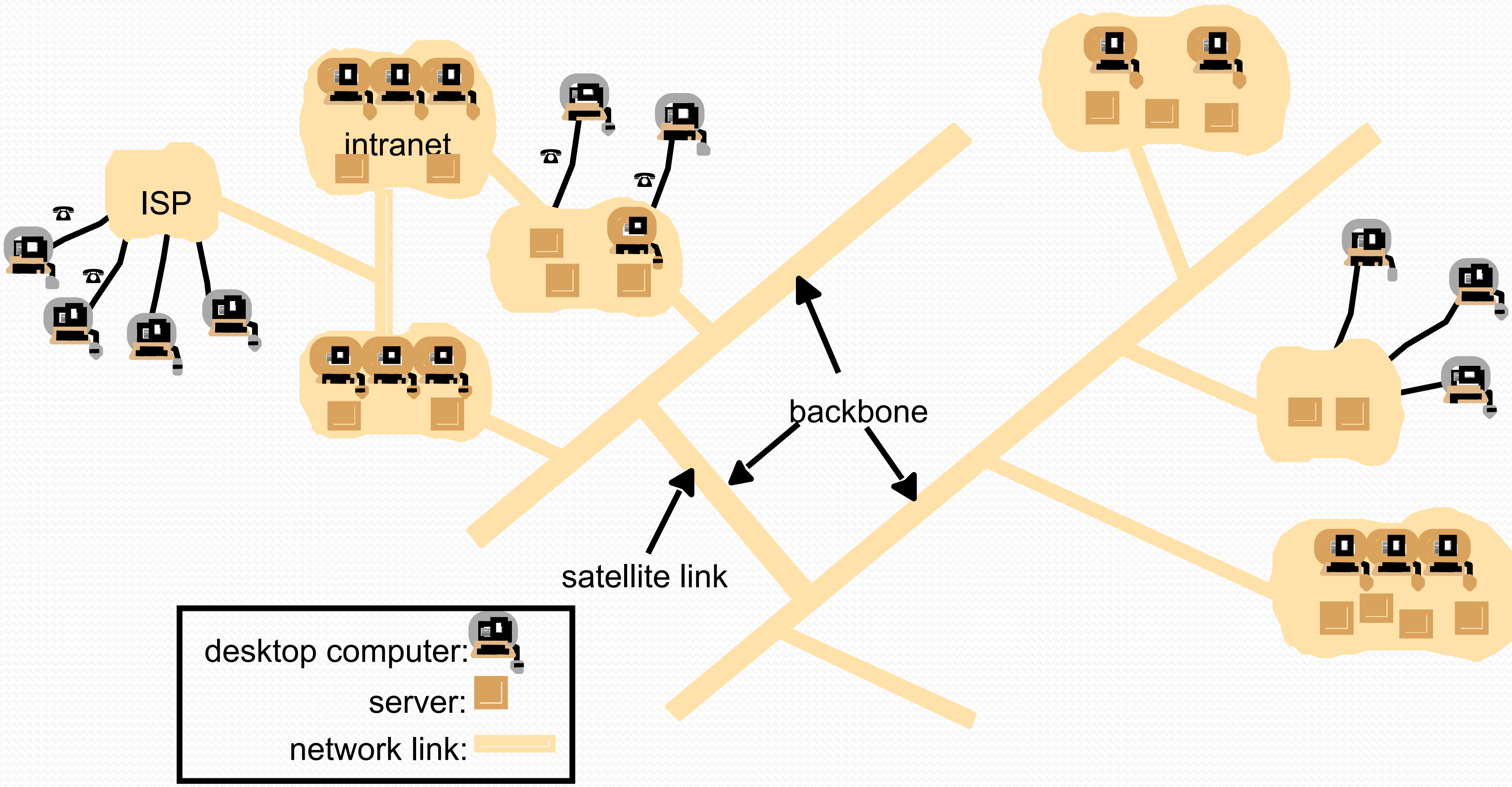
- The Internet and the associated WWW
- See next slide
  - Understanding of underlying technology in these examples is central to a knowledge of modern computing

Selected app domains with networked apps

Finance and commerce	eCommerce (Amazon) online banking and trading
The information society	Web information and search engines, ebooks, Wikipedia; social networking (Facebook)
Creative industries and entertainment	online gaming, music and film in the home, user-generated content (YouTube)
Healthcare	health informatics, on online patient records, monitoring patients
Education	e-learning, virtual learning environments, distance learning
Transport and logistics	GPS in route finding systems, map services (Google Maps, Google Earth)
Science	The Grid as an enabling technology for collaboration between scientists
Environmental management	sensor technology to monitor earthquakes, floods, or tsunamis

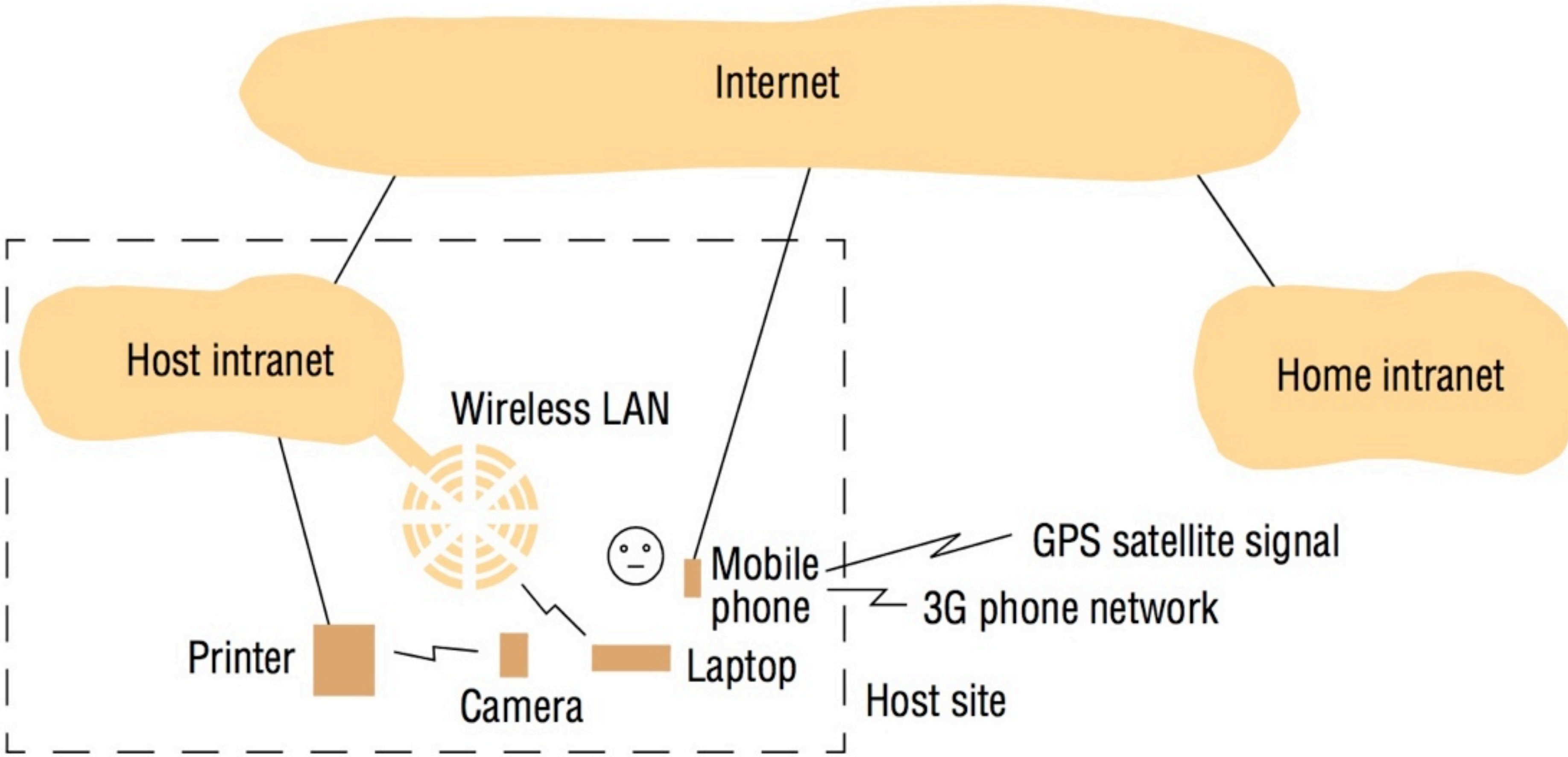
Trends in distributed systems

- Pervasive networking and the modern Internet



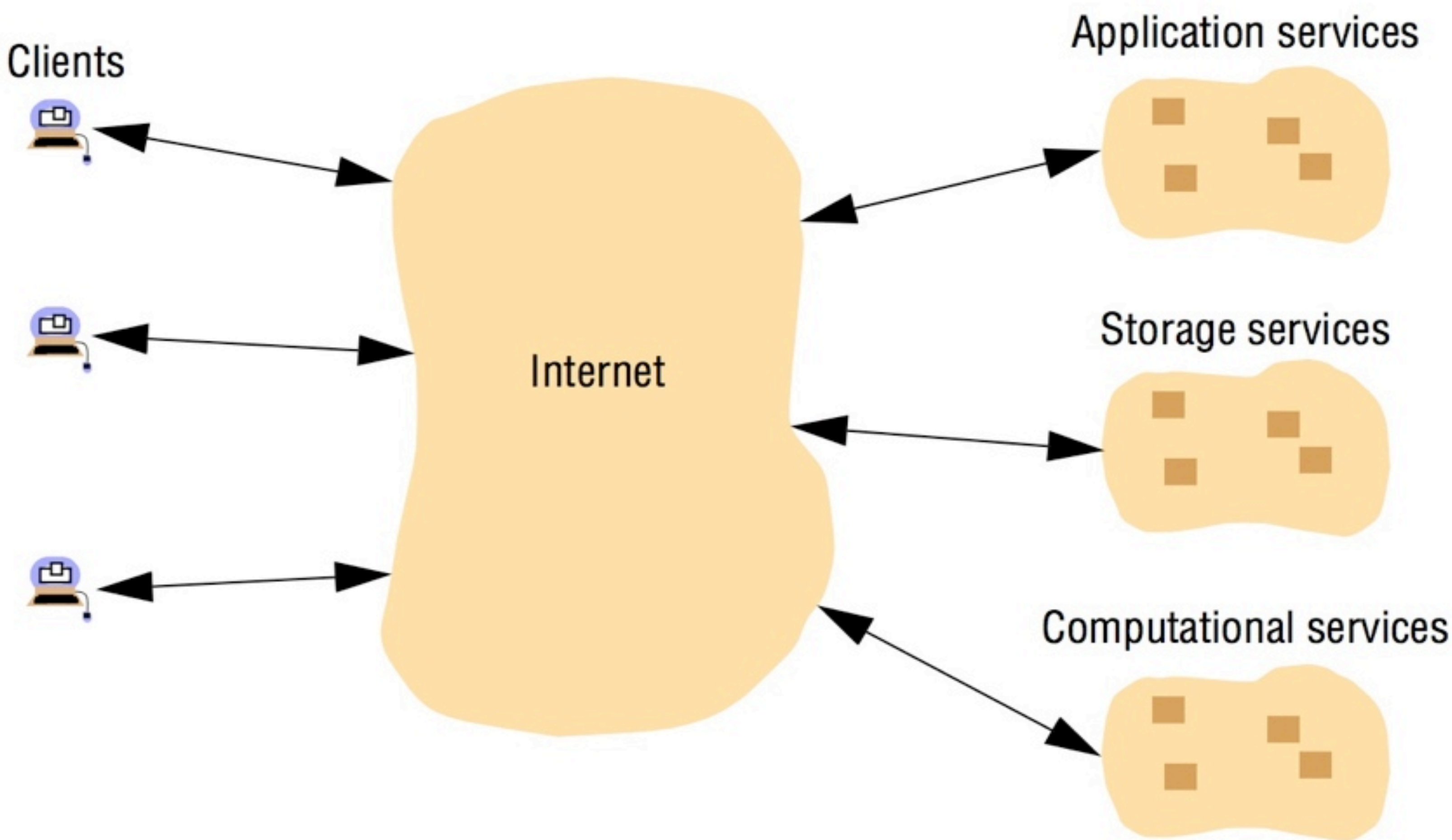
Trends in distributed systems (cont.)

- Mobile and ubiquitous computing (Internet of Things)



Trends in distributed systems (cont.)

- Distributed computing as a utility



Resource sharing and the Web

- Patterns of sharing vary widely in scope and how closely users work together
  - A search engine on the Web is used by people all over the world
  - In CSCW (computer-supported cooperative working) a group of users share resources in a small, closed group

Service, server, and client

- Requests are sent in messages from clients to a server and replies (services) are sent in messages from the server to the clients, e.g., a web browser requests a web page from a web server

- A complete interaction between a client and a server, from the point when the client sends its request to when it receives the server's response, is called a *remote invocation*. (cf. function call in an address space)

- The same process may be both a client and a server since servers sometimes invoke operations on other servers

Building a distributed system

- "The number of people who know how to build really solid distributed systems … is about ten."

  - Scott Shenker, Professor at UC Berkeley

- The point: it's hard to build a solid distributed system

Why is it hard to build one?

- Scale: hundreds or thousands of machines
  - Google: 4K-machine MapReduce cluster
  - Yahoo!: 4K-machine Hadoop cluster
  - Akamai: 70K machines distributed over the world
  - Facebook: 60K machines providing the service
  - Hard enough to program one machine!
- Dynamism: machines do fail!
  - 50 machine failures out of 20K machine cluster per day (reported by Yahoo!)
  - 1 disk failure out of 16K disks every 6 hours (reported by Google)
- What else?
  - Concurrent execution, consistency, etc.

OK, but who cares?

- This is where all the actions are!
  - What are the two biggest driving forces in the computing industry for the last five years?
  - It's the cloud!
  - And smartphones!
  - And they are distributed!
- Now --- it's all about distributed systems!
  - Well…with a bit of exaggeration… ;-)

Challenges in building distributed systems

- Heterogeneity of components
- Openness
- Security
- Scalability
- Failure handling
- Concurrency
- Transparency

Heterogeneity

- Networks
- Computer hardware
- Operating systems
- Programming languages
- Implementations by different developers

- Internet protocols
- Middleware – CORBA, Java RMI, Apache Thrift
- Mobile code – code sent from one computer to another and run at the destination, e.g., applets; virtual machine approach; JavaScript

Openness

- Is the system extensible in various ways?
- The degree to which new resource-sharing services can be added and be made available for use by various programs
- Achieved by publishing the key interfaces and by conforming to a uniform communication mechanism

Security

- Security for info resources has 3 components:
  - Confidentiality (protection against disclosure to unauthorized individuals)
  - Integrity (protection against alteration or corruption)
  - Availability (protection against interference with the means to access the resources)
- Encryption techniques are used to achieve security
- Two security challenges
  - Denial of service attacks (bombarding the service with a large number of pointless requests)
  - Security of mobile code (possible effects of running it is unpredictable)

Scalability

- Distributed systems operate effectively and efficiently at many different scales

- A system is scalable if it will remain effective when there is a significant increase in the number of resources and users

Growth of the Internet

Date	*Computers*	*Web servers*	*Percentage*
1993, July	1,776,000	130	0.008
1995, July	6,642,000	23,500	0.4
1997, July	19,540,000	1,203,096	6
1999, July	56,218,000	6,598,697	12
2001, July	125,888,197	31,299,592	25
2003, July	~200,000,000	42,298,371	21
2005, July	353,284,187	67,571,581	19

Challenges with scalability

- Controlling the cost of physical resources
  - For a system with *n* users to be scalable the quantity of resources required to support them should be at most *O(n)*
- Controlling the performance loss
  - Managing a set of data whose size is proportional to the number of users or resources, e.g., DNS table, with a hierarchical structure thus *O(log n)* performance for lookup
  - For a system to be scalable the maximum performance loss should be no worse than this
- Preventing software resources running out
  - Running out IP addresses (32 bits in 1970s vs. 128 bits being adopted requiring modifications to many software components)
- Avoiding performance bottlenecks
  - Name table in DNS was centralized in the past
  - Now partitioned between servers located throughout the Internet and administered locally

Failure handling (next)

- Failures in a distributed system are partial making failure handling particularly difficult
- Detecting failures – some are difficult to detect, e.g., a remote crashed server in the Internet
- Masking failures – making detected failures less severe, e.g.,
  - Retransmit message when they fail to arrive
  - Redundant files on a pair of disks
- When one of the components fails, only the work that was using the failed component is affected

Concurrency

- Several clients may attempt to access a shared resource at the same time, e.g., bid data in an auction
- Services and applications generally allow multiple client requests to be processed concurrently
- Any object that represents a shared resource in a distributed system must be responsible for ensuring correct semantics in a concurrent environment
- For an object to be safe in a concurrent environment its operations must be synchronized in such a way that its data remains consistent

Transparency

- Concealing from the user and the application programmer of the separation of components in a distributed system so that the system is perceived as a whole rather than as a collection of independent components

- The ANSA Reference Manual and the International Organization for Standardization's Reference Manual for Open Distributed Processing identify eight forms of transparency

Transparencies

- *Access transparency*: enables local and remote resources to be accessed using identical operations
- *Location transparency*: enables resources to be accessed without knowledge of their physical or network location (for example, which building or IP address)
- *Concurrency transparency*: enables several processes to operate concurrently using shared resources without interference between them
- *Replication transparency*: enables multiple instances of resources to be used to increase reliability and performance without knowledge of the replicas by users or application programmers
- *Failure transparency*: enables the concealment of faults, allowing users and application programs to complete their tasks despite the failure of hardware or software components
- *Mobility transparency*: allows the movement of resources and clients within a system without affecting the operation of users or programs
- *Performance transparency*: allows the system to be reconfigured to improve performance as loads vary
- *Scaling transparency*: allows the system and applications to expand in scale without change to the system structure or the application algorithms

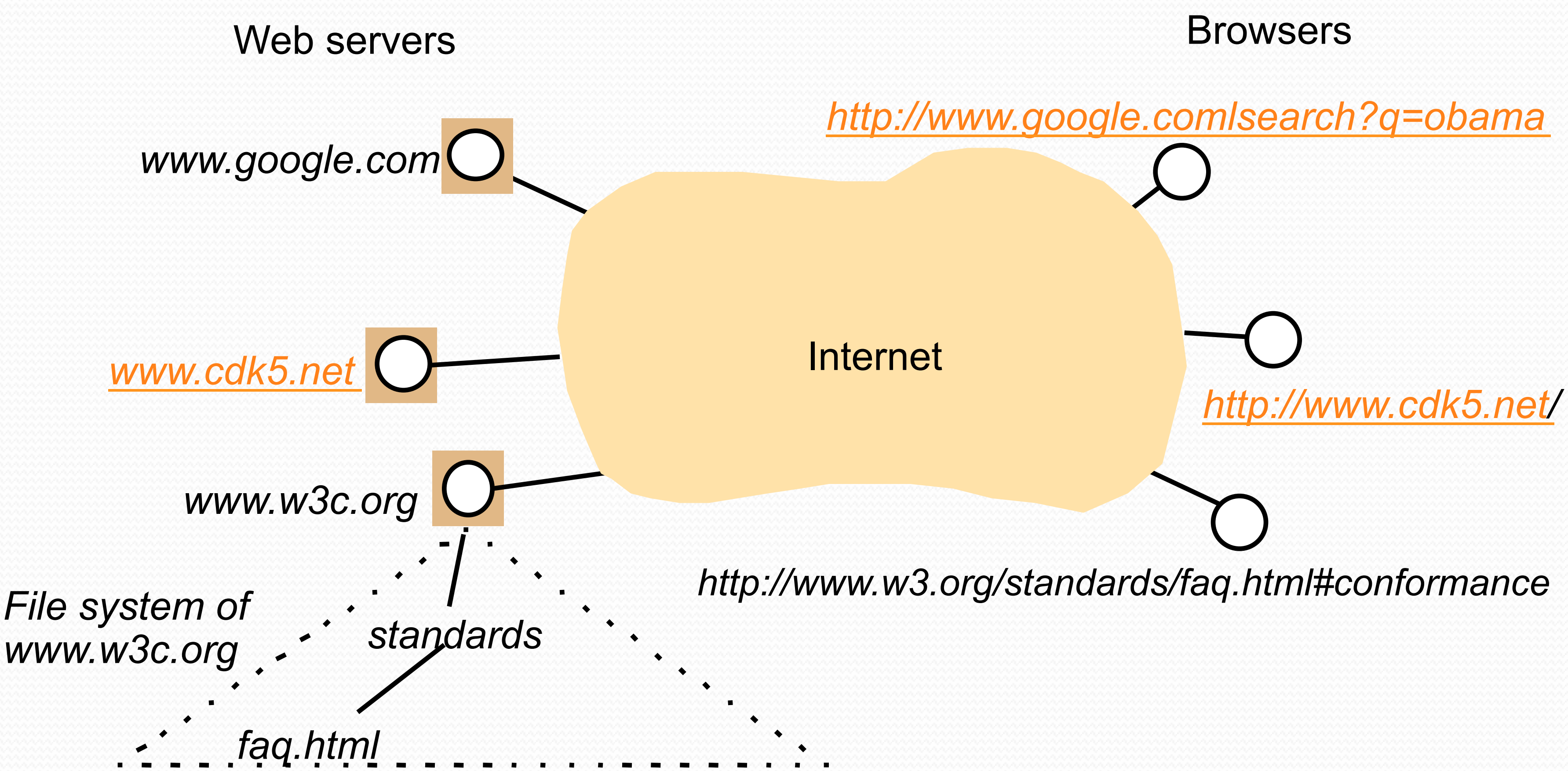
Case study: The WWW

- The Web is an open system – can be extended in new ways without disturbing its existing functionality
  - Its operation is based on communication standards and document standards
  - Open with respect to the types of resource that can be published and shared on it

The WWW (cont.)

- The Web is based on 3 main standard technological components:
  - HTML
  - URL’s
  - A client-server architecture with HTTP

Web servers and web browsers



HTML (HyperText Markup Langauge)

- Used to specify the text and images that make up the contents of a web page
  - headings, paragraphs, tables, images, and links
- An HTML text is stored in a file that a web server can access
- A browser retrieves the file from a web server and renders the content of the file and displays
- HTML5
  - New tags: <video>, <audio>, <canvas>, <section>, <article>, <header>, <nav>, etc.
  - Eliminate plugins

URLs (Uniform Resource Locators)

- Used to identify a resource
- The term used in web architecture documents is URI (Uniform Resource Identifier)
- The format of a URL:
  - http://servername [:port][/pathname][?query][#fragment]
  - For example,
    - http://www.cmc.edu
    - http://www.cmc.edu/cs135/index.html#intro
    - http://www.google.com/search?q=lee

HTTP (HyperText Transfer Protocol)

- Defines the ways in which browsers and other types of clients interact with web servers

- Main features for retrieving resources
  - Request-reply interactions
  - Content types – MIME types, e.g., text/html, image/GIF
  - One resource per request
  - Simple access control (password)

Dynamic pages

- Interacting with services that generate data rather than retrieving static data
  - Filling out a web form
  - Server has to ‘process’ the user’s input so the client’s request is a CGI program
  - Result of running the program is returned as HTML text

- Downloaded code (cf. CGI code)
  - JavaScript code
  - Applets

Web services

- Programmatic access to web resources is commonplace, i.e., programs other than browsers can be clients of the Web too
- HTML and HTTP standards are lacking for programmatic interoperation
- XML (Extensible Markup Language) to represent standard, structured, application-specific forms
  - Meta-language for describing data – portable between applications

Discussion of the Web

- Hugely successful
- Problems
  - Dangling links due to deleted resources
  - Users getting ‘lost in hyperspace’
  - Search engines are imperfect at producing what the user specifically intends
  - The Web faces problems of scale
    - Use of caching in browsers
    - Division of the server’s load across clusters of computers

Next time

- Distributed system models