- **I. Introduction**
- **II. Fundamental Concepts of Distributed Systems**
  - Architecture models; network architectures: OSI, Internet and LANs; interprocess communication
- **III. Time and Global States**
  - Clocks and concepts of time; Event ordering; Synchronization; Global states
- **IV. Coordination**
  - Distributed mutual exclusion; Multicast; Group communication, Byzantine problems (consensus)
- **V. Distribution and Operating Systems**
  - Protection mechanisms; Processes and threads; Networked OS; Distributed and Network File Systems (NFSs)
- **VI. Middleware**
  - Middleware; Distributed object models; Remote invocation; CORBA; Name and directory services
- **VII. Security**
  - Security concepts; Cryptographic algorithms; Digital signatures; Authentication; Secure Sockets

Distributed Systems

Definitions

- “*A system in which hardware or software components located at networked computers communicate and coordinate their actions only by message passing.*” [Coulouris]
- “*A system that consists of a collection of two or more independent computers which coordinate their processing through the exchange of synchronous or asynchronous message passing.*”
- “*A distributed system is a collection of independent computers that appear to the users of the system as a single computer.*” [Tanenbaum]
- “*A distributed system is a collection of autonomous computers linked by a network with software designed to produce an integrated computing facility.*”

Distributed Systems

- **Computer Networks vs. Distributed Systems**
  - Computer Network: the autonomous computers are explicitly visible (have to be explicitly addressed)
  - Distributed System: existence of multiple autonomous computers is transparent
  - However,
    - many problems in common,
    - in some sense networks (or parts of them, e.g., name services) are also distributed systems, and
    - normally, every distributed system relies on services provided by a computer network.

Distributed Systems

- **Reasons for distributing systems**
  - Functional distribution: computers have different functional capabilities.
    - Client / server
    - Host / terminal
    - Data gathering / data processing

    -> sharing of resources with specific functionalities
  - Inherent distribution stemming from the application domain, e.g.
    - cash register and inventory systems for supermarket chains
    - computer supported collaborative work
  - Load distribution / balancing: assign tasks to processors such that the overall system performance is optimized.

Distributed Systems

- **Reasons for distributing systems**
  - Replication of processing power: independent processors working on the same task
    - distributed systems consisting of collections of microcomputers may have processing powers that no supercomputer will ever achieve
      - 10000 CPUs, each running at 50 MIPS, yields 500000 MIPS,
        - then instruction to be executed in 0.002 nsec
        - equivalent to light distance of 0.6 mm
        - any processor chip of that size would melt immediately
  - Physical seperation: systems that rely on the fact that computers are physically seperated (e.g., to satisfy reliability requirements).
  - Economics: collections of microprocessors offer a better price/performance ration than large mainframes
    - mainframes: 10 times faster, 1000 times as expensive

Distributed Systems

- **Why Distributed Systems and not isolated hardware?**
  - Need to share data and resources amongst users
  - Enhance person-to-person communication
  - Flexibility: different computers with different capabilities can be shared amongst users
- **Problems with distributed, connected systems**
  - Software - how to design and manage it in a DS
  - Dependency on the underlying network infrastructure (the world wide wait…)
  - Easy access to shared data raises security concerns

Distributed Systems

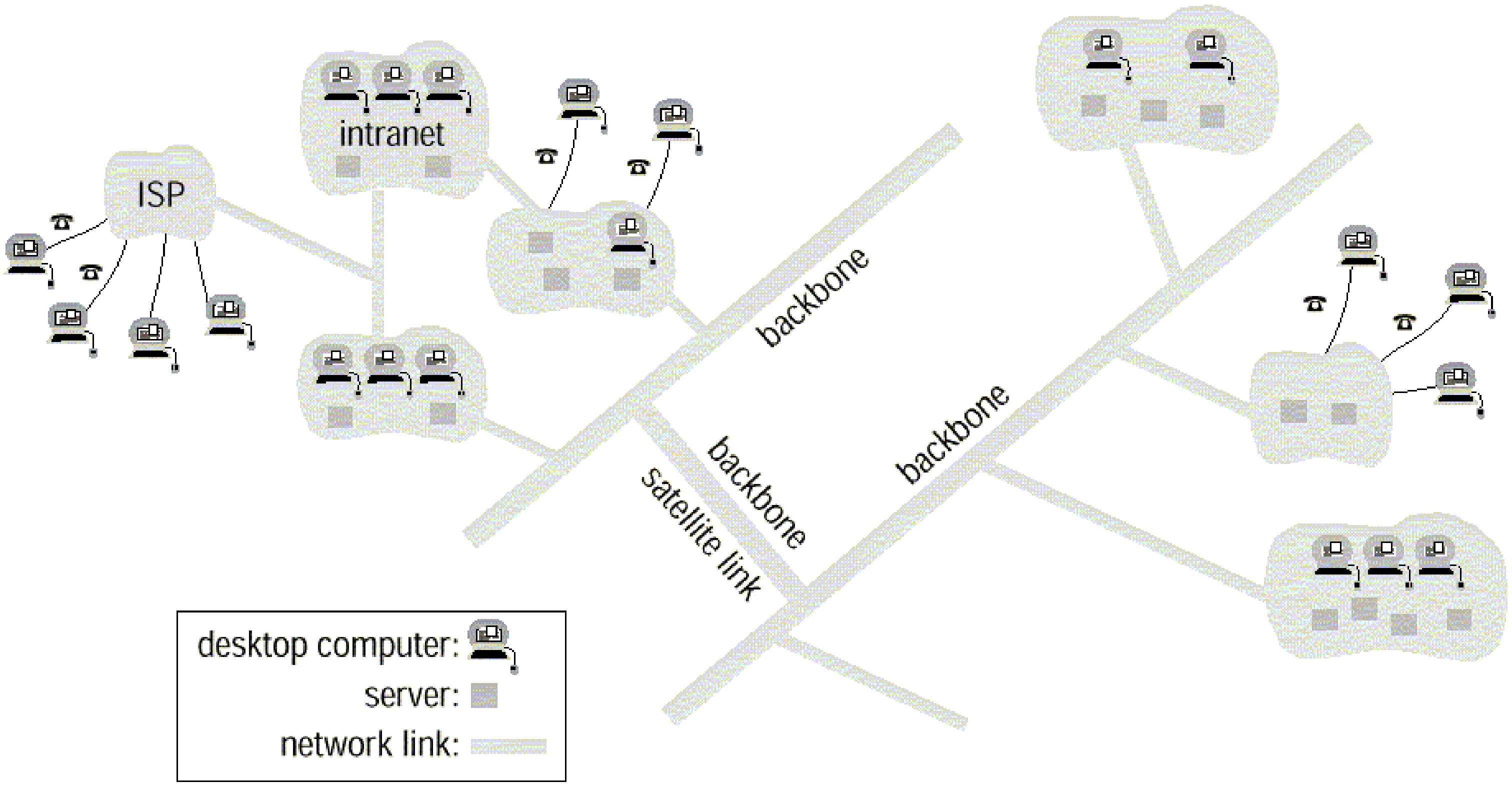
- **Consequences**
  - Distributed systems are concurrent systems
    - Every software or hardware component is autonomous
      - In the sequel, we will call such an autonomous component a "process"
        - Difference process/program
    - Components execute concurrent tasks
      - A and B are concurrent if either A can happen before B, or B can happen before A
    - Synchronization and coordination by message passing
    - Sharing of resources
    - Typical problems of concurrent systems
      - Deadlocks
      - Lifelocks
      - Unreliable communication
  - Absence of a global clock
    - Due to asynchronous message passing there are limits on the precision with which processes in a distributed system can synchronize their clocks

Distributed Systems

- **Consequences (cont.)**
  - Absence of a global state
    - In the general case, there is no single process in the distributed system that would have a knowledge of the current global state of the system
      - Due to concurrency and message passing communication
  - Specific failure modes
    - Processes run autonomously, in isolation
      - Failures of individual processes may remain undetected
      - Individual processes may be unaware of failures in the system context

Distributed Systems

- **Examples of Distributed Systems**
  - 1. The Internet
    - Heterogeneous network of computers and applications
    - Implemented through the Internet Protocol Stack
    - Typical configuration:



© Pearson Education 2001

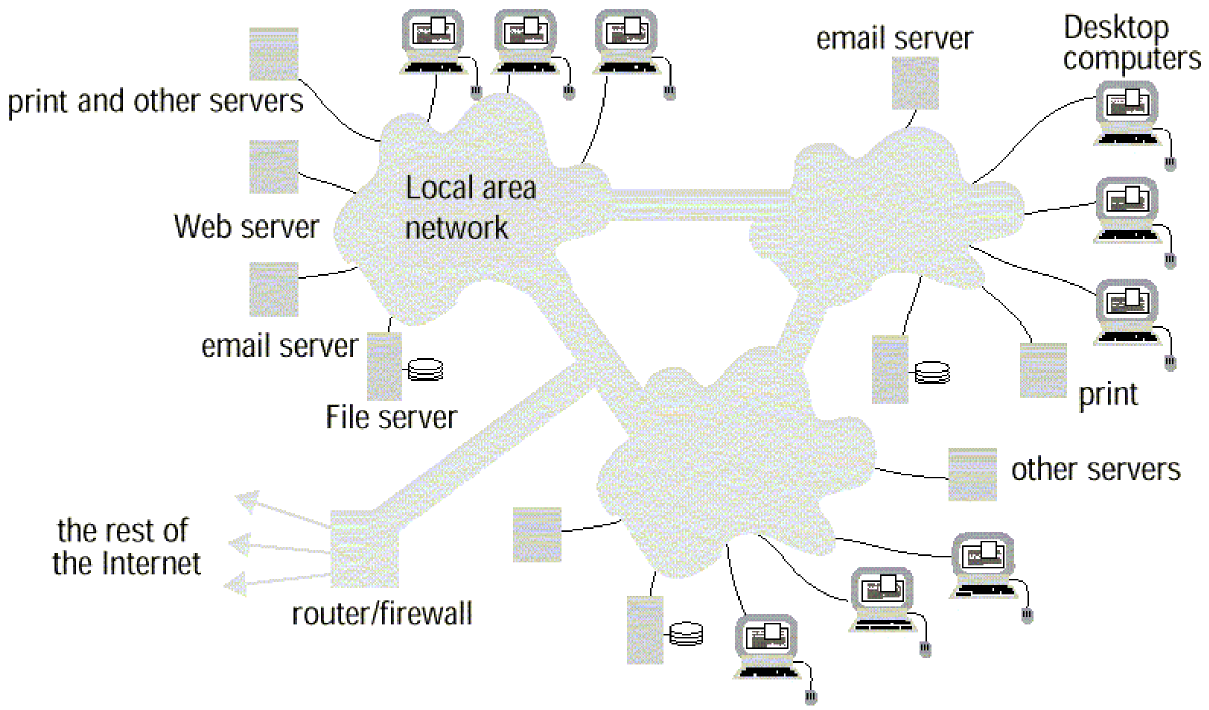
Distributed Systems

Examples of Distributed Systems

- 2. Distributed Multimedia-Systems
  - Often use Internet infrastructure
  - Characteristics
    - Heterogeneous data sources and sinks that need to be synchronized in real time
      - Video
      - Audio
      - Text
    - Often: Distribution services
      - Multicast
  - Examples
    - Teleteaching tools (mbone-based, etc.)
    - Video-conferencing
    - Video and audio on demand

Distributed Systems

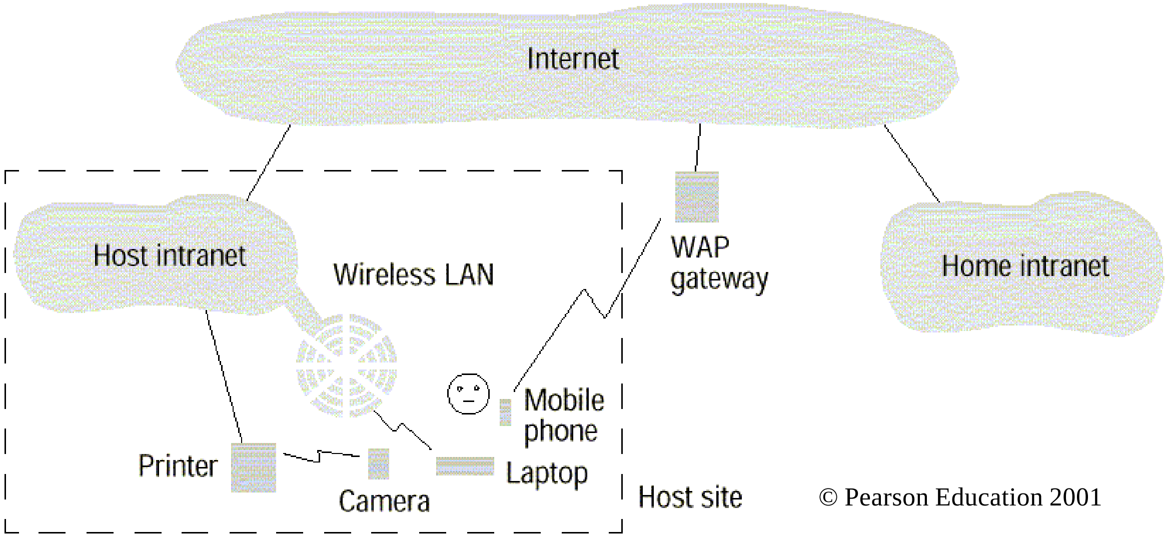
- **Examples of Distributed Systems**
  - 3. Intranets
    - Locally administered network
    - Usually proprietary (e.g., the University campus network)
    - Interfaces with the Internet
      - Firewalls
    - Provides services internally and externally



© Pearson Education 2001

Examples of Distributed Systems

- 4. Mobile and Ubiquitous Computing Systems
  - Cellular phone systems (e.g., GSM, UMTS)
    - Resources being shared
      - * Radio frequencies
      - * Transmission times on one frequency (UMTS: multiplexing)
      - * The mobile on the move
  - Laptop computers
    - Wireless LANs (faculty campus WLAN, "MoPo")
  - Handheld devices, PDAs etc.
  - Wearable devices



© Pearson Education 2001

Distributed Systems

- **Examples of Distributed Systems**
  - 5. Embedded systems
    - The networked coffee mug
    - Avionics control systems
      - Flight management systems in aircraft
    - Automotive control systems
      - Mercedes S-Klasse automobiles these days are equipped with 50+ autonomous embedded processors
      - Connected through proprietary bus-like LANs
    - Consumer Electronics
      - Audio HiFi equipment

Distributed Systems

Examples of Distributed Systems

- 6. Telephony systems
  - Examples
    - POTS
    - ISDN
    - Intelligent Networks
    - Advanced Intelligent Networks
  - Shared resources
    - Network
    - Management
    - Phones
- 7. Network management
  - Administration of network resources
  - State: resource and connection status
  - Example
    - SNMP

Distributed Systems

- **Examples of Distributed Systems**
  - 8. Network File Systems
    - Architecture to access file systems across a network
    - Famous example
      - Network File System (NFS), originally developed by SUN Microsystems for remote access support in a UNIX context
  - 9. The World Wide Web
    - Open client-server architecture implemented on top of the Internet
    - Shared resouces
      - Information, uniquely identified through a Uniform Resource Locator (URL)
    - Variants: Intranet-based Webs

Challenges in the design of Distributed Systems

- 1. **Heterogeneity** of
    - underlying network infrastructure,
    - computer hard- and software (e.g., operating systems, compare UNIX socket and Winsock calls),
    - programming languages (in particular, data representations).
- Some approaches
    - Middleware (e.g., CORBA): transparency of network, hard- and software and programming language heterogeneity
    - Mobile Code (e.g., JAVA): transparency from hard-, software and programming language heterogeneity through virtual machine concept

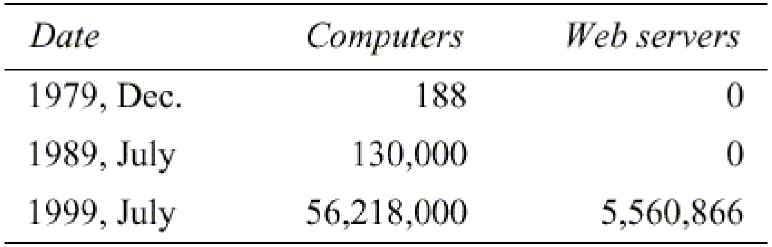
Distributed Systems

- **Challenges in the design of Distributed Systems**
  - 2. **Openness**
    - Ensure extensibility and maintainability of systems
      - Adherence to standard interfaces
  - 3. **Security**
    - Privacy
    - Authentication
    - Availability

    more about this later.

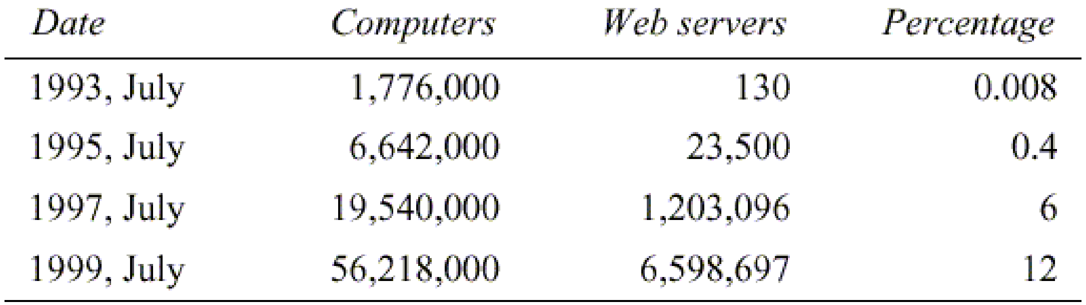
Challenges in the design of Distributed Systems

- 4. **Scalabity**



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

Computers in 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

Computers vs. Web Servers in the Internet

© Pearson Education 2001

- Does the system remain effective given expectable growth?
  - Physical resources
  - Control performance loss and performance bottlenecks
    - www.amazon.com is more than one computer
    - hierarchical structures in name serving
  - correct dimensioning of software resources
    - IP addresses: from 32 to 128 bits

Distributed Systems

- **Challenges in the design of Distributed Systems**
  - 5. Handling of **failures**
    - Detection (may be impossible)
    - Masking
      - retransmission
      - redundancy of data storage
    - Tolerance
      - exception handling (e.g., timeouts when waiting for a web resource)
    - Redundancy
      - redundant routes in network
      - replication of name tables in multiple domain name servers
  - 6. **Concurrency**
    - Consistent scheduling of concurrent threads (so that dependencies are preserved, e.g., in concurrent transactions)
    - Avoidance of dead- and lifelock problems

Distributed Systems

- **Challenges in the design of Distributed Systems**
  - 7. **Transparency**: concealing the heterogeneous and distributed nature of the system so that it appears to the user like one system.
    - Transparency categories (according to ISO's Reference Model for ODP)
      - *Access*: access local and remote resources using identical operations
        - e.g., network mapped drive using Samba server, NFS mounts
      - *Location*: access without knowledge of location of a resource
        - e.g., URLs, email addresses
      - *Concurrency*: allow several processes to operate concurrently using shared resources in a consistent fashion
      - *Replication*: use replicated resource as if there was just one instance
      - *Failure*: allow programs to complete their task despite failures
        - retransmit of email messages
      - *Mobility*: allow resources to move around
        - e.g., 700 phone number - URLs are not!
      - *Performance*: adaption of the system to varying load situations without the user noticing it
      - *Scaling*: allow system and applications to expand without need to change structure or application algorithms