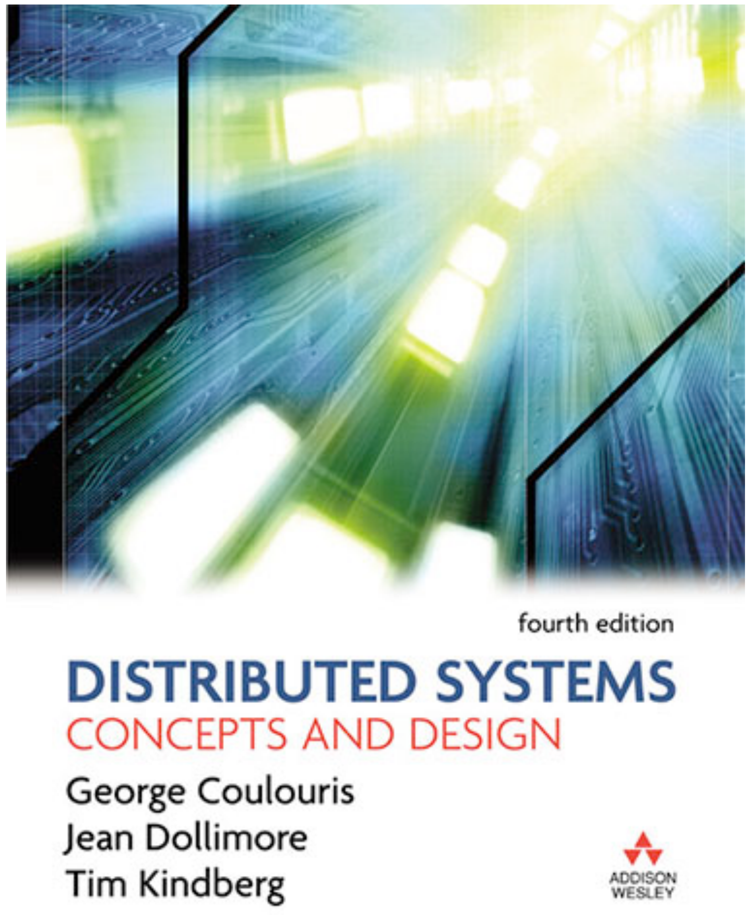
Characterization of Distributed Systems

Nicola Dragoni
Embedded Systems Engineering
DTU Informatics



Introduction

- Networks of computers are everywhere:
  - mobile phone networks
  - corporate networks
  - campus networks
  - home networks
  - Internet
  - ...

Introduction

- Networks of computers are everywhere:
  - mobile phone networks
  - corporate networks
  - campus networks
  - home networks
  - Internet
  - ...

DISTRIBUTED SYSTEMS

Distributed System

- A possible definition: *a distributed system is a system in which hardware or software components located at networked computers communicate and coordinate their actions only by passing messages*.

- Networked computers (i.e., computers that are connected by a network) may be spatially separated by any distance:
  - separate continents
  - same building
  - same room
  - ...

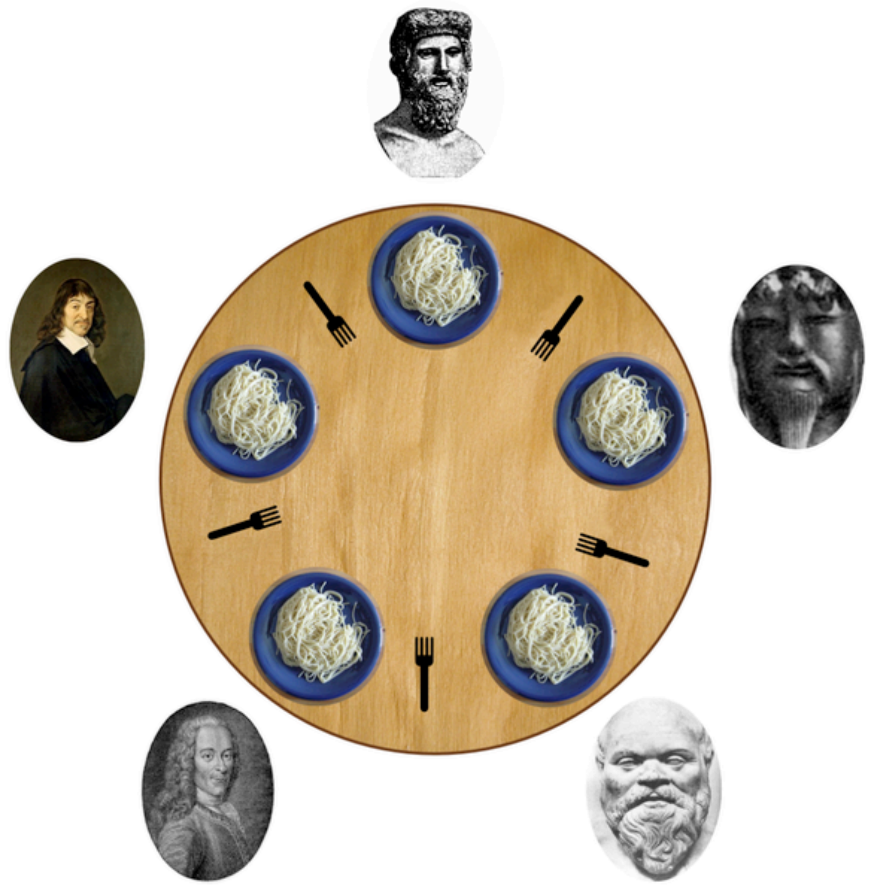
Fundamental Characteristic: Concurrency

- In a network of computers, concurrent program execution is the norm.
- I can do my work on my computer *while*

  you do your work on your computers,

  sharing resources (such as web pages or files)

  when necessary.


Dining Philosophers Problem

Fundamental Characteristic: No Global Clock

- There is no single global notion of the correct time.
- Direct consequence of the fact that *when programs need to cooperate they coordinate their actions by exchanging messages*.
- The *only* communication is by sending messages through a network.

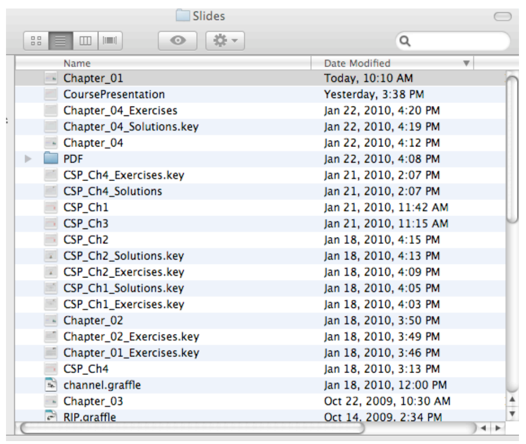
Fundamental Characteristic: Independent Failures

- All computer systems can fail. Distributed systems can fail in new ways:
- Faults in the network result in the isolation of the computers that are connected to it.
    - But this does not mean that they stop running! The programs running on them may not be able to detect whether the networks has failed or has become unusually slow.
- Failure of a computer or a crash (i.e., unexpected termination of a program somewhere in the system) is not immediately made known to the other components with which it communicates.
- Each component of the system can fail independently, leaving the others still running.

Why Distributed Systems?

- The motivation for constructing and using distributed systems stems from a desire to share resources.

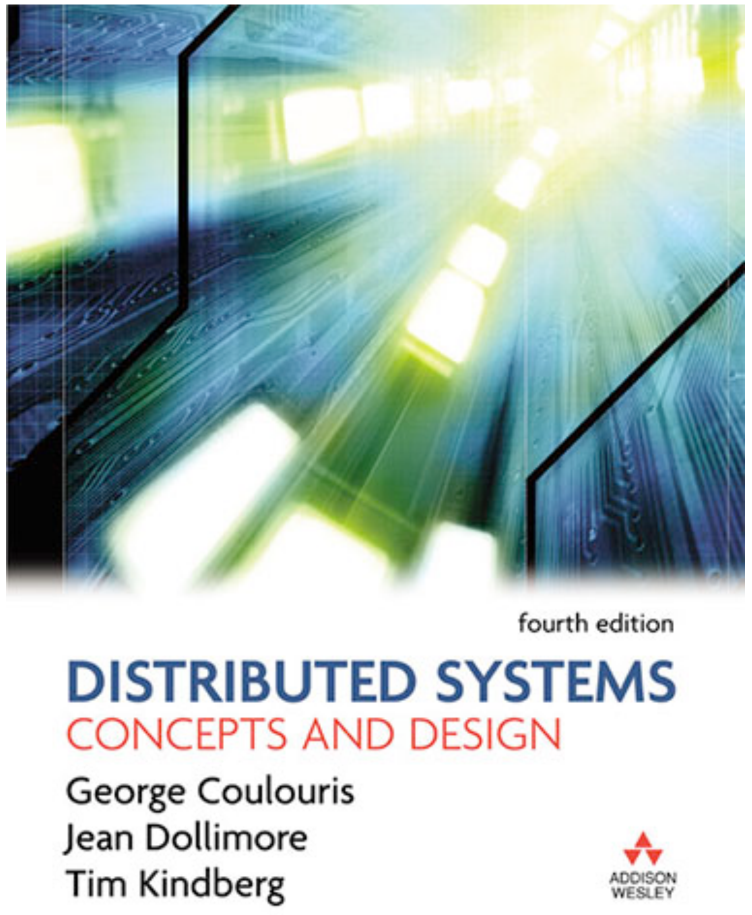
- Resource = abstract term that characterizes the range of things that can be usefully be shared in a networked computer system:
  - Hardware components: disks, printers, ...
  - Software entities: files, databases, and data objects of all kinds.



Slides

Name	Date Modified
Chapter_01	Today, 10:10 AM
CoursePresentation	Yesterday, 3:38 PM
Chapter_04_Exercises	Jan 22, 2010, 4:20 PM
Chapter_04_Solutions.key	Jan 22, 2010, 4:19 PM
Chapter_04	Jan 22, 2010, 4:12 PM
PDF	Jan 22, 2010, 4:08 PM
CSP_Ch4_Exercises.key	Jan 21, 2010, 2:07 PM
CSP_Ch4_Solutions	Jan 21, 2010, 2:07 PM
CSP_Ch1	Jan 21, 2010, 11:42 AM
CSP_Ch3	Jan 21, 2010, 11:15 AM
CSP_Ch2	Jan 18, 2010, 4:15 PM
CSP_Ch2_Solutions.key	Jan 18, 2010, 4:13 PM
CSP_Ch2_Exercises.key	Jan 18, 2010, 4:09 PM
CSP_Ch1_Solutions.key	Jan 18, 2010, 4:05 PM
CSP_Ch1_Exercises.key	Jan 18, 2010, 4:03 PM
Chapter_02	Jan 18, 2010, 3:50 PM
Chapter_02_Exercises.key	Jan 18, 2010, 3:49 PM
Chapter_01_Exercises.key	Jan 18, 2010, 3:46 PM
CSP_Ch4	Jan 18, 2010, 3:13 PM
channel.graffle	Jan 18, 2010, 12:00 PM
Chapter_03	Oct 22, 2009, 10:30 AM
RIP.graffle	Oct 14, 2009, 2:34 PM

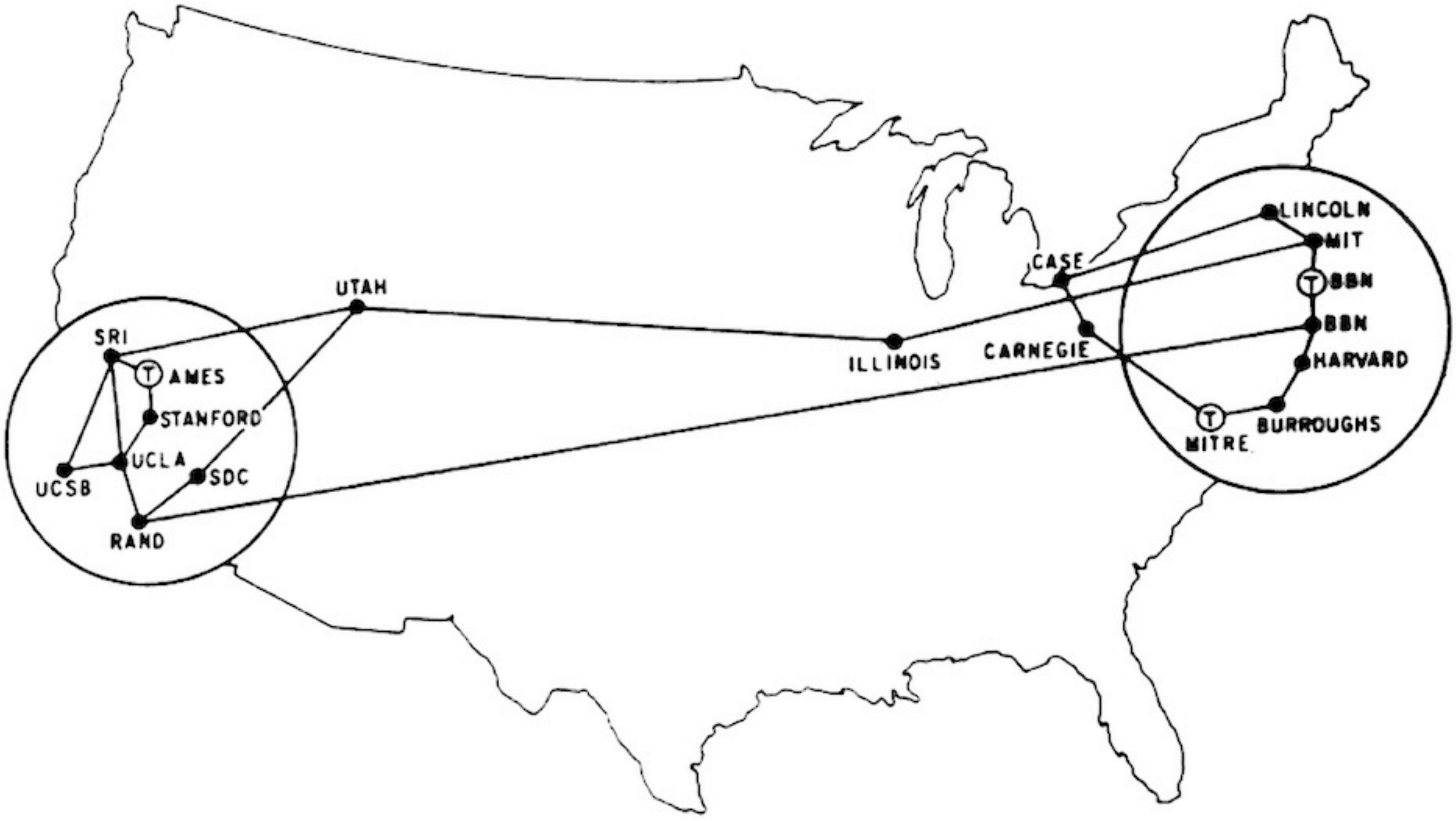
Characterization of Distributed Systems



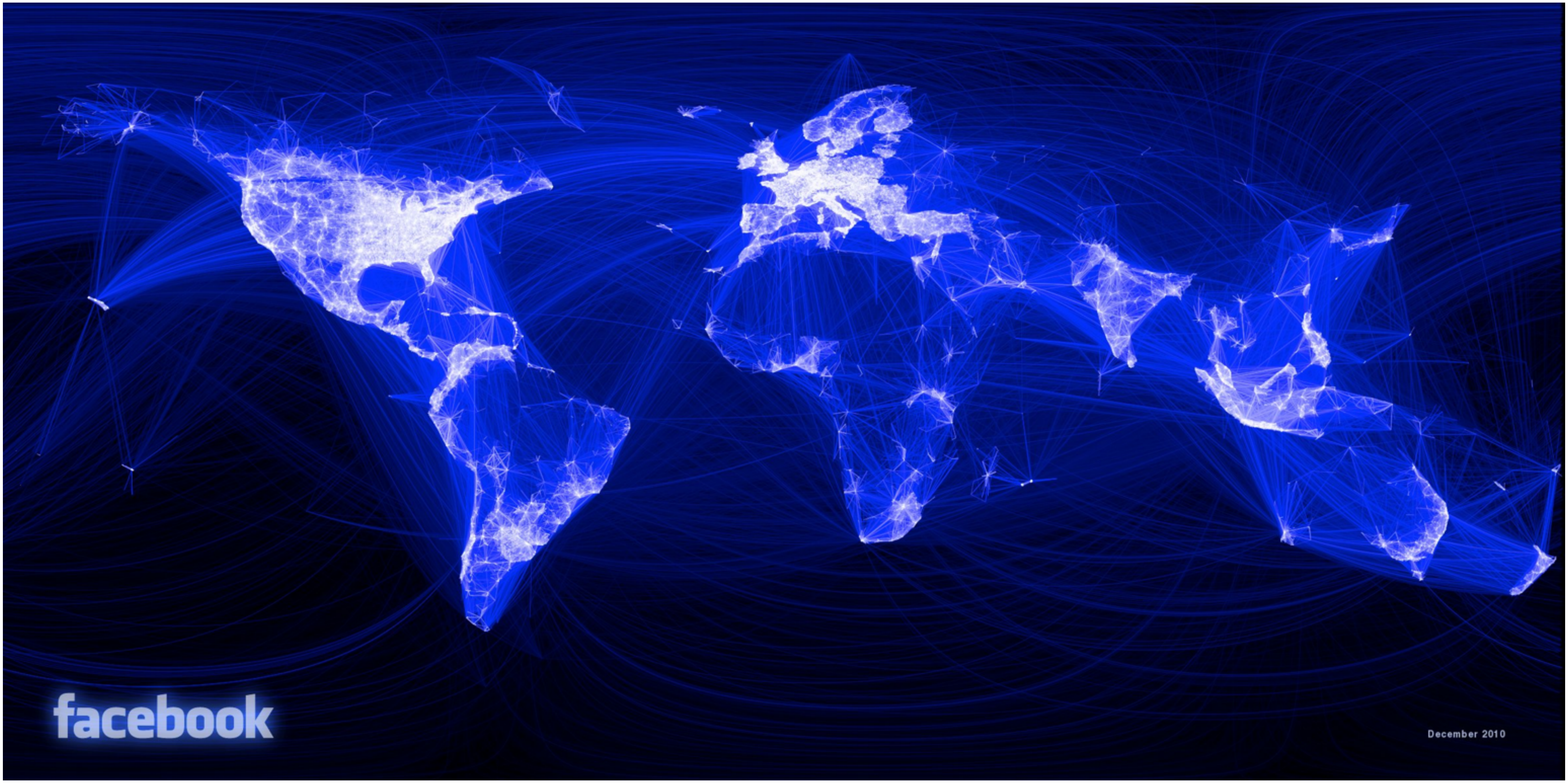
Example 1: The Internet

- A vast interconnected collection of computer networks of many different types.
  - Programs running on the computers connected to it interact by passing messages, employing a common means of communication (Internet protocols).
- A very large distributed system.
  - It enables users, *wherever they are*, to make use of open-ended services (WWW, email, file transfer, multimedia services, ...)

A Map of the First Internet (ARPANET)



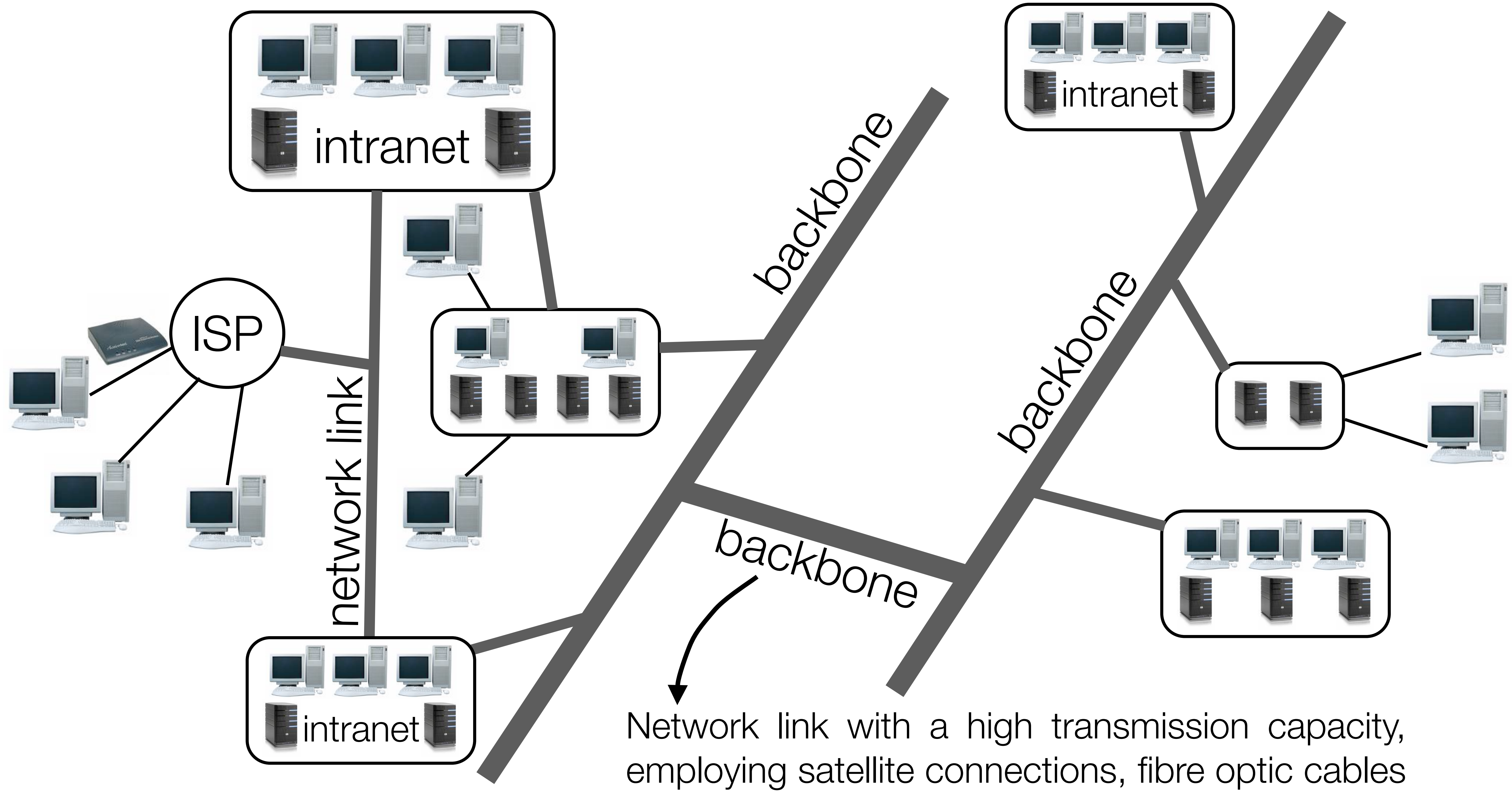
Facebook (December 2010)



Web (November 2003)

http://www.opte.org/maps/

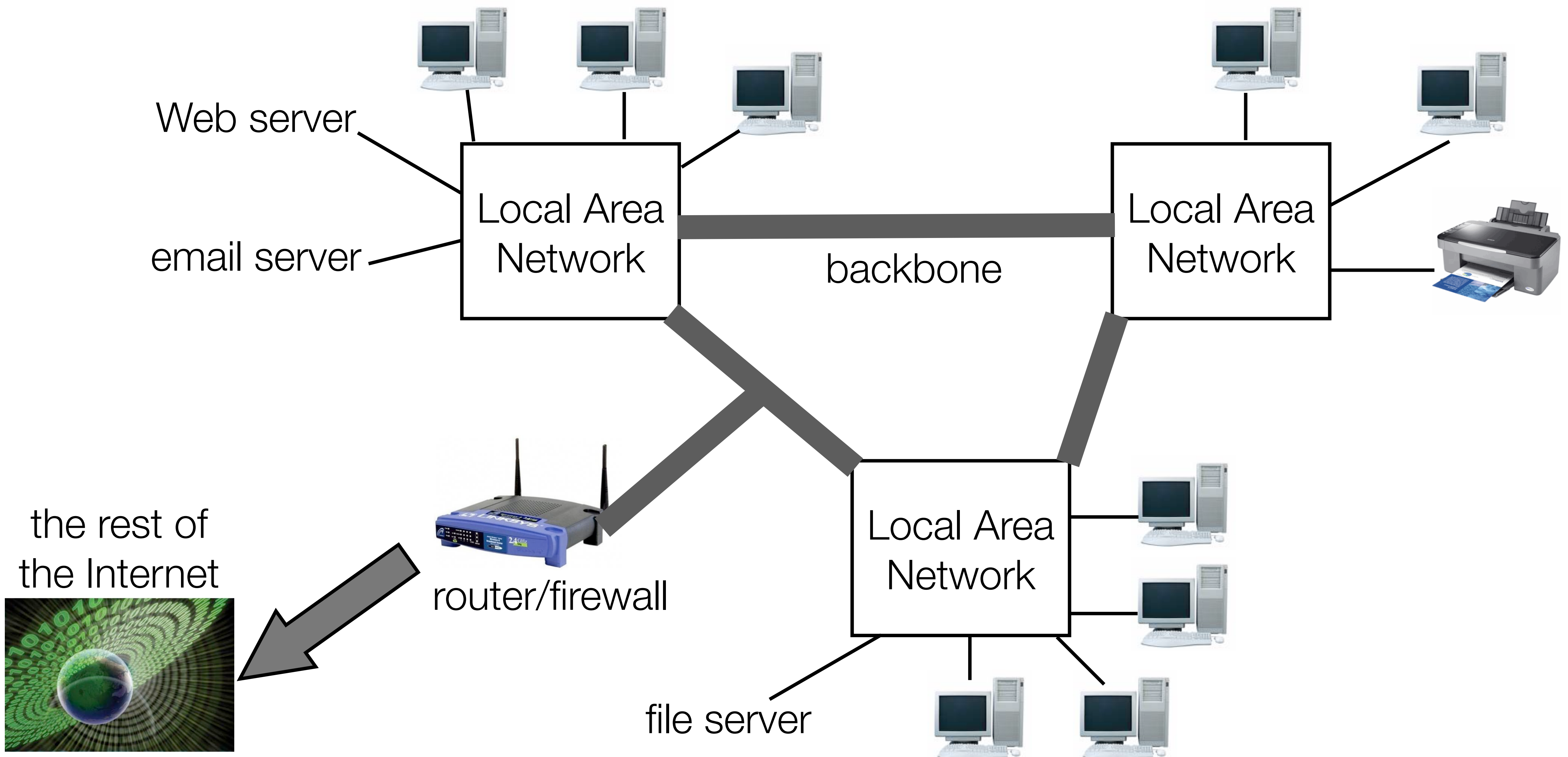
Example 1: A Typical Portion of the Internet



Network link with a high transmission capacity, employing satellite connections, fibre optic cables and other high-bandwidth circuits.

Example 2: Intranets

- An intranet is a portion of the Internet that is separately administered and has a boundary that can be configured to enforce local security policies.



Example 2: Router and Firewall

- An intranet is connected to the Internet via a router, which allows:
  - the users *inside the intranet* to make use of services elsewhere (Web, email)
  - users in *other intranets* to access the services it provides.

Example 2: Router and Firewall

- An intranet is connected to the Internet via a router, which allows:
  - the users *inside the intranet* to make use of services elsewhere (Web, email)
  - users in *other intranets* to access the services it provides.

- The role of a firewall is to protect an intranet by preventing unauthorized messages leaving or entering.
  - Implemented by filtering incoming and outgoing messages, for example according to their source or destination.

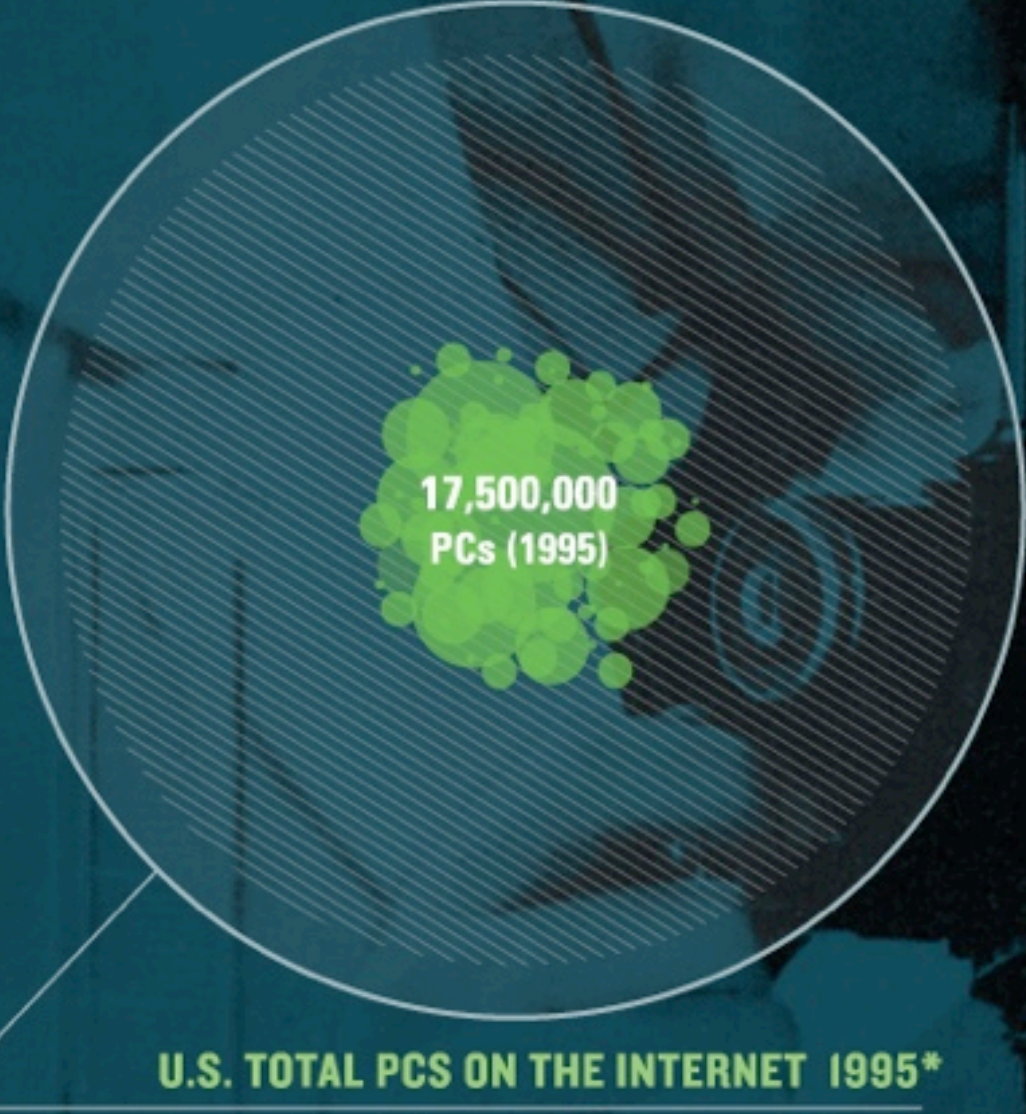
Example 3: Mobile and Ubiquitous Computing

- Mobile computing (also called nomadic computing) is the performance of computing tasks while the user is on the move, or visiting places other than their usual environment.

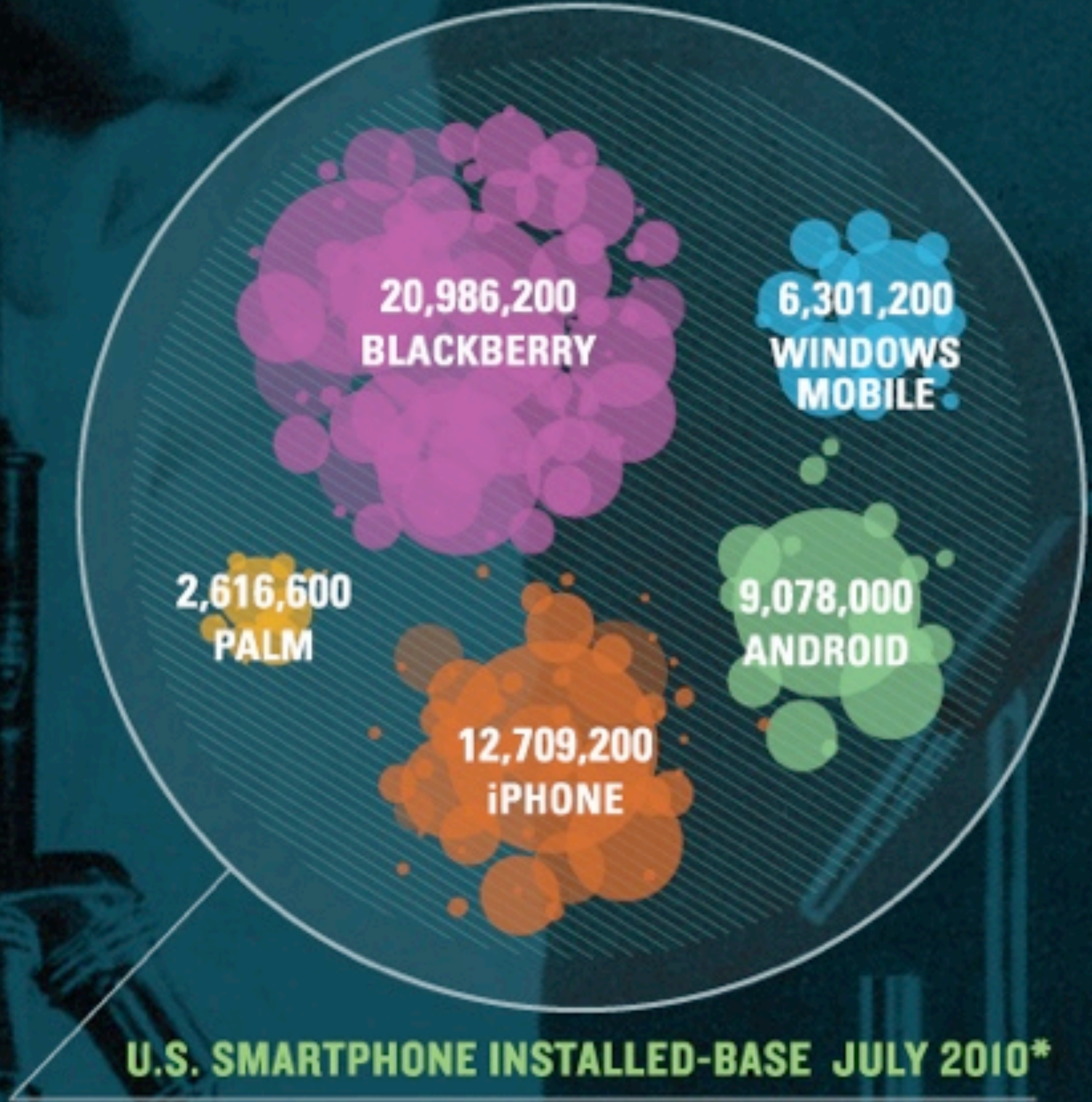
- Users who are away from their “home” intranet (the intranet at work, or their residence) are still provided with access to resources via the devices they carry with them.

SOURCE: MOCANA.COM

EPIDEMIC-READY? VULNERABLE SMARTPHONE POPULATIONS CROSS AN IMPORTANT THRESHOLD



U.S. TOTAL PCS ON THE INTERNET 1995*



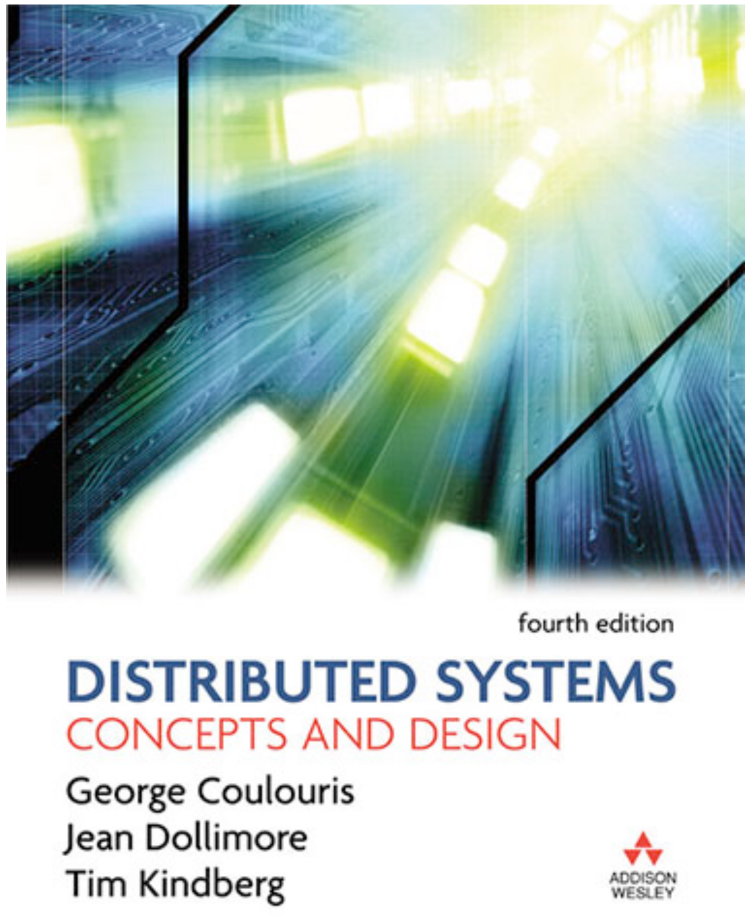
U.S. SMARTPHONE INSTALLED-BASE JULY 2010*

*comScore, Harris Interactive

Until recently, smartphone security hasn't been of much concern to IT professionals or consumers. But that's about to change, if we consider recent history.

Experts have cited the low relative "infectable populations" of smartphones as a reason for why hackers wouldn't bother to go after them. Just recently, however, populations of several smartphone operating systems are approaching or have already surpassed the number of PCs on the Internet in 1995. That's when viruses and hackers started to become a big problem for PCs.

Characterization of Distributed Systems



Resource Sharing as Motivation for Distr. Systems

- We routinely share hardware and software resources:
  - printers, fax, disks, ...
  - database, files, web pages, search engines, ...
- Resources in a distributed system are physically encapsulated within computers and can only be accessed by communication.
- For effective sharing, each resource must be managed by a program that offers a communication interface enabling the resource to be accessed and updated reliably and consistently.

Service

- A service is a distinct part of a computer system that manages a collection of related resources and presents their functionality to users and applications.
- Examples:
  - we access shared files through a file service
  - we send documents to printers through a printing service
  - we buy goods through an electronic payment service
- The only access we have to a service is via its set of operations.
  - A file service provides read, write, and delete operations on files.

Clients and Servers

- A server is a running program (a *process*) on a networked computer that accepts requests from programs (usually running on other computers) to perform a service and responds appropriately.
- The requesting processes are referred to as clients.
- Requests are sent in messages from clients to a server.
  - When a client sends a request for an operation to be carried out, we say that the client invokes an operation upon the server.
- Replies are sent in messages from the server to the clients.
- Remote invocation: 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).

Clients vs Servers

- The same process can be both a client and a server, since servers sometimes invoke operations on other servers.
- The terms “client” and “server” apply only to the roles played in a single request.
- But in general they are distinct concepts:
  - clients are active and server are passive (reactive)
  - server run continuously, whereas clients last only as long as the applications of which they form a part.

The Web: Client-Server Resource Sharing System

- The World Wide Web is an evolving and open system for publishing and accessing resources and services across the Internet.
- For instance, through Web browsers (clients) users can
  - retrieve and view documents of many types
  - listen to audio streams
  - view video streams
  - and in general interact with an unlimited set of services.

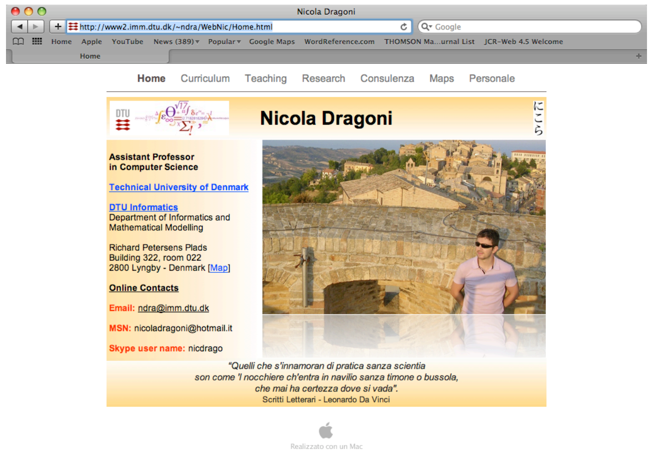
[Web] Main Technological Components

1. The HyperText Markup Language (HTML) is a language for specifying the contents and layout of pages as they are displayed by Web browsers.

2. Uniform Resource Locators (URLs) which identify documents and other resources stored as part of the Web.

3. A client-server system architecture, with standard rules for interaction (the HyperText Transfer Protocol - HTTP) by which browsers and other clients fetch documents and other resources from Web servers.

Web Browsers and Web Servers Example



Nicola Dragoni

http://www2.imm.dtu.dk/~ndra/WebNic/Home.html

Home Curriculum Teaching Research Consulenza Maps Personale

Nicola Dragoni

Assistant Professor in Computer Science

Technical University of Denmark

DTU Informatics
Department of Informatics and Mathematical Modelling

Richard Petersens Plads
Building 322, room 022
2800 Lyngby - Denmark [Map]

Online Contacts

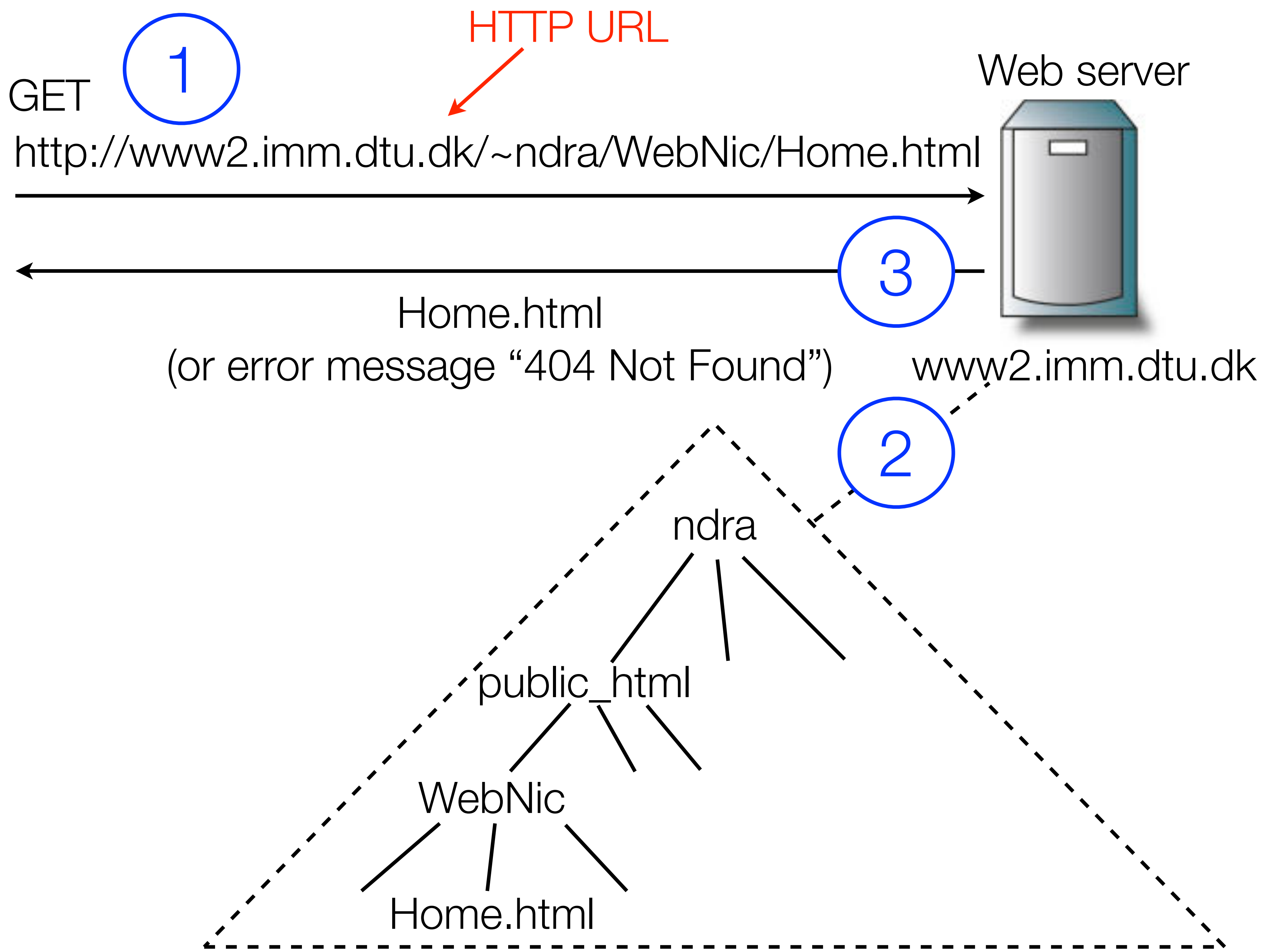
Email: ndra@imm.dtu.dk

MSN: nicoladragoni@hotmail.it

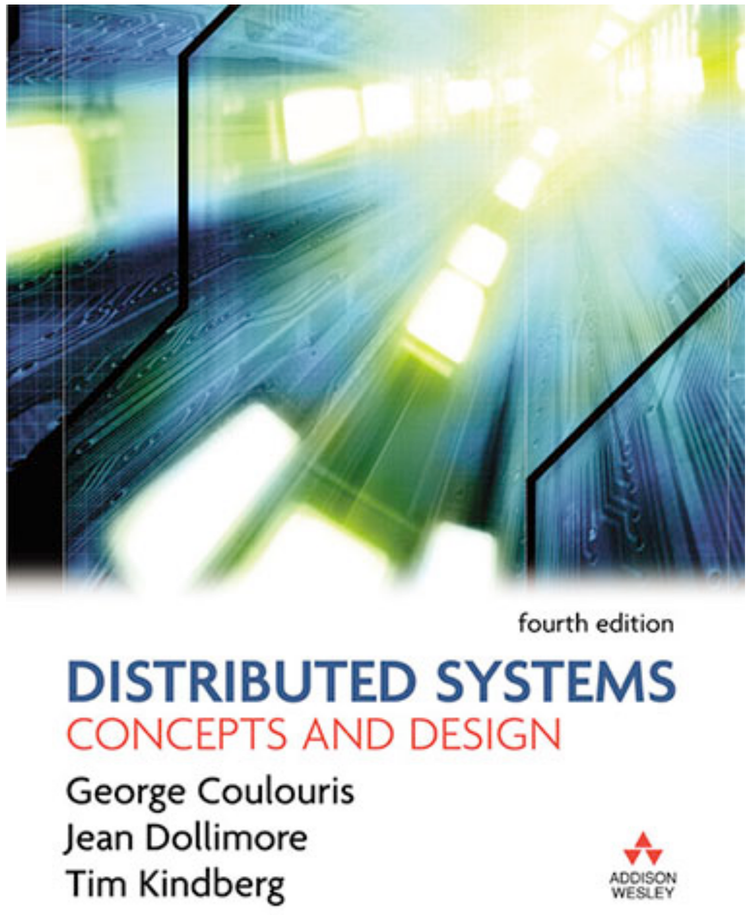
Skype user name: nicdrago

"Quelli che s'innamoran di pratica sanza scientia son come 'l nocchiere ch'entra in navilio sanza timone o bussola, che mai ha certezza dove si vada".
Scritti Letterari - Leonardo Da Vinci

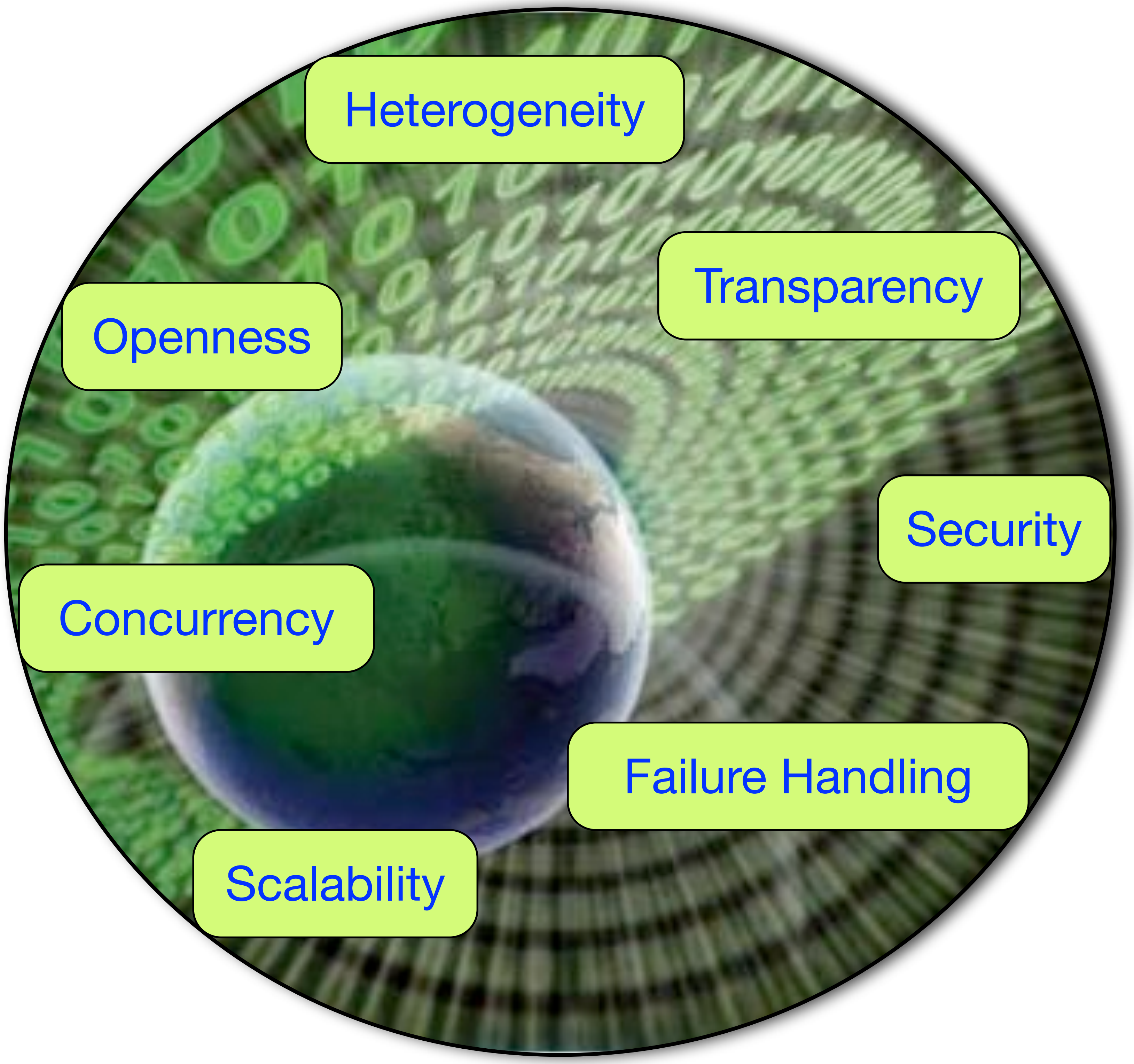
Realizzato con un Mac


HTTP URL
GET
1
http://www2.imm.dtu.dk/~ndra/WebNic/Home.html
Web server
3
Home.html
(or error message “404 Not Found”)
www2.imm.dtu.dk
2
ndra
public_html
WebNic
Home.html

Characterization of Distributed Systems



Design Challenges for Distributed Systems



Heterogeneity of Components

- Heterogeneity (i.e., variety and difference) applies to the following:
  - networks
  - computer hardware
  - operating systems
  - programming languages
  - implementations by different developers

Heterogeneity of Components

- Heterogeneity (i.e., variety and difference) applies to the following:
  - networks
  - computer hardware
  - operating systems
  - programming languages
  - implementations by different developers

Heterogeneity can be addressed by means of:
- protocols (such as Internet protocols)
- middleware (software layer that provides a programming abstraction)

Openness

- The openness of a computer system is the characteristic that determines whether the system can be *extended* and *re-implemented* in various ways.

- In distributed systems it is determined primarily by the degree to which new resource sharing services can be added and be made available for use by a variety of client programs.

Openness

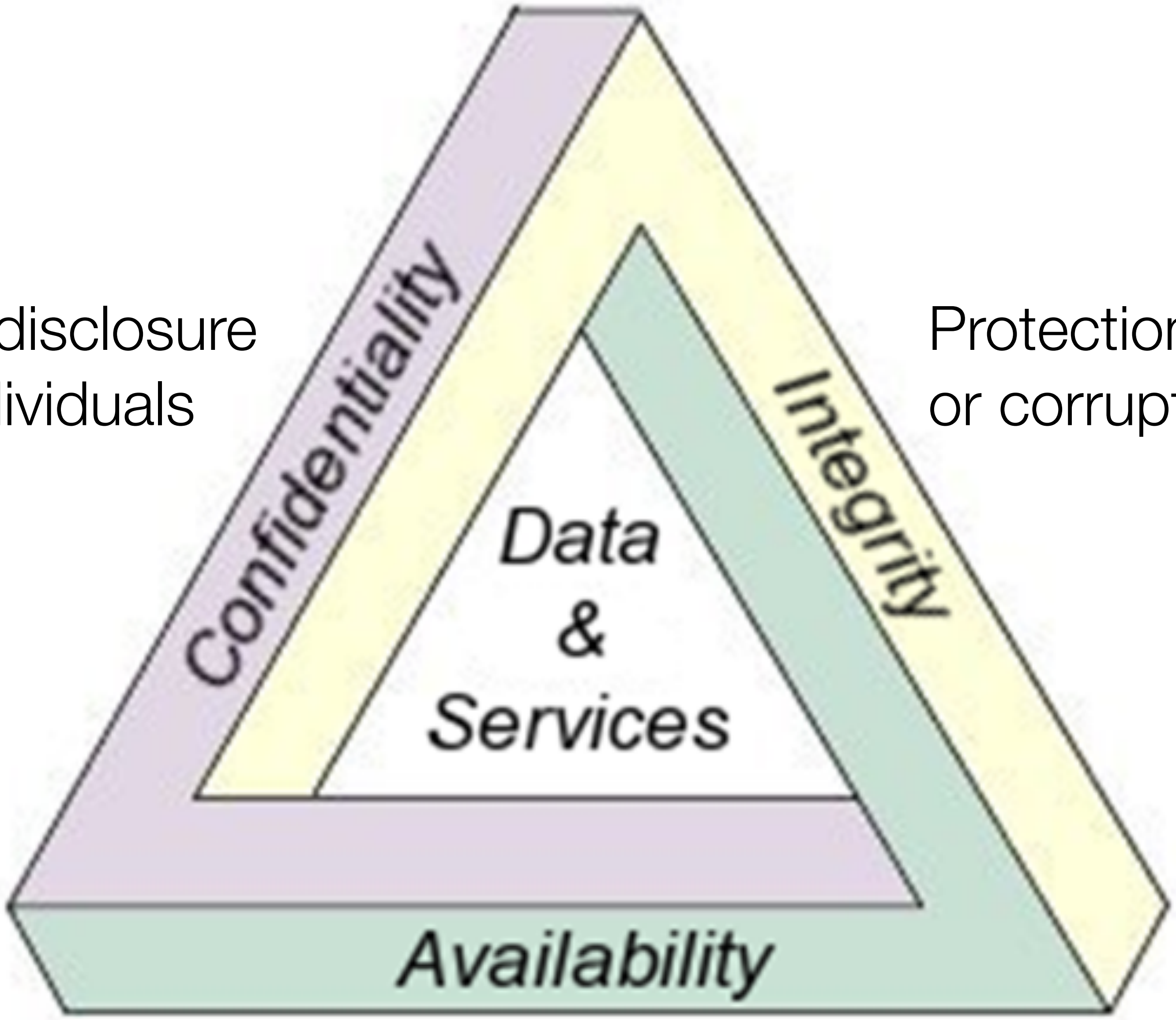
- The openness of a computer system is the characteristic that determines whether the system can be *extended* and *re-implemented* in various ways.

- In distributed systems it is determined primarily by the degree to which new resource sharing services can be added and be made available for use by a variety of client programs.

- Open distributed systems may be extended
    - at the hardware level by the addition of computers to the network
    - at the software level by the introduction of new services and the re-implementation of old ones.

Security

Protection against disclosure to unauthorized individuals

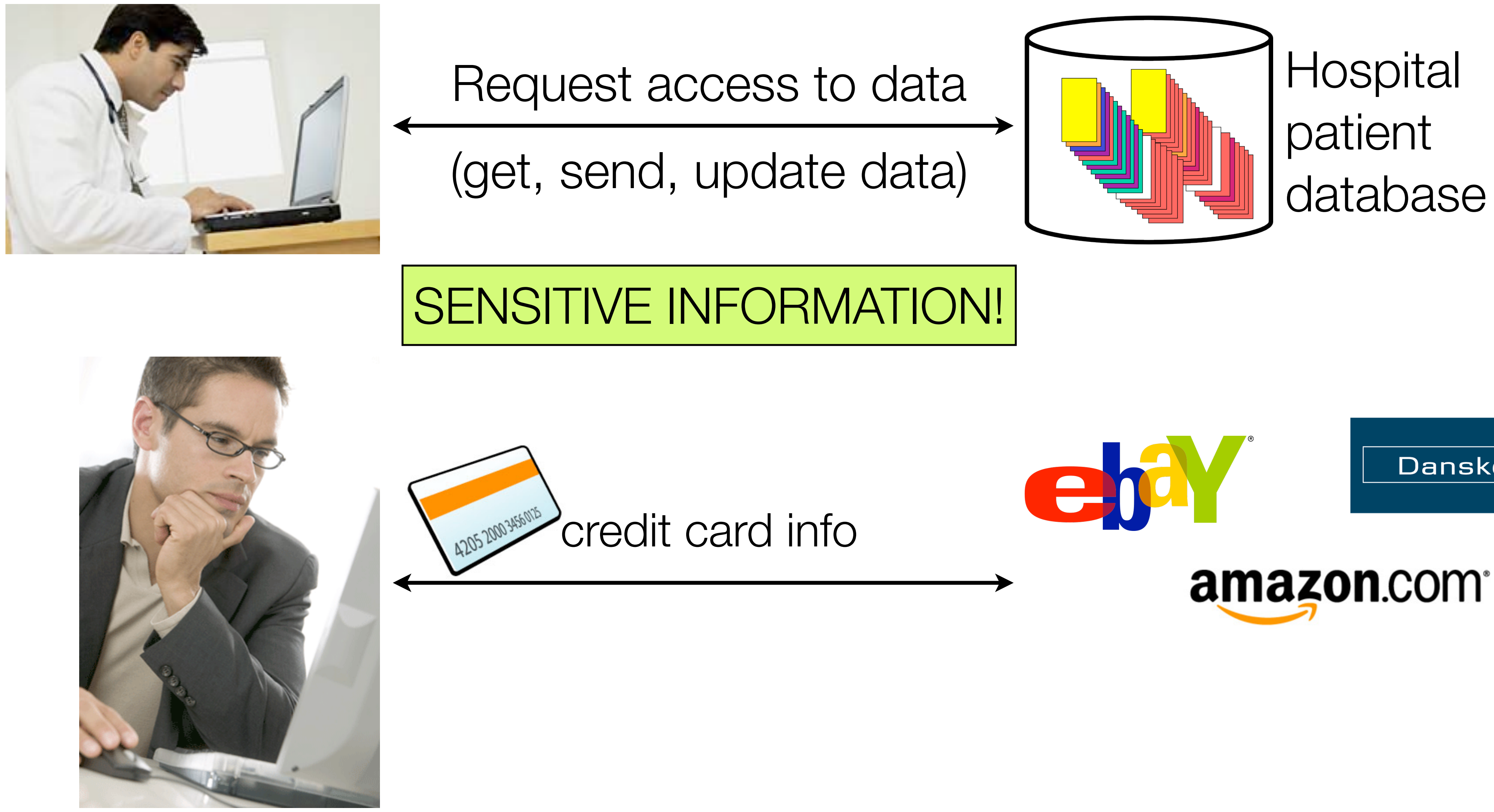


Protection against alteration or corruption

Protection against interference with the means to access the resources

Security in Distributed Systems

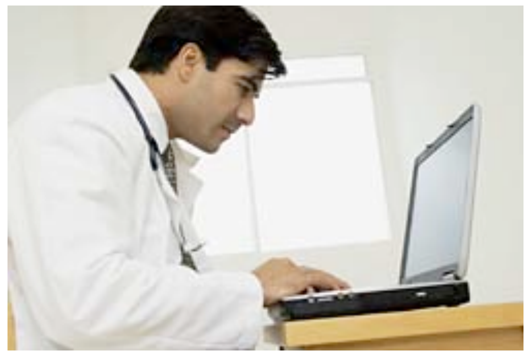
- In a distributed system, clients send requests to access data managed by servers, which involves sending information in messages over a network.



Security in Distributed Systems

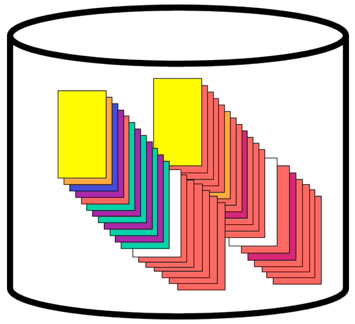
- In a distributed system, clients send requests to access data managed by servers, which involves sending information in messages over a network.

encryption techniques



Request access to data

(get, send, update data)



Hospital patient database

SENSITIVE INFORMATION!

credit card info

encryption techniques

Danske Bank

amazon.com

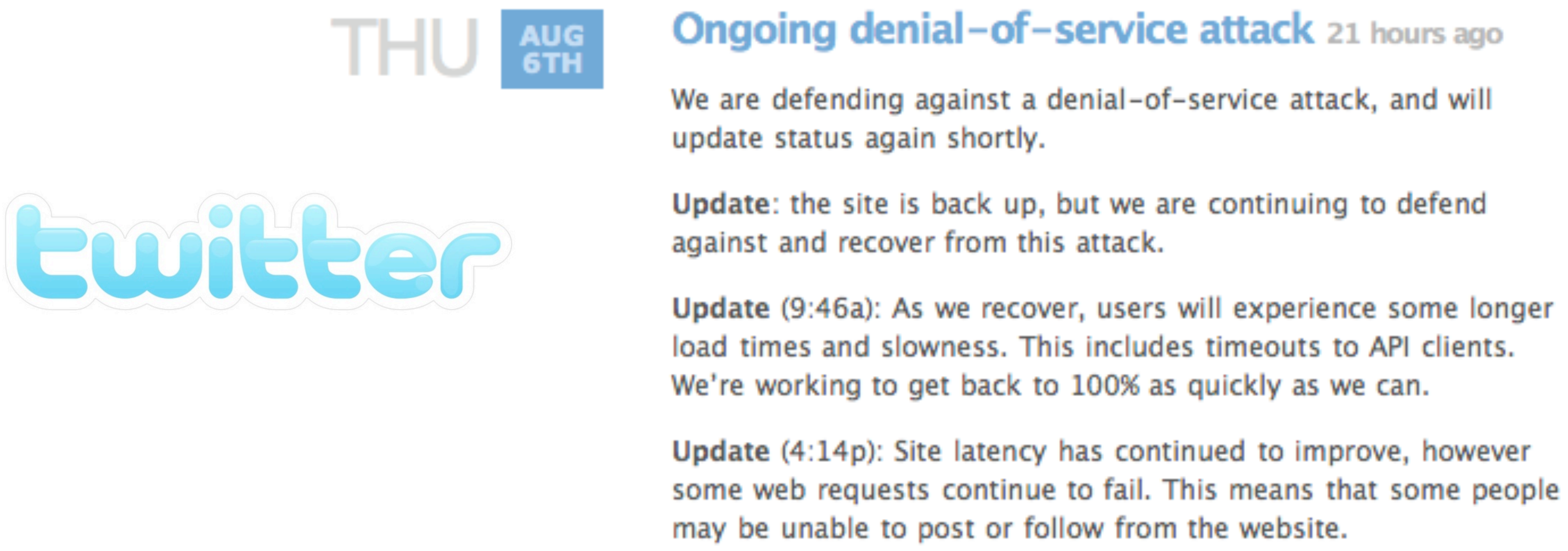
Open Security Challenge: Denial of Service Attack

- A bad guy may wish to disrupt a service for some reason:
  - he bombards the service with such a large number of pointless requests that the serious users are unable to use it.

Open Security Challenge: Denial of Service Attack

- A bad guy may wish to disrupt a service for some reason:
  - he bombards the service with such a large number of pointless requests that the serious users are unable to use it.

- On August 6, 2009, Twitter was shut down for hours due to a DoS attack:



THU AUG 6TH

twitter

Ongoing denial-of-service attack 21 hours ago

We are defending against a denial-of-service attack, and will update status again shortly.

Update: the site is back up, but we are continuing to defend against and recover from this attack.

Update (9:46a): As we recover, users will experience some longer load times and slowness. This includes timeouts to API clients. We're working to get back to 100% as quickly as we can.

Update (4:14p): Site latency has continued to improve, however some web requests continue to fail. This means that some people may be unable to post or follow from the website.

Scalability

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

- The Internet provides an illustration of a distributed system in which the number of computers and services has increased dramatically.

Date	*Computers*	*Web servers*
1979, Dec.	188	0
1989, July	130,000	0
1999, July	56,218,000	5,560,866
2003, Jan.	171,638,297	35,424,956

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
		42,298,371	

Scalability Challenges

1. Controlling the cost of physical resources: as the demand for a resource grows, it should be possible to extend the system, *at reasonable cost*, to meet it.

- In general, for a system with n users to be scalable, the quantity of physical resources required to support them should be at most O(n) (i.e., proportional to n).

- Example: is a single file server can support 20 users, then two such servers should be able to support 40 users.

Scalability Challenges

2. Controlling the performance loss: for a system to be scalable, the maximum performance loss should be no worse than O(log n), where n is the size of the set of data to be accessed.

- O(log n) is the time taken to access *hierarchically structured data*.
- Algorithms that use hierarchic structures scale better than those that use linear structures.
- But even with hierarchic structures an increase in size will result in some loss of performance.
- Frequently accessed data can be replicated.

Scalability Challenges

3.Preventing software resources running out

Example: Internet IP addresses (computer addresses in the Internet)

- In the late 1970s, it was decided to use 32 bits, but the supply of available Internet addresses is running out.
- For this reason, a new version of the protocol with 128-bit Internet addresses is being adopted and this will require modifications to many software components.

Scalability Challenges

3.Preventing software resources running out

Example: Internet IP addresses (computer addresses in the Internet)

- In the late 1970s, it was decided to use 32 bits, but the supply of available Internet addresses is running out.
- For this reason, a new version of the protocol with 128-bit Internet addresses is being adopted and this will require modifications to many software components.

- How to solve this problem? Not easy!
  - It is difficult to predict the demand that will be put on a system years ahead.
  - Over-compensating for future growth may be worse than adapting to a change when we are forced to (for instance, larger Internet addresses will occupy extra space in messages and in computer storage).

Scalability Challenges

4. Avoiding performance bottleneck: in general, algorithms should be decentralized to avoid performance bottlenecks.

Example: Domain Name System (DNS) (an Internet service that translates domain names into IP addresses).

- In the predecessor of DNS, a name table was kept in a single master file that could be downloaded to any computers that needed it.
- Fine when there were only a few hundred computers in the Internet!
- It soon became a serious performance and administrative bottleneck!

Scalability Challenges

4.Avoiding performance bottleneck: in general, algorithms should be decentralized to avoid performance bottlenecks.

Example: Domain Name System (DNS) (an Internet service that translates domain names into IP addresses).

- In the predecessor of DNS, a name table was kept in a single master file that could be downloaded to any computers that needed it.
- Fine when there were only a few hundred computers in the Internet!
- It soon became a serious performance and administrative bottleneck!

- In distributed systems, some shared resources are accessed very frequently.

Example: many users may access the same Web page, causing a decline in performance.

- We shall see that caching and replication may be used to improve the performance of resources that are very heavily used.

Failure Handling

- Computer systems *sometimes* fail.
- When faults occur in hardware or software, programs may produce incorrect results or they may stop before they have completed the intended computation.
- Failures in distributed systems are partial:
  - any process, computer or network may fail independently of the others.
  - some components fail while others continue to function.
- Therefore the handling of failures in distributed systems is particularly difficult.

Concurrency

- Both services and applications provide resources that can be shared by different clients in a distributed system.

- There is therefore a possibility that several clients will attempt to access a shared resource at the same time.

Example: Online Auction

- A data structure that records bids for an auction may be accessed very frequently when it gets close to the deadline time.

- Each resource (servers, objects in applications, ...) must be designed to be safe in a concurrent environment.

Transparency

- Transparency: the concealment from the user and the application programmer of the separation of components in a distributed systems, so that the system is perceived as a whole rather than a collection of independent components.

- Aim: to make certain aspects of distribution invisible to the application programmer so that they need only be concerned with the design of their particular application.

- The ANSA Reference Manual and the International Organization for Standardization's Reference Model for Open Distributed Processing (RM-ODP) identify 8 forms of transparency.

Transparencies

Network transparency

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.