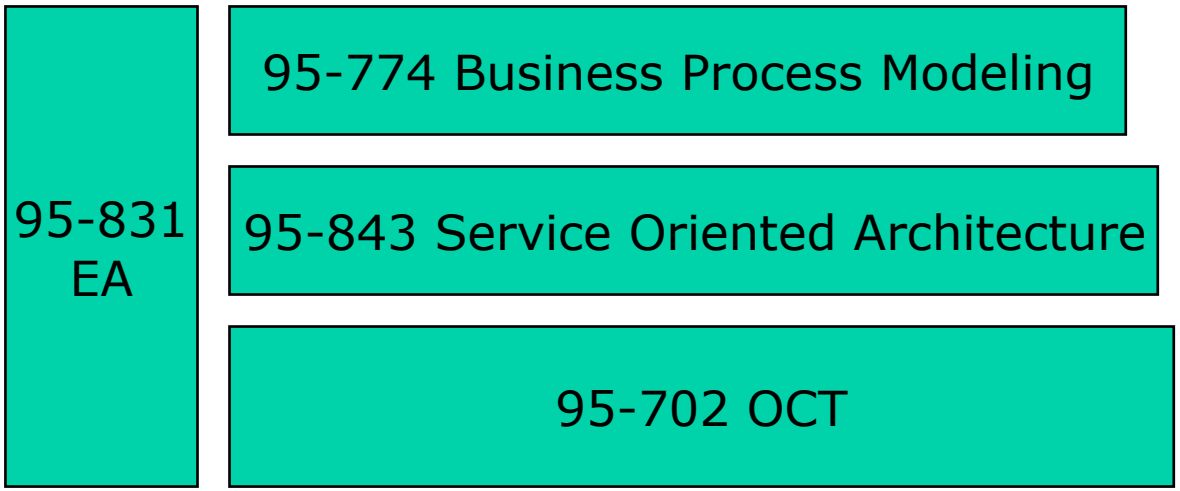
95-702 Distributed Systems

Lecture 1: Introduction

Course Web Site

- http://www.andrew.cmu.edu/~mm6

How Related to Other Courses



Course Technologies

- IDE (Netbeans)
- Java Web Applications (Glassfish)
- Message Oriented Middleware (Sun's Message Queue)
- Web Services (JDK 6, Glassfish)
- Distributed Objects (Java RMI, and EJB's)
- Mobile platform (Android)

Getting Started Notes

- See the schedule for instructions on getting started with the course technologies.
- The installation includes Netbeans, Glassfish and the Android emulator.
- Not to be turned in but please begin this assignment now and let us know of any problems.
- Homework 1 is also assigned.

Structure of the Course

- Lectures / class participation
- Demonstrations (with your active involvement)
- Homework (pencil and paper and programming) The secret is to start early.
- Midterm
- Final examination

Readings

- Readings from the required text are assigned for each lecture -- read them in advance.
- Readings from the web will also be assigned.
- For this week, read Coulouris chapters 1 and 2

Grading

- Homework/Programming (5-7) 50%
- Midterm 20%
- Final Exam 30%

- We will be very fussy about deadlines. One second late is late.
- All times are Adelaide times. Blackboard may show Pittsburgh times. We will work from Adelaide time.
- Use the discussion board for all queries with a response that should be heard by the entire class.

Characterization of Distributed Systems

- Components are located on networked computers and execute concurrently.
- Components communicate and coordinate only by passing messages.
- There is no global clock.
- What was the “Pony Express” like?

Main Motivations for Constructing DS

- Communications and Resource sharing.
- We want to share:
  - Programs
  - Data
  - CPU cycles
  - Files
  - Printers
  - Etc..
- What do we share when we use "Cloud Computing"?

Challenges in Constructing DS

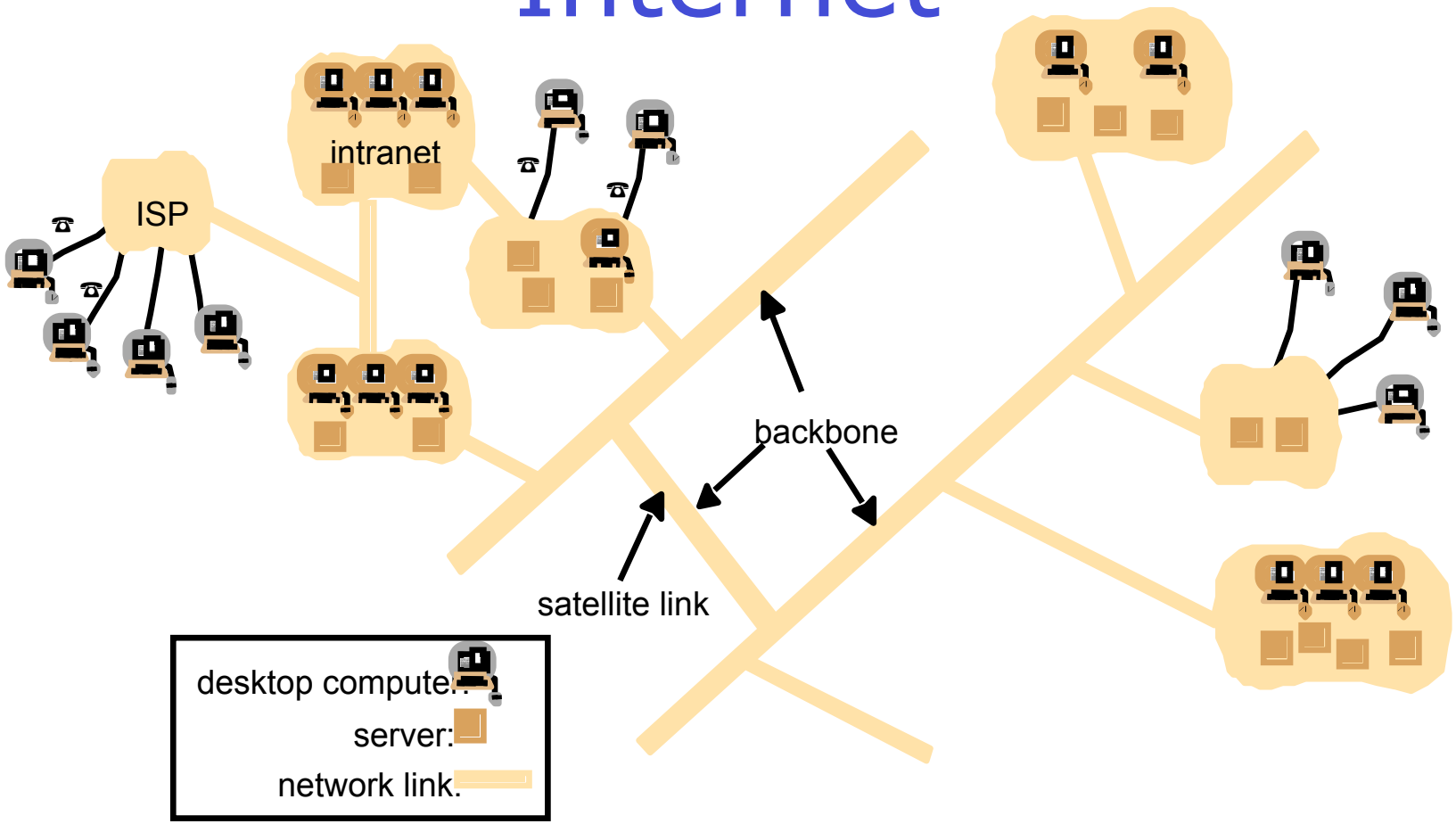
- Heterogeneity of components
- Openness
- Security (Eve and Mallory)
- Scalability
- Failure handling
- Concurrency of components
- Transparency

Which of these are not challenges when constructing standalone systems?

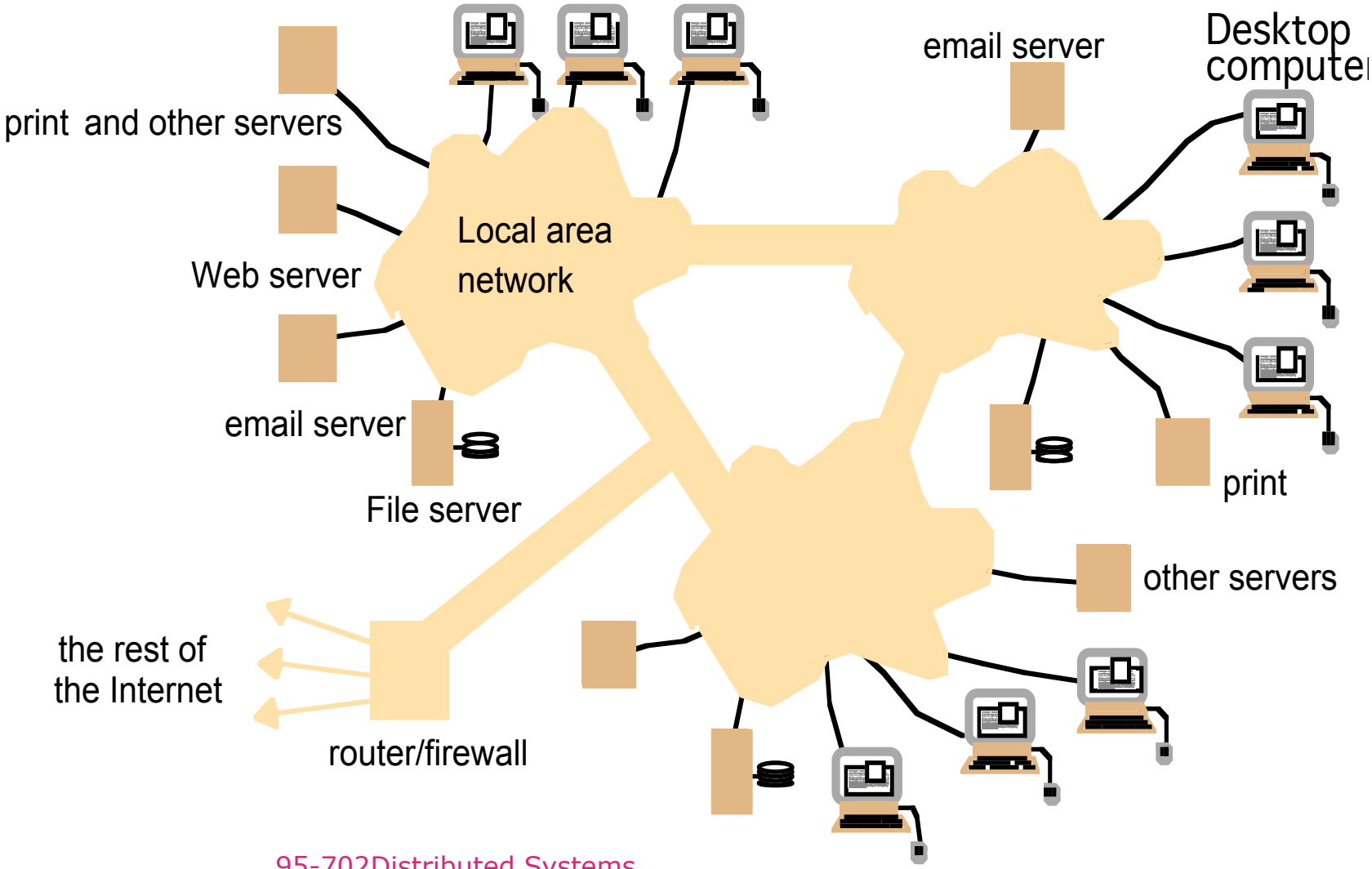
Example Distributed Systems

- The internet
  A collection of diverse networks
  A very large distributed system
  providing services such as email,
  file transfer, telnet, and recently,
  WWW, Web Services, and multimedia
- Intranets (a portion of the internet separately administrated) and connected to the internet via a router
- Mobile and ubiquitous computing
- Sensor networks

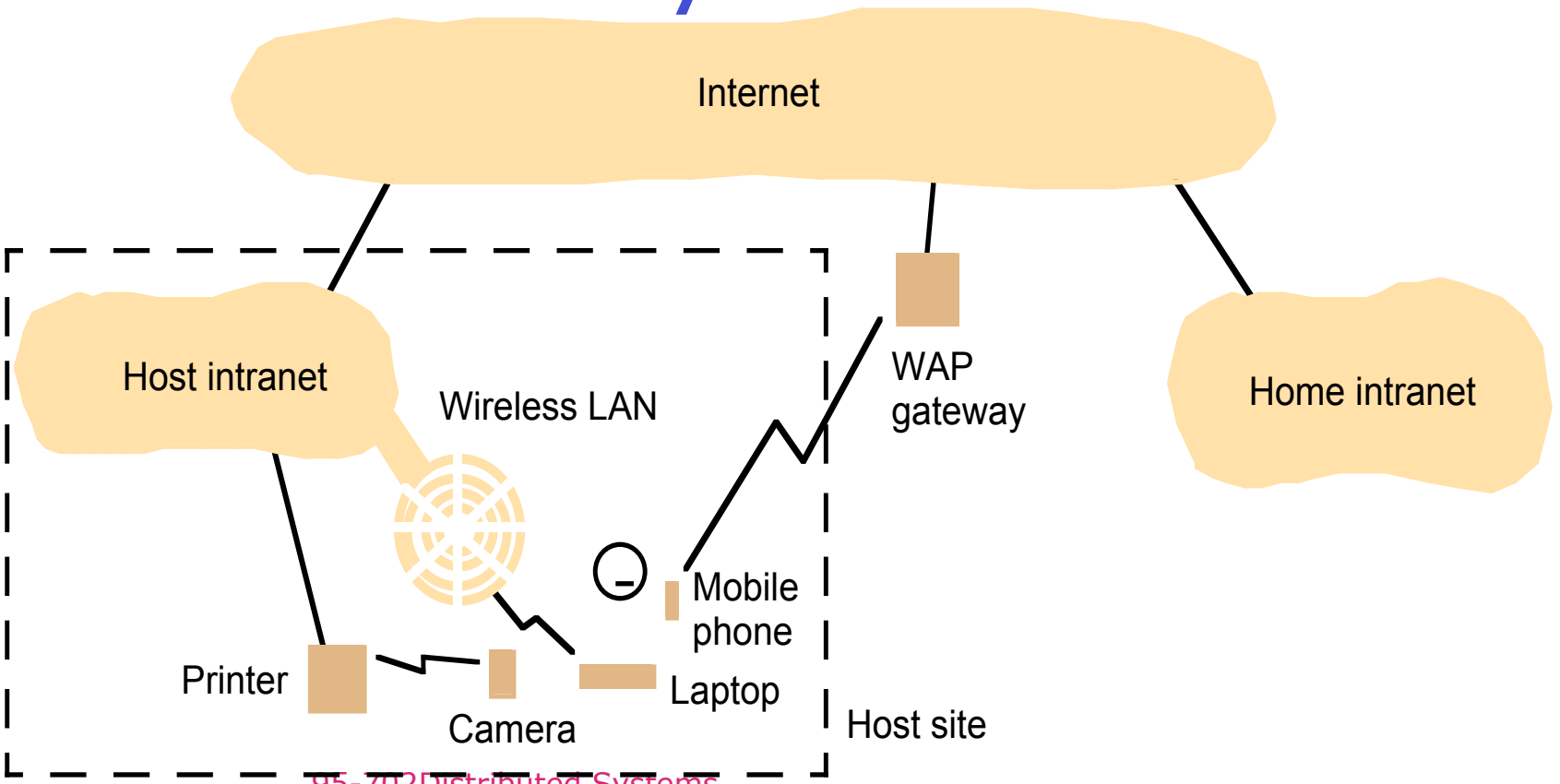
A typical portion of the Internet



A typical intranet



Portable and handheld devices in a distributed system



Resource Sharing and the Web

- A *server* is a running program on a networked computer that accepts requests from programs running on other computers to perform a service and respond appropriately
- The requesting processes are referred to as *clients*
- WWW, Web Services, networked printers and email fit this model

The World Wide Web(1)

- Created by Sir Tim Berners-Lee at European centre for nuclear research (CERN) in Switzerland in 1989 (Knighted 2003)
- Provides a hypertext structure allowing documents to contain links to other documents
- Is an open system (can be extended and implemented in new ways, standards are public and widely implemented)

The World Wide Web (2)

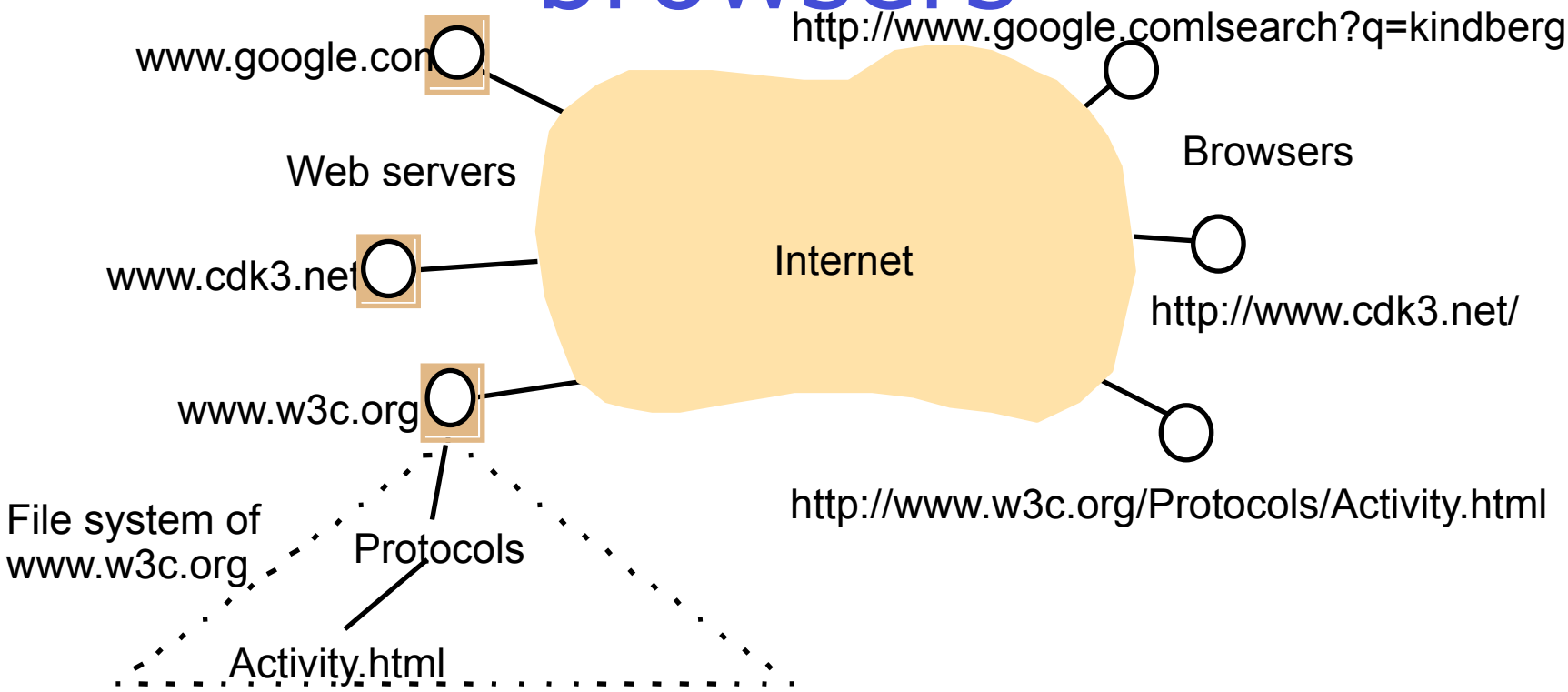
- The web is based on three main standard technological components

  (1) HTML for presentation of content and Links

  (2) URL's to point to a resource and specify a protocol

  (3) HTTP to describe the request and reply protocol

Web servers and web browsers



A Request May Cause

- A simple file transfer
- A process to be run on the server and content sent to the browser (CGI programs, servlets, JSP pages, etc.)
- Program code to be downloaded and executed in the browser (JavaScript, Applets, Java Web Start, etc.)

Challenges to DS Design(1)

- Heterogeneity applies to
  - Networks (Ethernet, Wireless,..)
  - Computer Hardware (PC's, PDA's,..)
  - Operating Systems (Linux, OS X, Windows,..)
  - Programming Languages (Java, C++, C#,..)
  - Different developers
- *Middleware* provides a programming abstraction that addresses these issues

Challenges to DS Design(2)

- Open
  - -- The system can be extended and re-implemented in a variety of ways
  - -- The key specifications are published
  - -- The system is independent of a particular vendor

Challenges to DS Design(3)

- Security

  Some resources are highly valued.

  Confidentiality is often required.

  Integrity is often required.

  Availability is often essential.
- Cryptography will help with much of this.
- Denial of Service and Mobile Code are not yet easy to handle

Challenges to DS Design(4)

- Scalability

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

  For a system with n users to be scalable the quantity of physical resources required to support them should be O(n).

Challenges to DS Design(5)

- Scalability

  For system to be scalable, the loss in performance attributed to additional users or resources should be O(Log n)

  Examples of bottleneck avoidance
  - distributed algorithms
  - Domain Name System
  - caching
  - replication

Computers in the Internet

Date	*Computers*	*Web servers*
1979, Dec.	188	0
1989, July	130,000	0
1999, July	56,218,000	5,560,866

The internet has been scalable and extensible.
However, the 32 bit IP address was too small.
Moving to IPv6, 128 bits.

Computers vs. Web Servers on 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

Netcraft reports 118,000,000 web sites active in May of 2007.

Challenges to DS Design(6)

- Failure Handling

  Particularly difficult in DS

  Failures are often partial

  Issues include:

  Detecting failures

  Masking or hiding failures

  with,e.g.,retries

  Tolerating Failures

  Recovery from failures or rolling back changes

Challenges to DS Design(7)

- Concurrency

Multiple client requests are often allowed to take place concurrently. Forcing one request at a time would limit throughput.
Standard techniques exist to protect against conflicts, E.g., Java and C# synchronization.

Challenges to DS Design(8)

- Types of Transparency (or concealment)

Access transparency: enables local and remote resources to be accessed using identical operations.

Location transparency: enables resources to be accessed without knowledge of their location.

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.

Challenges to DS Design(9)

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.

Pitfalls when Developing Distributed Systems

Some false assumptions that may be made by designers:

- The network is reliable.
- The network is secure.
- The network environment is homogeneous.
- Latency is zero.
- Bandwidth is infinite.
- Transport cost is zero.
- There is one administrator.