Introduction to Distributed Systems

Brian Nielsen

bnielsen@cs.aau.dk

About me

- Associate Professor
- Distributed Systems and Semantics Research Unit
- Department of Computer Science
- Aalborg University, Denmark

AALBORG UNIVERSITY

- CISS: Center of Embedded Software Systems
- Research
  - Distributed programming (Linda w. multiple tuple spaces)
  - Distributed Multi-media
  - Real-Time Operating Systems
  - Automated testing of embedded, real-time, distributed systems

About You

- ??
- $\Rightarrow$ Different background and expectations

Motivation

- Nearly all modern computer based systems are network and operate dependently on other systems.
- Distributed systems are **fundamentally different** from centralized systems.
- Therefore**, all** students should obtain knowledge about concepts in distributed systems, knowledge about their construction, and an understanding of advantages and disadvantages of their use.

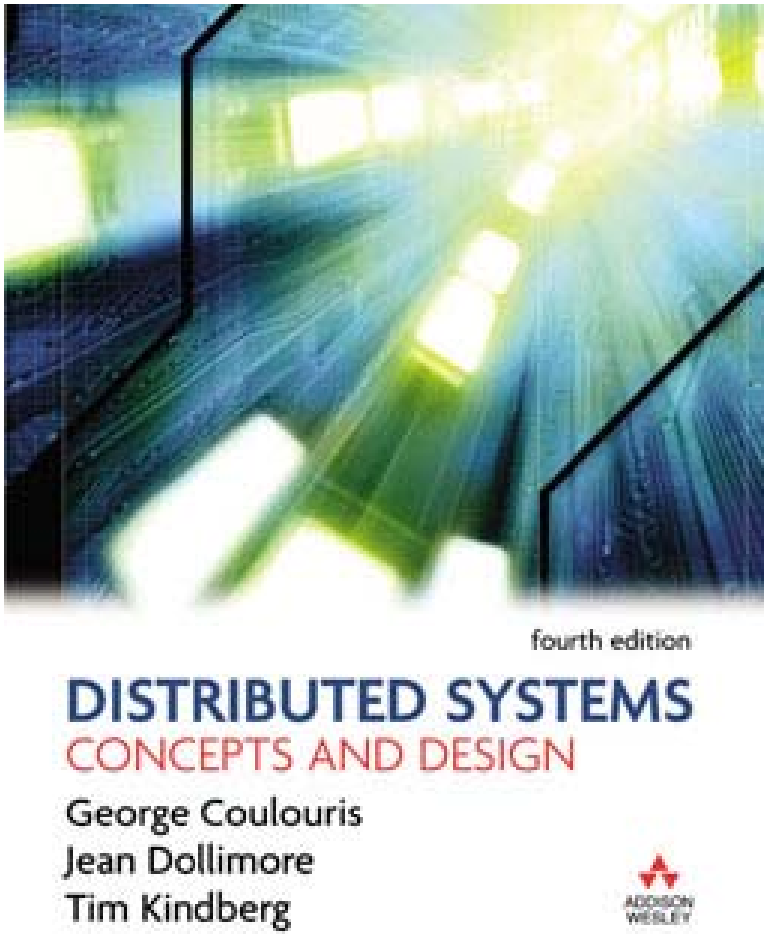
Goals

- Knowledge and overview of course topics
- Use correct terminology
- Fundamental properties of DS, their consequences on architecture, and behavior
- Understand typical DS problems and algorithms for their solution
- Compare distributed algorithms wrt. semantics, performance, and fault tolerance
- Skill in implementation of basic distributed systems/algorithms

Course Form

- 3 ECTS (3*30 study hours), 15 lectures
- 1 lecture = 6 study hrs
  - 3*30 min of lectures
  - 1.5 hrs of exercises (in groups)
  - 1.5 hrs of reading homework
  - 0.5 hrs of exam-preparation
- 1 “big” study/programming assignment subject to examination
- Examination: ??

Text Book



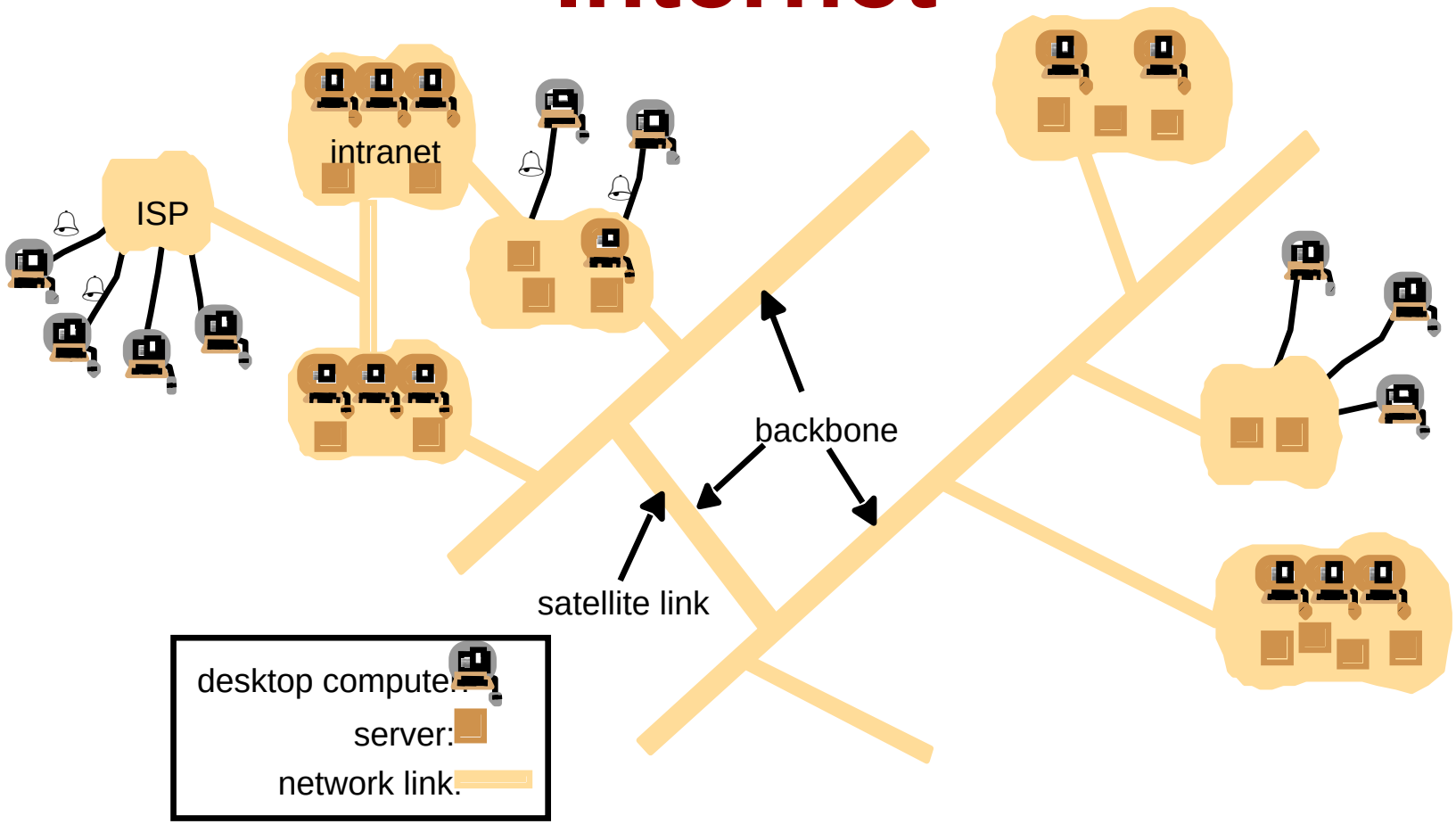
- Coulouris, Dollimore and Kindberg
- **Distributed Systems: Concepts and Design**
- Edition 4
- www.cdk4.net

Pre-requisites

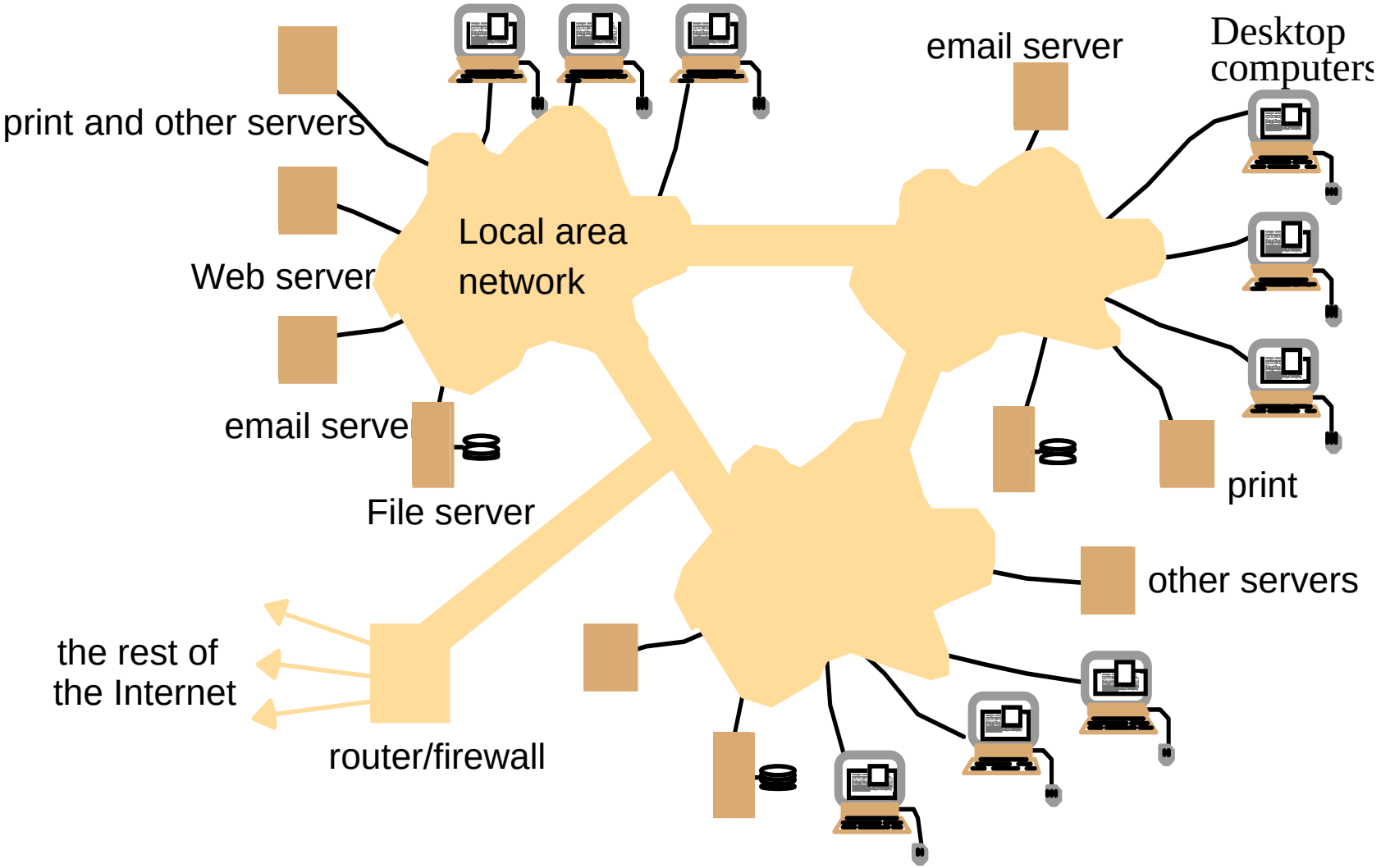
- Programming
  - Practical programming in e.g. Java, C, C++
  - Basic data-structures and algorithms
  - Preferably also concurrent programming
- Networks
  - IP-stack, IP-addressing, IP-routing, message enveloping, TCP/UDP, Sliding-window, congestion and flow control, socket-programming, basic encryption technology
  - Else read chapter 3
- Operating Systems
  - Processes, threads, concurrency, non-determinism, kernel and user level, synchronization (semaphores, monitors), address spaces, virtual memory, file-systems
  - Else read chapter 6

Examples of Distributed Systems

A typical portion of the Internet



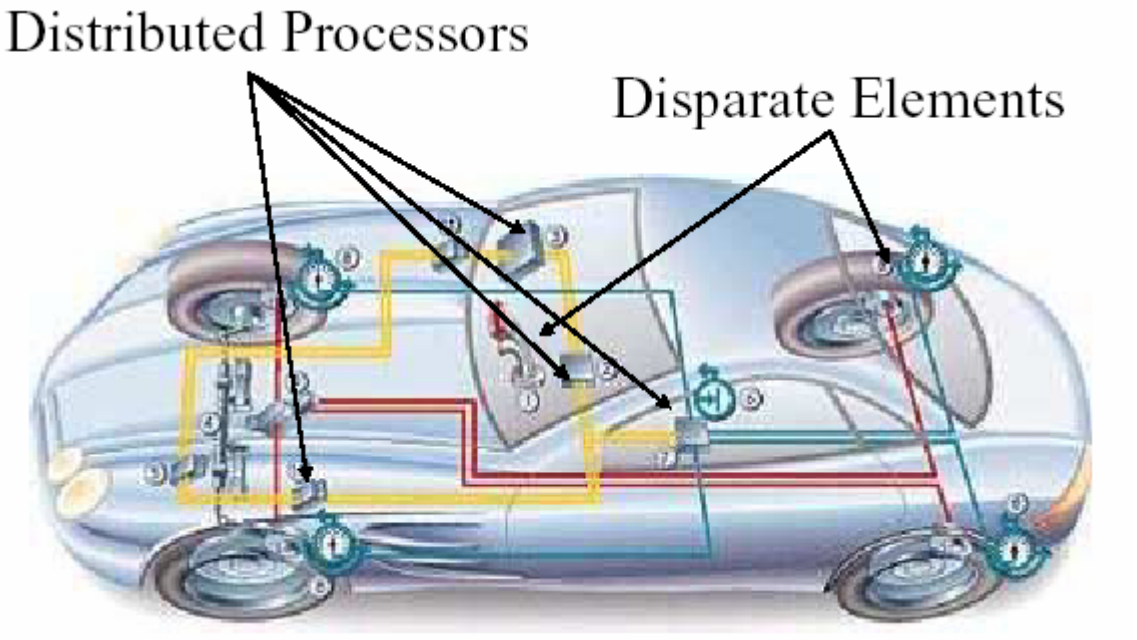
A typical intranet



Intranets

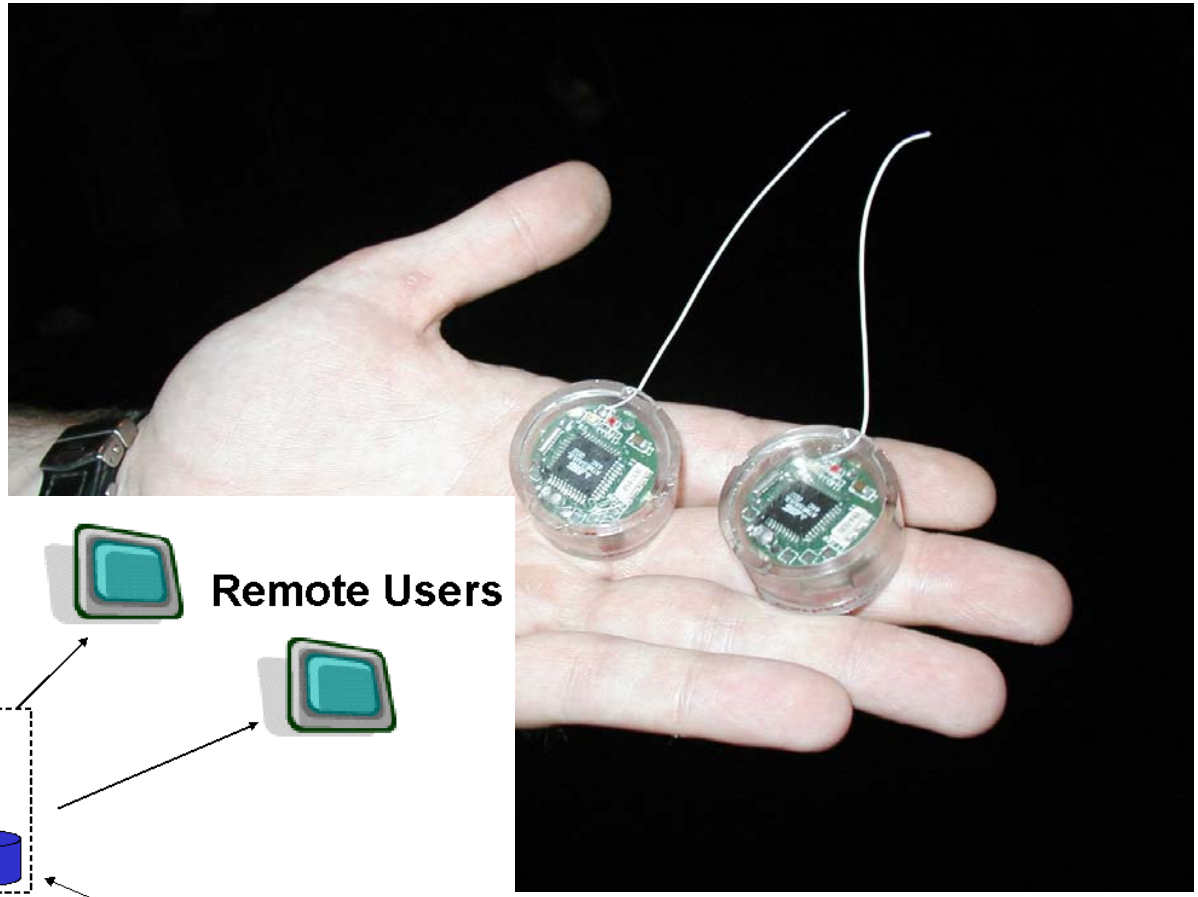
- A portion of the Internet that
  - is separately administered
  - usually proprietary
  - provides internal and external services
  - can be configured to enforce local security policies
    - may use a firewall to prevent unauthorized messages leaving or entering
  - may be connected to the internet via a router
- Services:
  - File, print services, backup, program-sharing, user-, system-administration, internet access

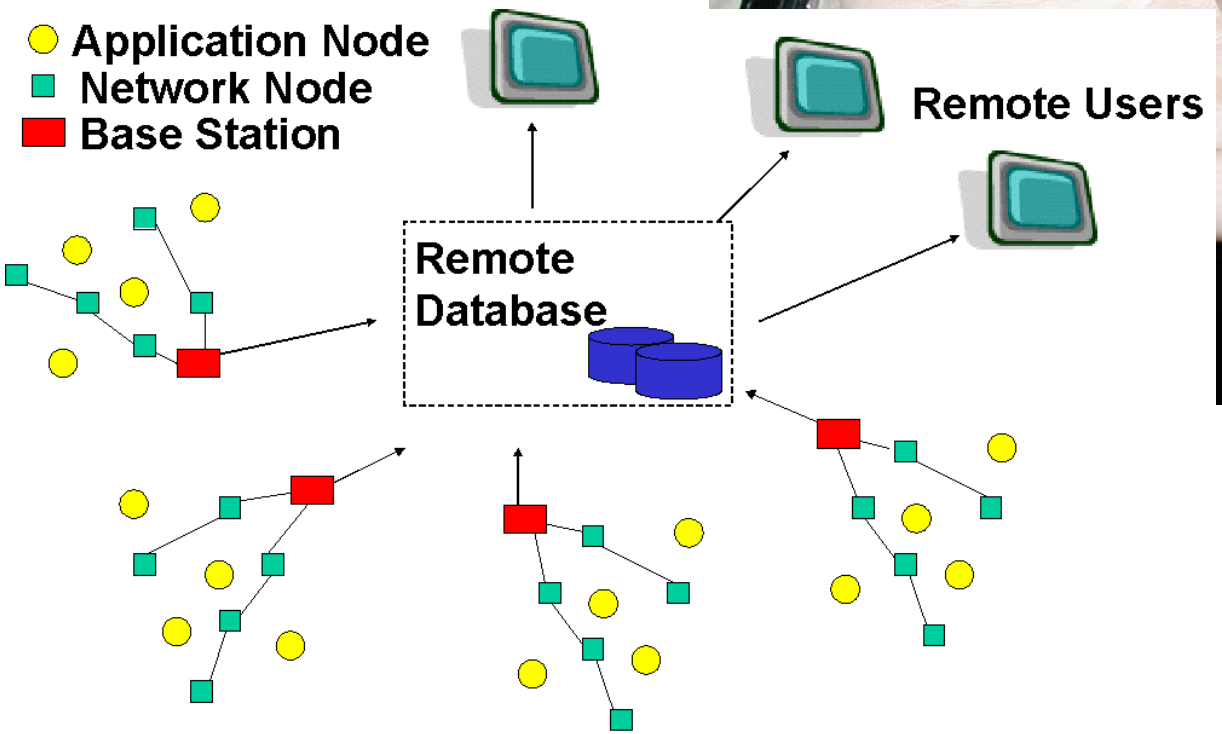
Automotive Control



- 80+ ECU's interconnected in controller area networks
- Vehicle dynamics (engine, brake, gear control,…)
- Instrumentation control (lights, indicators, windows,…)
- System Integration
- Information and entertainment systems
- Drive-by-wire

Sensor-networks





Distributed Computing

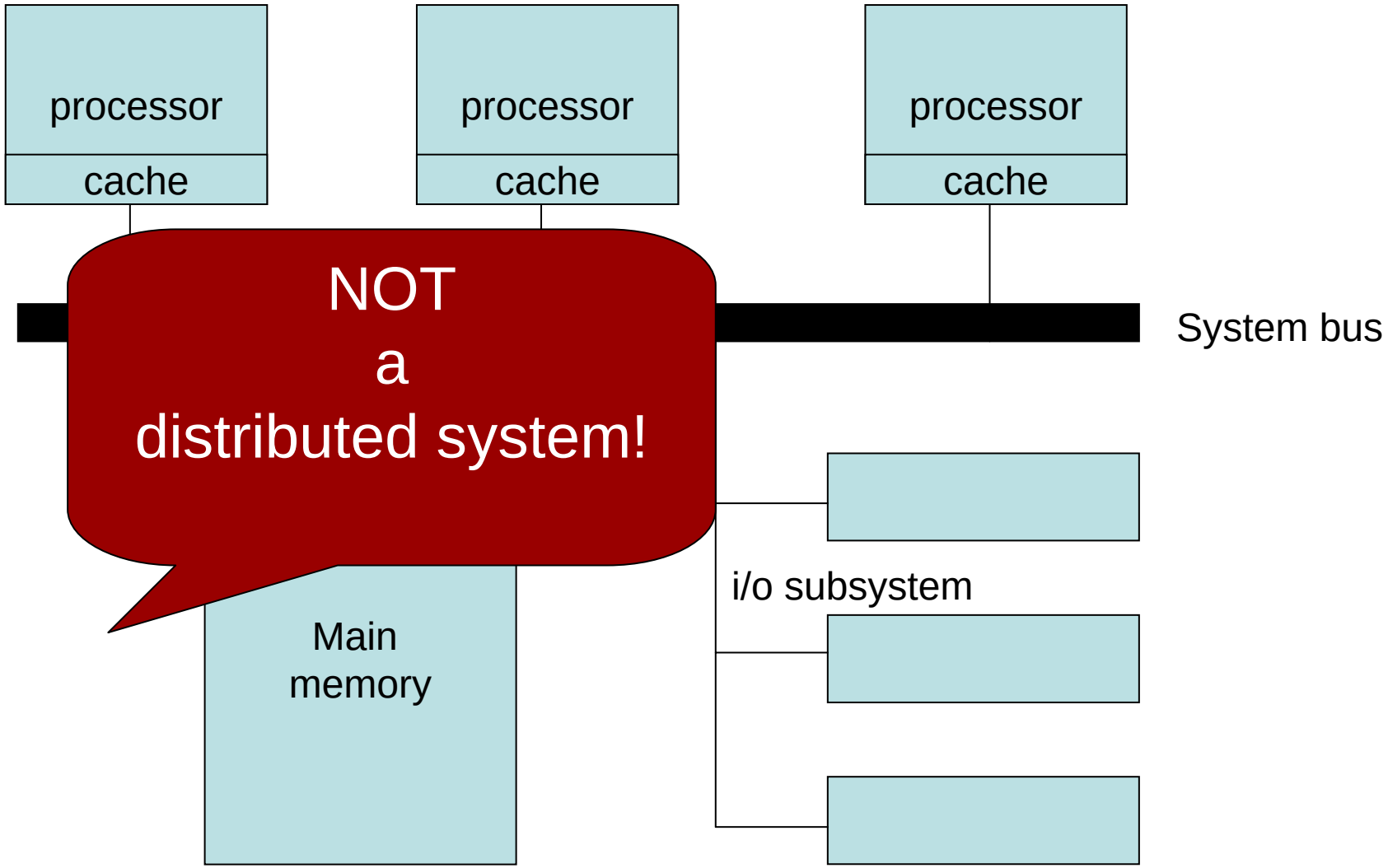
- Speed up huge computations by using multiple computers
- NOWs (network of workstations) / cluster computing

- Dedicated computers
- seti@home: project to scan data retrieved by a radio telescope to search for radio signals from another world;
- Grid Computing: www.grid.org (Millions CPUs world-wide – source 2004)

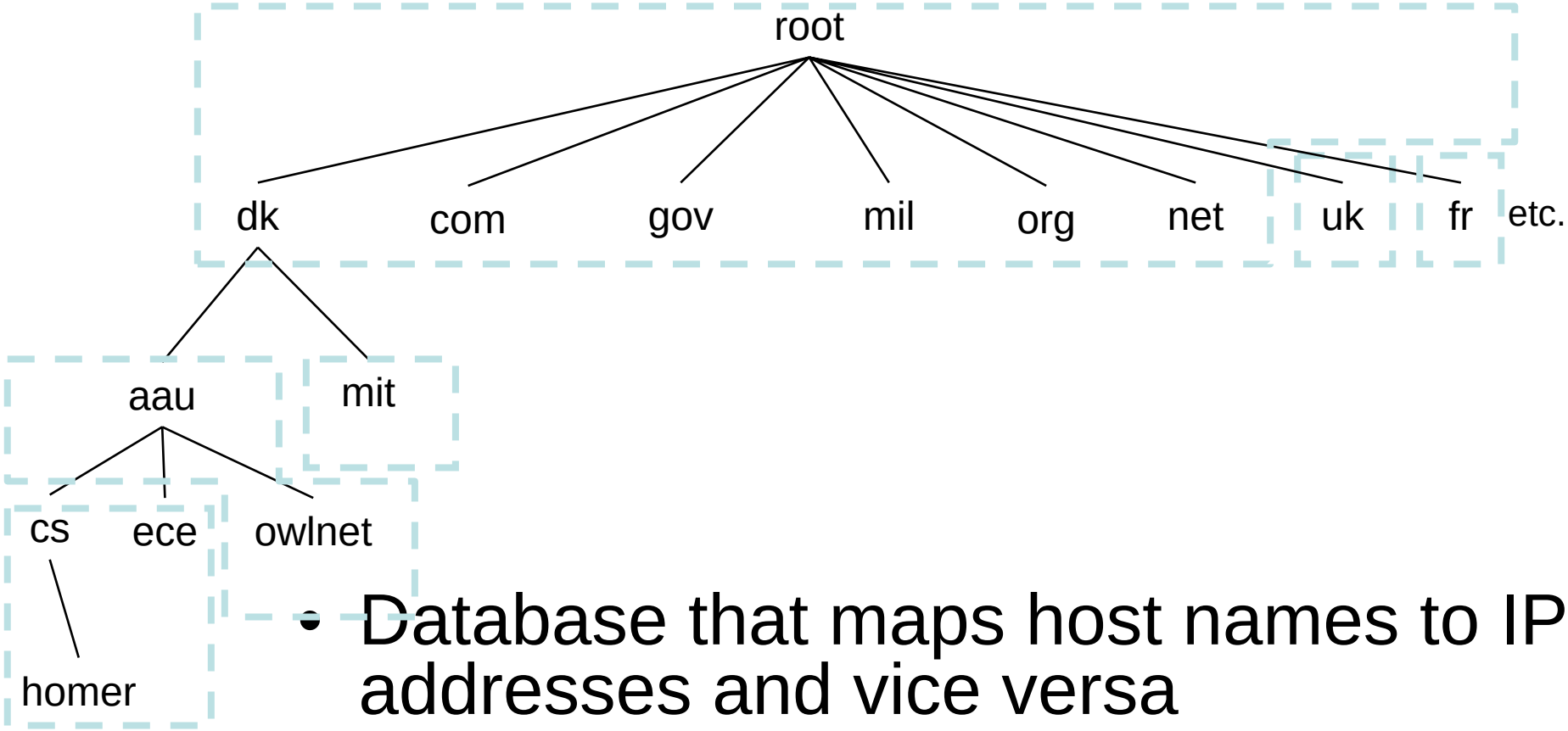
Mission-critical applications

- Embedded systems, automotive, avionics
- Control Systems
- Banking, stock markets, stock brokerages
- Health care, hospital automation
- Control of power plants, electric grid
- Telecommunications infrastructure
- Electronic commerce and electronic cash on the Web
- Corporate “information” base: a company’s memory of decisions, technologies, strategy
- Military command, control, intelligence systems
- …

Shared Memory Multi-Processor



Domain Name Service



- Database that maps host names to IP-addresses and vice versa
- homer.cs.aau.dk =130.225.194.13
- Client--server interaction on UDP Port 53

DNS: History

- Initially all host-addess mappings were in a file called hosts.txt (in /etc/hosts)
  - Changes were submitted to SRI (Stanford Research Institute) by email
  - New versions of hosts.txt ftp'd periodically from SRI
  - An administrator could pick names at their discretion
  - Any name is allowed: eugenesdesktopatrice (flat namespace)

- As the internet grew this system broke down because:
  - SRI couldn't handled the load
  - Hard to enforce uniqueness of names
  - Many hosts had inaccurate copies of hosts.txt

- Domain Name System (DNS) was born in '83

DNS zone data records

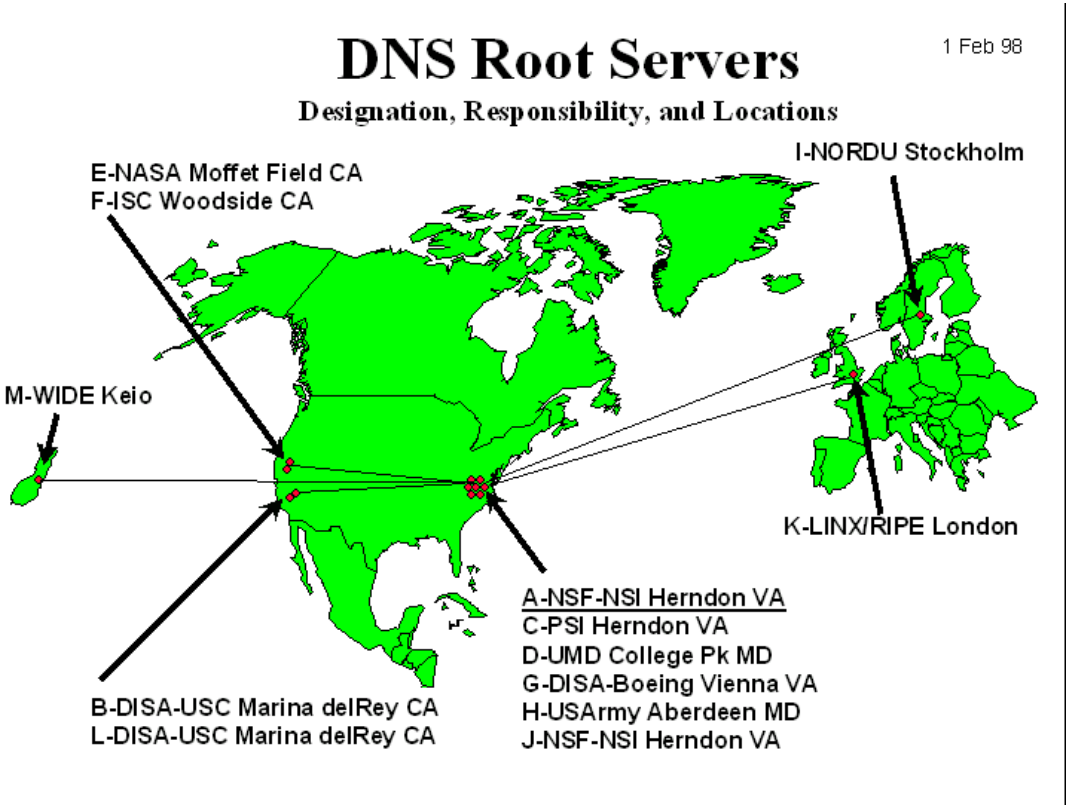
Example Domain dcs.qmul.ac.uk

domain name	*time to live*	*class*	*type*	*value*
www	1D	IN	CNAME	apricot
apricot	1D	IN	A	138.37.88.248
dcs	1D	IN	NS	dns0.dcs
dns0.dcs	1D	IN	A	138.37.88.249
dcs	1D	IN	NS	dns1.dcs
dns1.dcs	1D	IN	A	138.37.94.248
dcs	1D	IN	NS	cancer.ucs.ed.ac.uk

CNAME = Canonical name for an alias
A = IP Address
NS = Authorative name server

DNS: Root Name Servers

- Contacted by local name server that can not resolve name
- Root name server:
  - Contacts authoritative name server if name mapping not known
  - Gets mapping
  - Returns mapping to local name server
- ~ Dozen root name servers worldwide



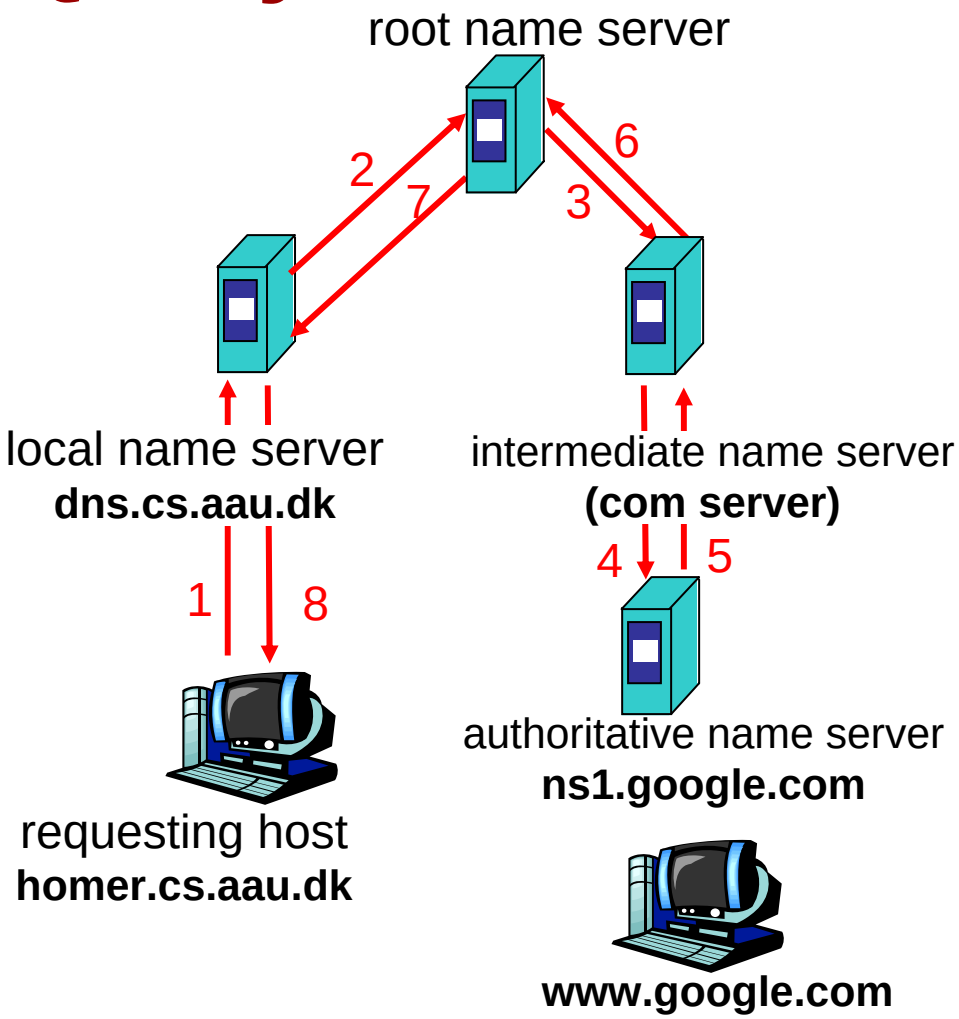
Example of Recursive DNS Query

Root name server:

- May not know authoritative name server
- May know intermediate name server: who to contact to find authoritative name server?

Recursive query:

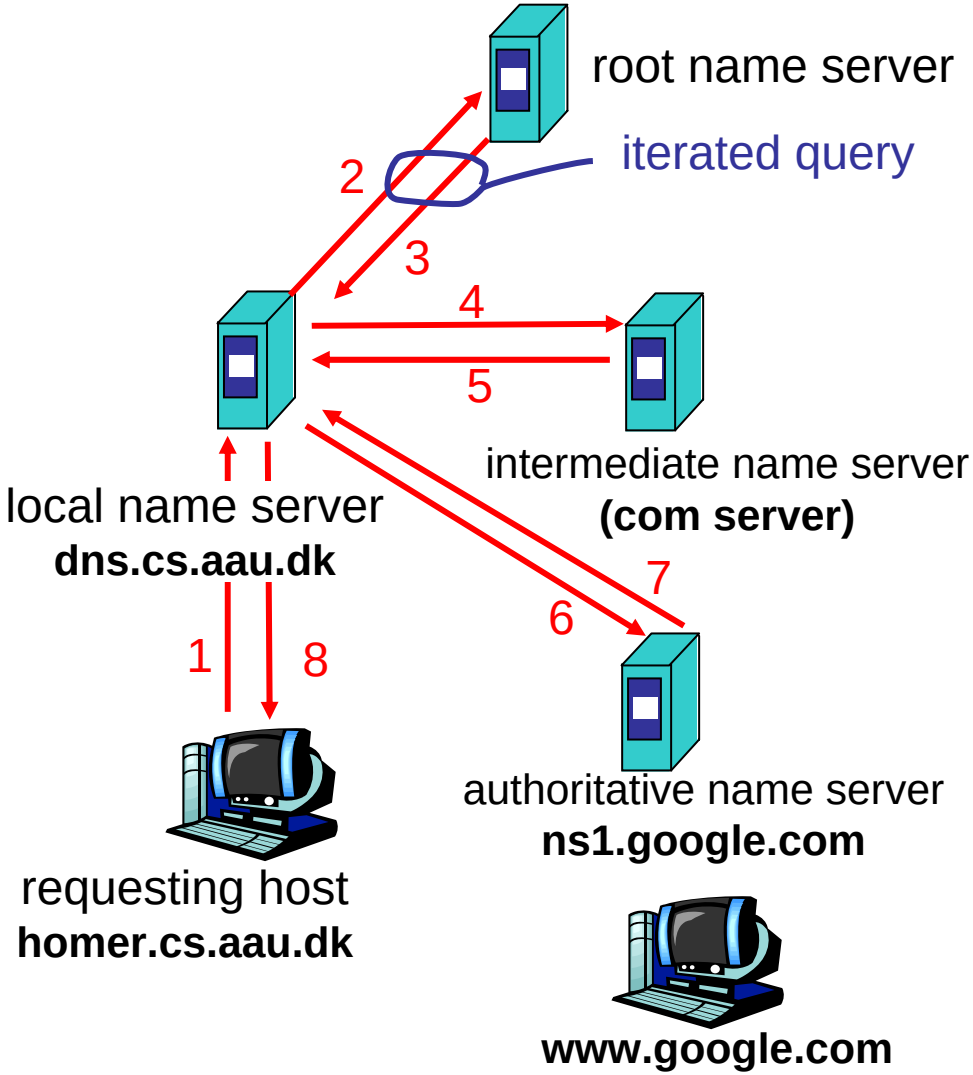
- Puts burden of name resolution on contacted name server
- Heavy load?



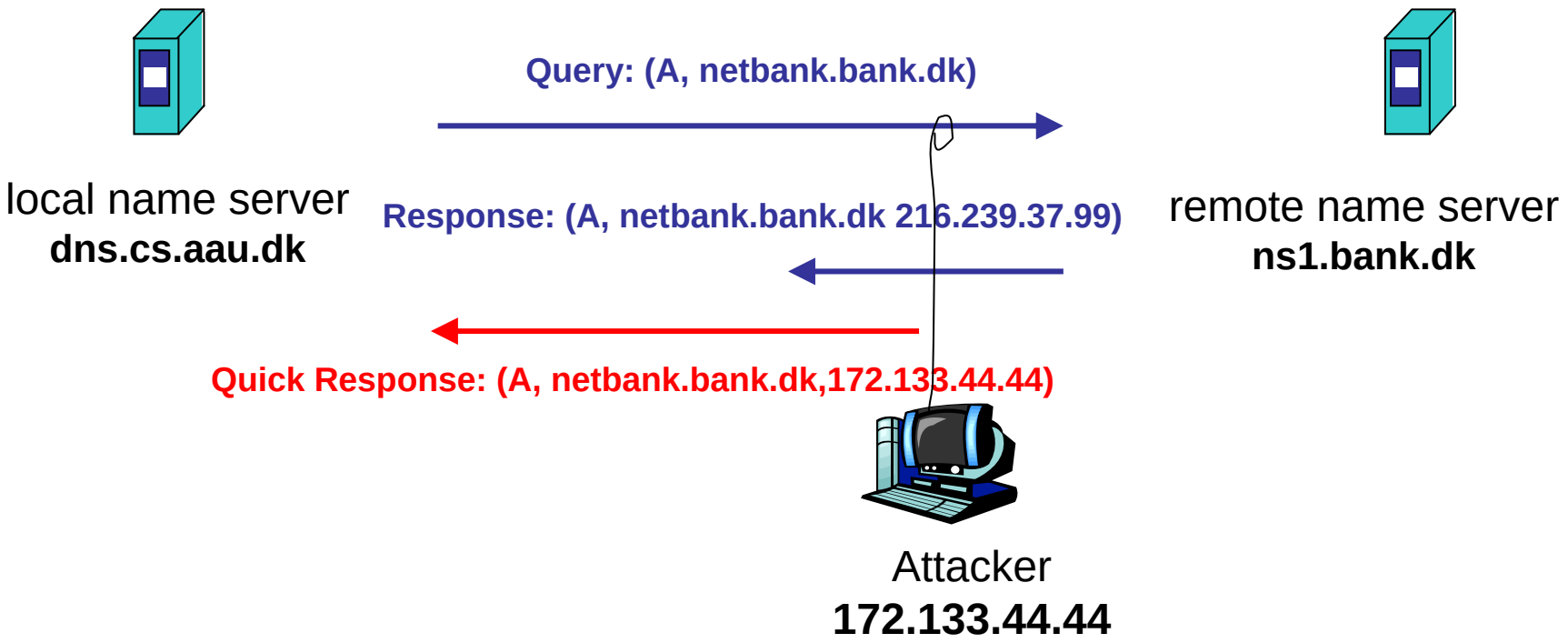
Example of Iterated DNS Query

Iterated query
  Contacted server replies with name of server to contact
- "I don't know this name, but ask this server" (delegation):

This is how today's DNS system behaves



DNS-spoofing



- Until the TTL expires, 172.133.44.44 serves netbank.dk

Definition

Definition

- *A **distributed system** is the one in which hardware and software components at **networked computers** communicate and coordinate their activity only by **passing messages.***
- Examples: Internet, intranet and mobile computing systems.

Consequences

- Concurrent execution of processes
  - Users work independently & share resources
  - non-determinism, race-conditions, synchronization, deadlock, liveness, …
- No global clock
  - Coordination is done by message exchange
  - There are limits to the accuracy with which computers in a network can synchronize their clocks
- No global state
  - Generally, there is no single process in the distributed system that would have a knowledge of the current global state of the system
- **Units may fail independently.**
  - Network faults can result in the isolation of computers that continue executing
  - A system failure or crash might not be immediately known to other systems

Why a Distributed System?

- Functional distribution
  - computers have different functional capabilities yet may need to **share resources**
    - Client / server
    - Data gathering / data processing
- Inherent distribution in application domain
    - physically or across administrative domains
    - cash register and inventory systems for supermarket chains
    - computer supported collaborative work
- Economics
  - collections of microprocessors offer a better price/ performance ratio than large mainframes

Why a Distributed System?

- Better performance
  - Load balancing
  - Replication of processing power
- Increased Reliability
  - Exploit independent failures property and
  - Redundancy

Models

Architectural

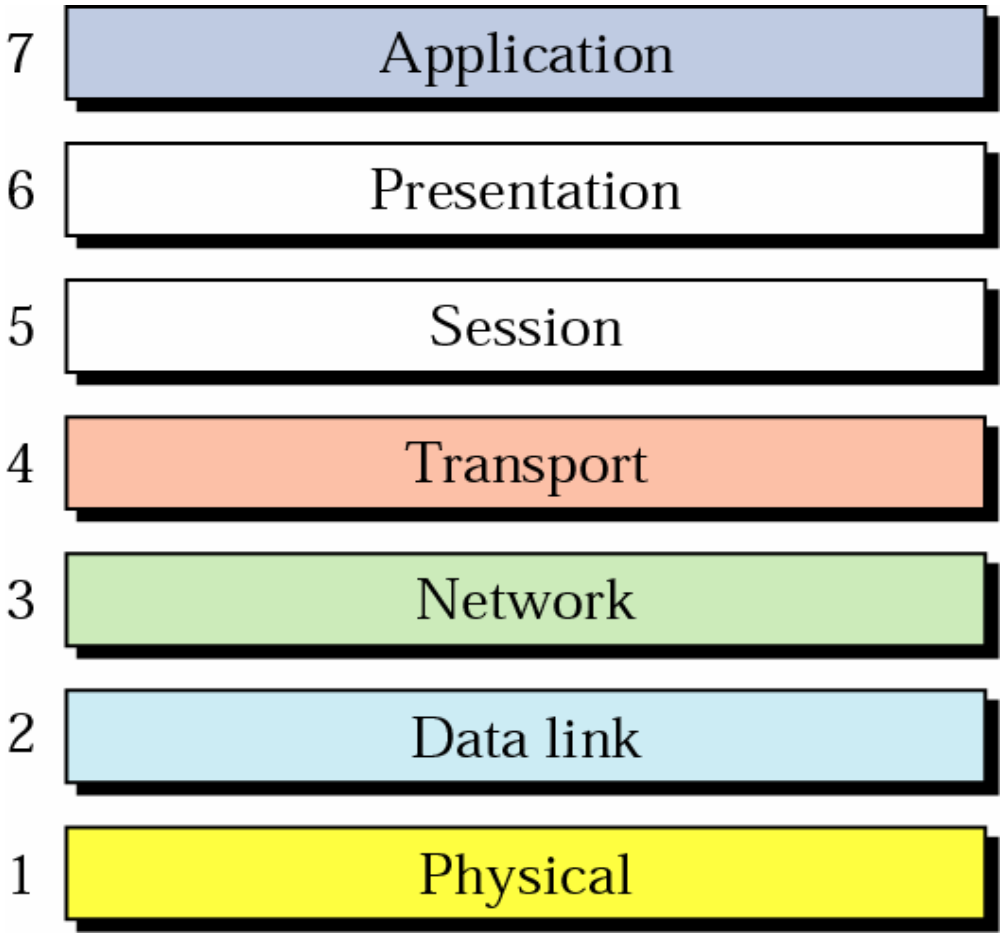
Fundamental (semantic assumptions)

Architectural models

- Software layers
- System architecture

OSI-model

- **Open Systems Interconnection model (ISO standard)**

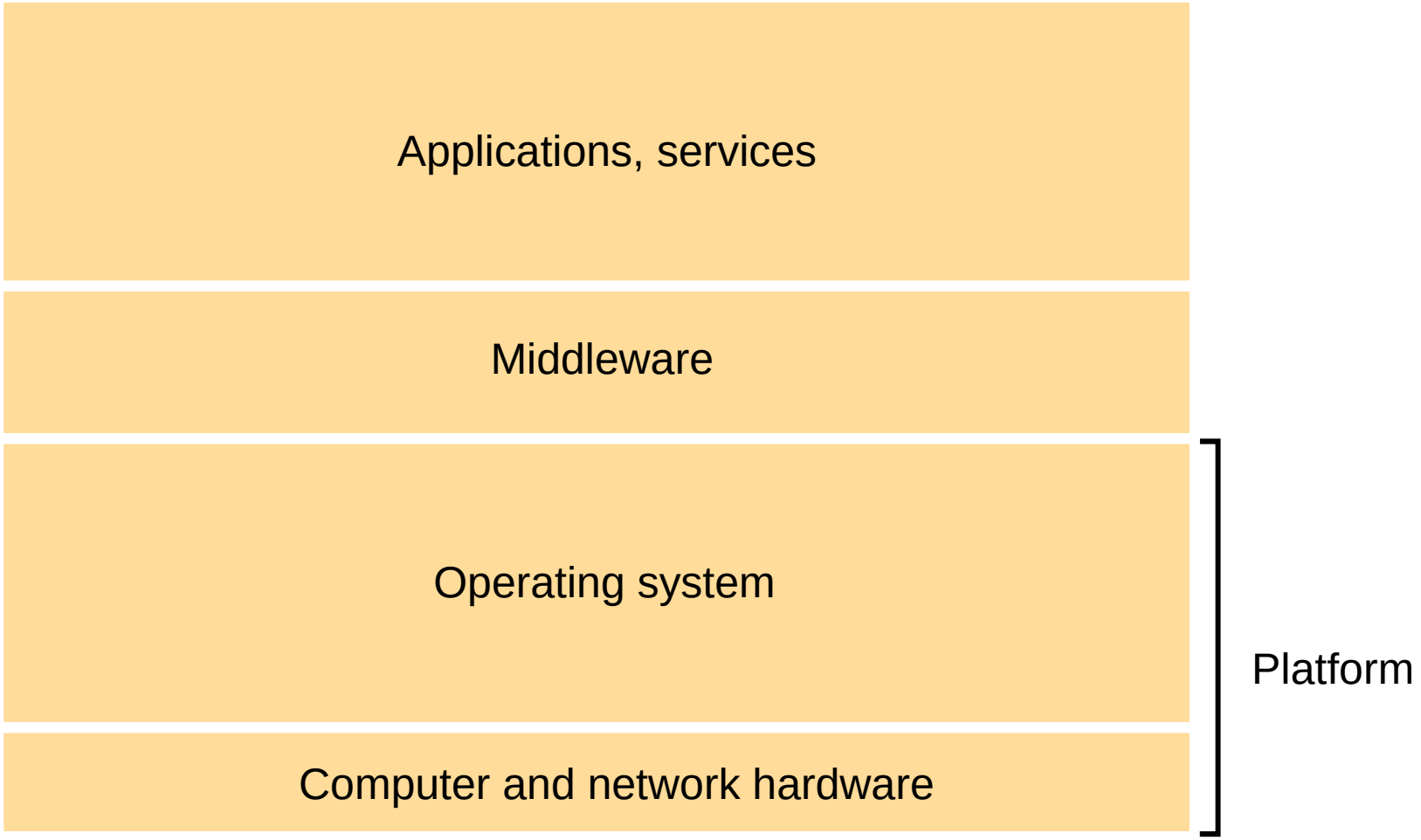


- **Session Layer:** Dialog controller
  - Establish
  - Maintain
  - Synchronize
  - Terminate

Presentation layer: handles syntax and semantics

- Data translation
- Encryption/decryption
- Compression/expansion

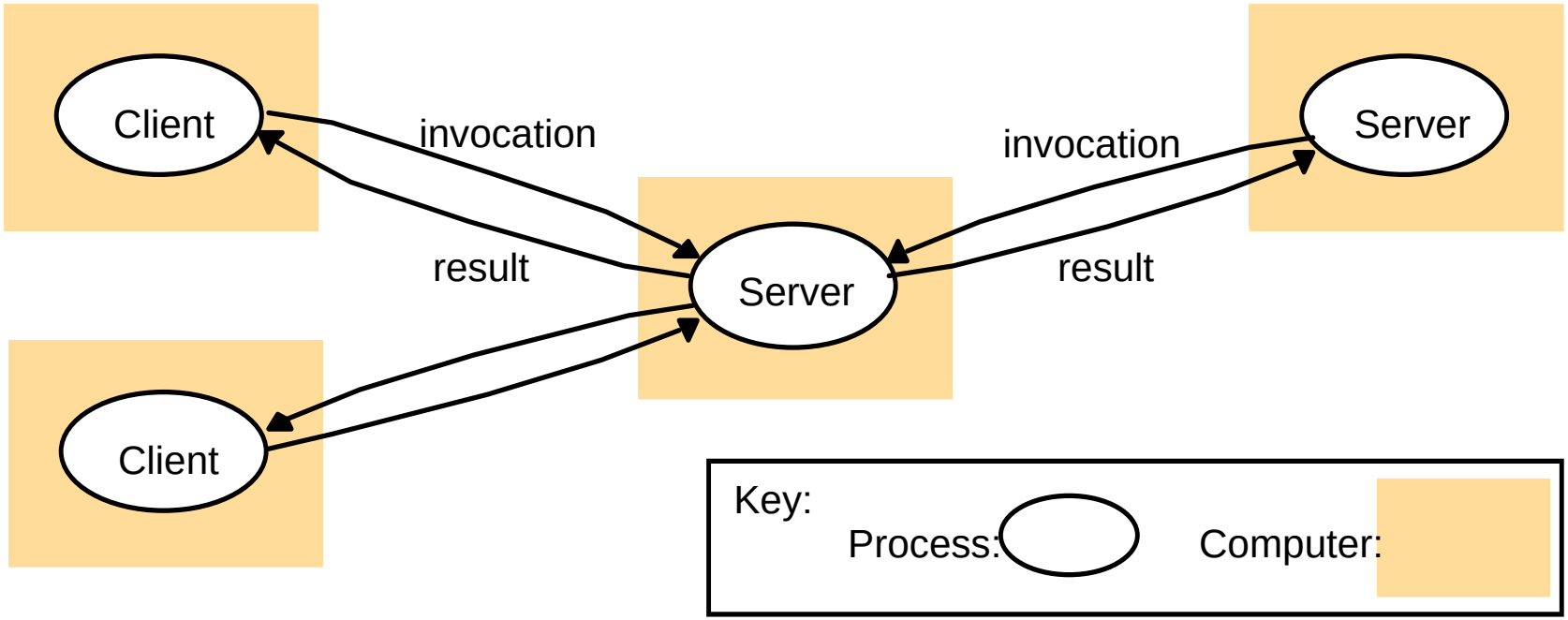
Service layers



Middleware

- Software layer (library of functions) that simplifies programming
  - Masks heterogeneity
  - Provides a convenient programming model
    - Objects/ processes
    - Communication primitives
    - Synchronization
    - Group and multicasting
    - Naming and Localization services
    - Event notification
  - Corba, JavaRMI, .NET Remoting, MPI, ISIS,…

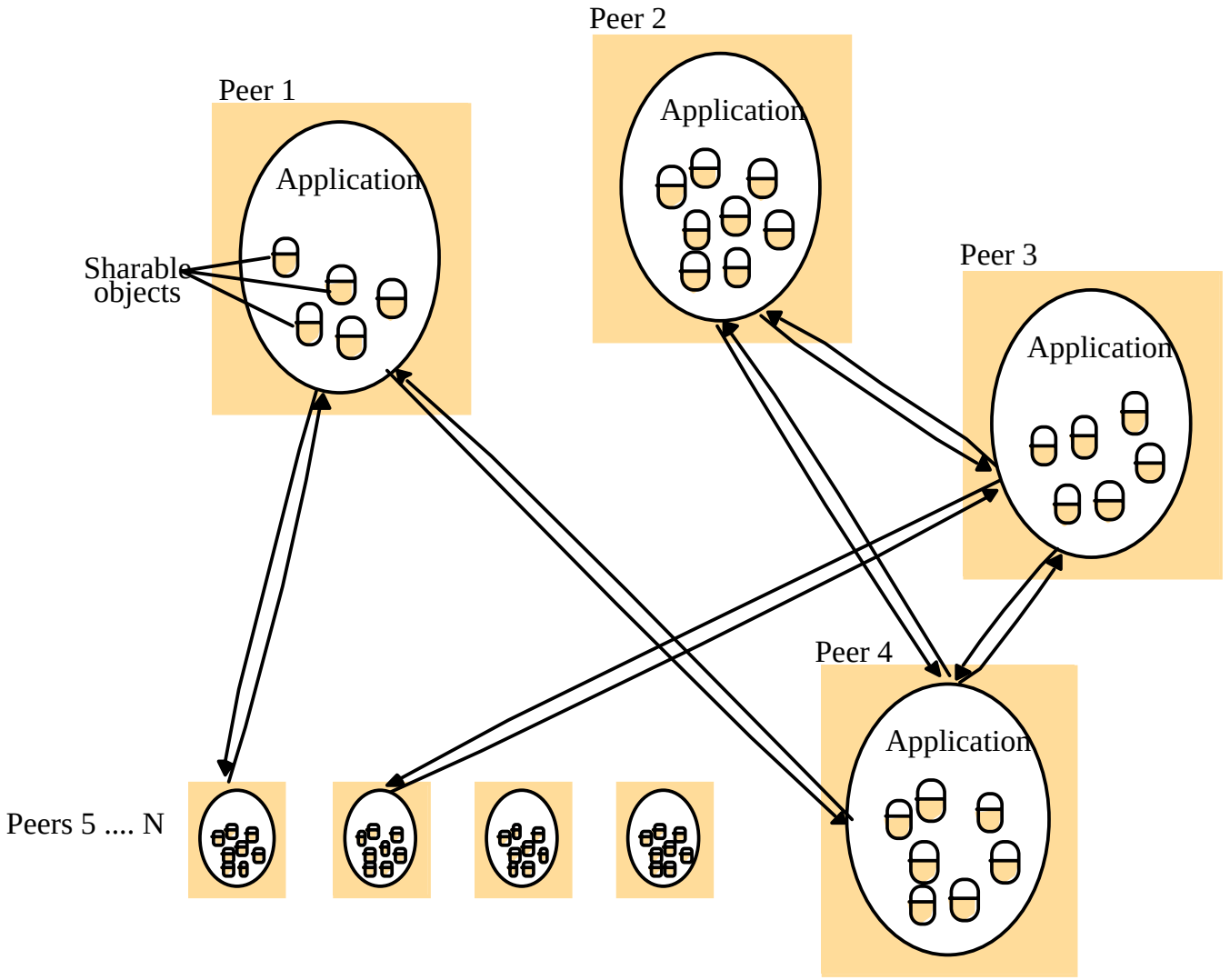
Clients / Server model



Variations:

- thin / thick / smart (dynamic) clients
- multiple server services
- multi-tier systems

A distributed application based on peer processes



Client Server vs Peer based

Client-Server

- Most widely used model
- Functional specialization
- Asymmetrical
- Tends to be Centralized
- Tends to scale poorly

Peer

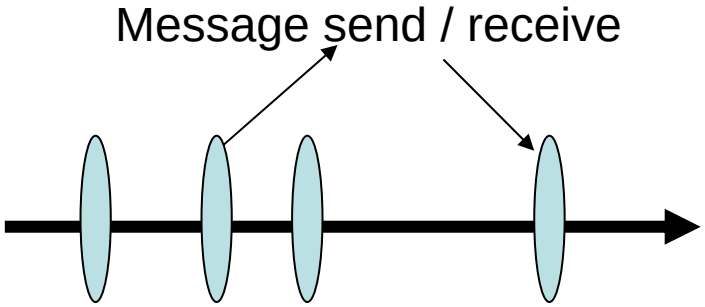
- Symmetrical, computers runs same algorithms / same responsibilities
- Truly Distributed
- Share / exploit resources at a large number of participants

Fundamental Models

Design and solutions depend on fundamental assumptions on

- Process Interaction
- Failures
- Security threats

Interaction Model



- Process:
  - executing program with private state
  - sending and receiving messages
- Distributed Algorithms:
  - *A definition of the steps to be taken by each of the processes of which the system is composed, especially the messages transmitted between them*
- *Communication Performance is a limiting characteristic*
  - *Latency, bandwidth, Jitter*
- *It is impossible to maintain a single notion of time*
  - *Computer clocks have drift*
  - *GPS: 1 micro Sec*
  - *Message Passing (eg.NTP) 100ms*

Communication speed

```
static void Main(string[] args)
 {
        uint i = 0; long y; int x = 500;
        const uint MAX_ITER = 1000000; // 1million
        DateTime tmStart; DateTime tmEnd;

    tmStart = DateTime.Now;
    for (i = 0; i <= MAX_ITER; i++)
        y = x * i;
    tmEnd = DateTime.Now;
    TimeSpan tmDiff = tmEnd - tmStart;
    Console.WriteLine(MAX_ITER + "iterations took "+
                    (tmDiff.TotalMilliseconds ) + "ms");
}
```

What is the total execution time of this program?

Communication speed

X is located on a remote host:

```
static void Main(string[] args)
 {
        uint i = 0; long y;
        const uint MAX_ITER = 1000000; // 1million
        DateTime tmStart; DateTime tmEnd;

    tmStart = DateTime.Now;
    for (i = 0; i <= MAX_ITER; i++)
        y = READ_REMOTE(x) * i;
    tmEnd = DateTime.Now;
    TimeSpan tmDiff = tmEnd - tmStart;
    Console.WriteLine(MAX_ITER + "iterations took "+
                (tmDiff.TotalMilliseconds ) + "ms");
 }
```

Read_Remote

Roundtrip time:
Ping www.cs
Ping krak.dk
Ping google.com

Interaction model 1: Asynchronous systems

No known bounds for:

- The execution speed of a process
- Message delay on the network
- Clock drift

Interaction model 2: (Partly) Synchronous systems

- Known upper and lower bound for each process step
- Known upper bound for the time it task for a message to be received
- Known upper bound for clock drift

Failure Model

- The system might need to tolerate failures
  - processes
    - might stop / crash
    - degrade gracefully
    - exhibit Byzantine failures
  - may also be failures of
    - communication mechanisms

Omission and arbitrary failures

Class of failure	*Affects*	*Description*
Fail-stop	Process	Process halts and remains halted. Other processes may detect this state.
Crash	Process	Process halts and remains halted. Other processes may not be able to detect this state.
Omission	Channel	A message inserted in an outgoing message buffer never arrives at the other end's incoming message buffer.
Send-omission	Process	A process completes a *send* but the message is not put in its outgoing message buffer.
Receive-omission	Process	A message is put in a process's incoming message buffer, but that process does not receive it.
Arbitrary (Byzantine)	Process or channel	Process/channel exhibits arbitrary behaviour: it may send/transmit arbitrary messages at arbitrary times, commit omissions; a process may stop or take an incorrect step.

Timing failures

Class of Failure	*Affects*	*Description*
Clock	Process	Process's local clock exceeds the bounds on its rate of drift from real time.
Performance	Process	Process exceeds the bounds on the interval between two steps.
Performance	Channel	A message's transmission takes longer than the stated bound.

Security model

- Protection of objects
- Securing processes and their interaction
  - Goals
    - Secrecy, integrity, authentication, authorization,..
  - Attacks
    - man-in-the-middle, eaves-dropping, play-back, …

END