Distributed Computing

Lesson 2: Distributed Systems

Thomas Weise · 汤卫思

tweise@hfuu.edu.cn · http://www.it-weise.de

Hefei University, South Campus 2
Faculty of Computer Science and Technology
Institute of Applied Optimization
230601 Shushan District, Hefei, Anhui, China
Econ. & Tech. Devel. Zone, Jinxiu Dadao 99

合肥学院 南艳湖校区/南2区
计算机科学与技术系
应用优化研究所
中国 安徽省 合肥市 蜀山区 230601
经济技术开发区 锦绣大道99号

Outline

1. Distributed Systems
2. General Features
3. Pros and Cons
4. Design Principles

website

Introduction

- We will get an understanding of what distributed systems are

Introduction

- We will get an understanding of what distributed systems are
- We will get an understanding of opportunities and challenges in distributed systems

Introduction

- We will get an understanding of what distributed systems are
- We will get an understanding of opportunities and challenges in distributed systems
- We will learn some of the basic design principles for distributed systems

Distributed Systems

What is a distributed system?

What are distributed algorithms?

Distributed Systems

- Various slightly different definitions exist [1–5], but let us settle for the following:

Distributed Systems

- Various slightly different definitions exist [1–5], but let us settle for the following:
- A distributed system is a set of autonomous systems (nodes, computers) which are connected by a network and communicate via the exchange of messages.

Distributed Systems

- Various slightly different definitions exist [1–5], but let us settle for the following:
- A distributed system is a set of autonomous systems (nodes, computers) which are connected by a network and communicate via the exchange of messages.
- Distributed algorithms are algorithms which can be executed by multiple computers in a distributed system and cooperatively try to solve a given problem.

Programming of Distributed Systems

- In courses on computer programming, you have learned that it is impossible to avoid programming mistakes.

Programming of Distributed Systems

- In courses on computer programming, you have learned that it is impossible to avoid programming mistakes.
- You have learned how important it is to test and debug programs.

Programming of Distributed Systems

- In courses on computer programming, you have learned that it is impossible to avoid programming mistakes.
- You have learned how important it is to test and debug programs.
- This is also true for programming distributed applications.

Programming of Distributed Systems

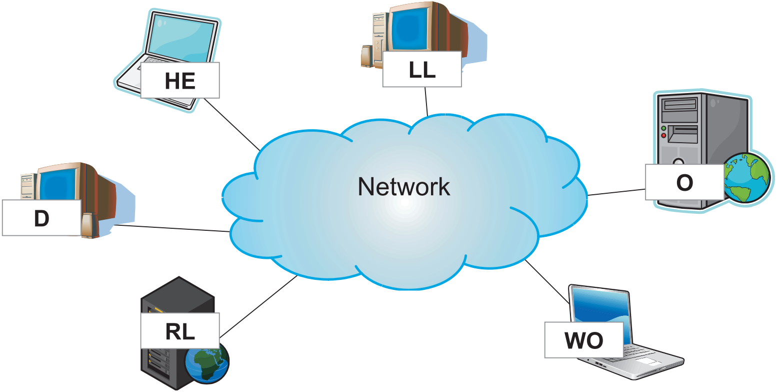
- In courses on computer programming, you have learned that it is impossible to avoid programming mistakes.
- You have learned how important it is to test and debug programs.
- This is also true for programming distributed applications.
- With the exception that there now are several entirely new possible sources of errors.

Programming of Distributed Systems

- In courses on computer programming, you have learned that it is impossible to avoid programming mistakes.
- You have learned how important it is to test and debug programs.
- This is also true for programming distributed applications.
- With the exception that there now are several entirely new possible sources of errors.
- Yes, this is going to get scary, be prepared.

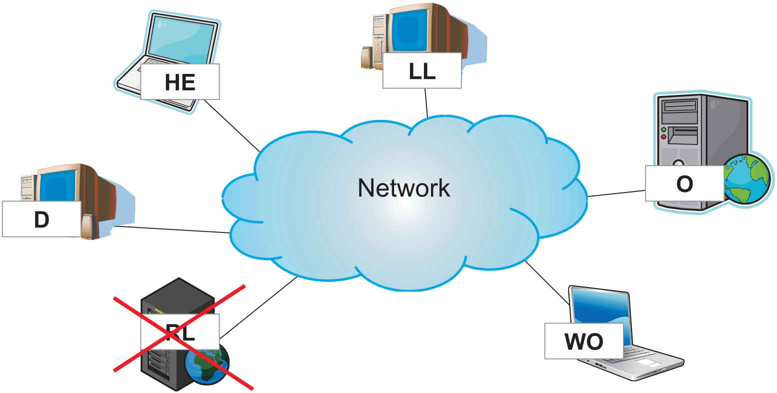
Network of Computers

- Multiple independent computers connected by communication network



Computers/Links Fail

- Computers and communication links may fail independently from each other (and without obvious reason)



Latency and Unreliability

- Information exchange only via message exchange

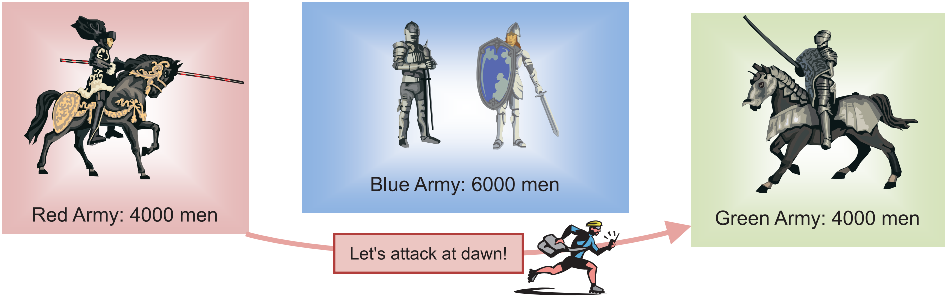
Red Army: 4000 men

Blue Army: 6000 men

Green Army: 4000 men

Latency and Unreliability

- Information exchange only via message exchange



Red Army: 4000 men

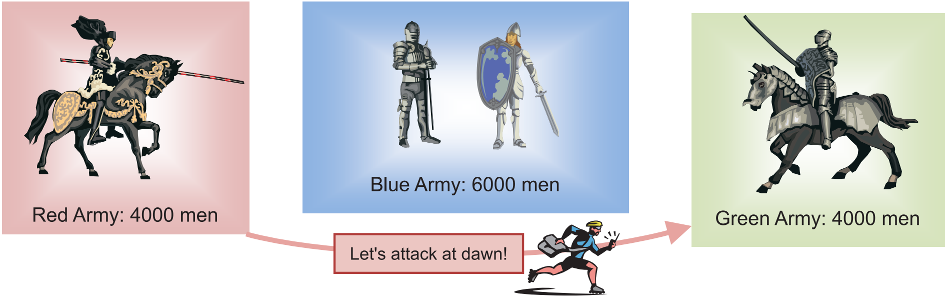
Blue Army: 6000 men

Green Army: 4000 men

Let's attack at dawn!

Latency and Unreliability

- Information exchange only via message exchange
- Message latencies not deterministic, messages may take over each other (definite physical limit: speed of light)



Red Army: 4000 men

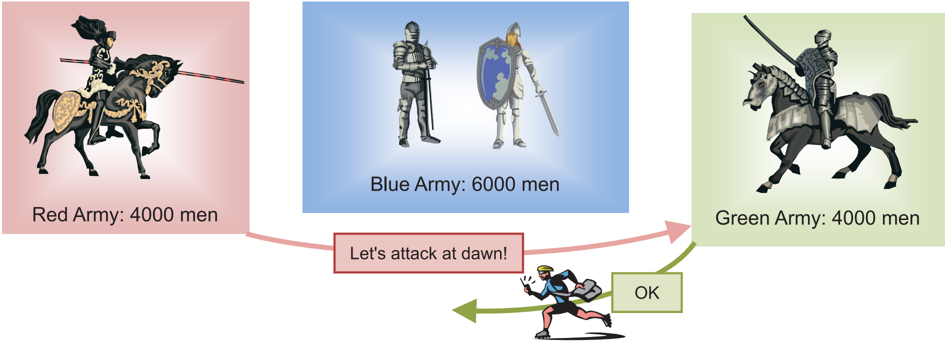
Blue Army: 6000 men

Green Army: 4000 men

Let's attack at dawn!

Latency and Unreliability

- Information exchange only via message exchange
- Message latencies not deterministic, messages may take over each other (definite physical limit: speed of light)
- Network may be unreliable, messages may get lost/modified



Red Army: 4000 men

Blue Army: 6000 men

Green Army: 4000 men

Let's attack at dawn!

OK

Uncertainty: No Global View

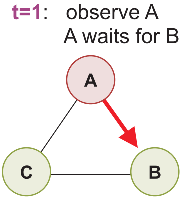
- Friedemann Mattern [3]: Uncertainty Principles of DS

Uncertainty: No Global View

- Friedemann Mattern [3]: Uncertainty Principles of DS:
  1. Multiple processes can never be observed simultaneously

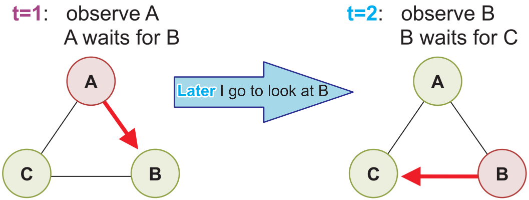
Uncertainty: No Global View

- Friedemann Mattern [3]: Uncertainty Principles of DS:
  1. Multiple processes can never be observed simultaneously



Uncertainty: No Global View

- Friedemann Mattern [3]: Uncertainty Principles of DS:
  1. Multiple processes can never be observed simultaneously



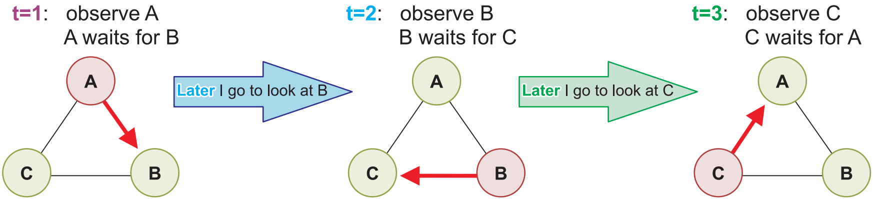
t=1: observe A
A waits for B

Later I go to look at B

t=2: observe B
B waits for C

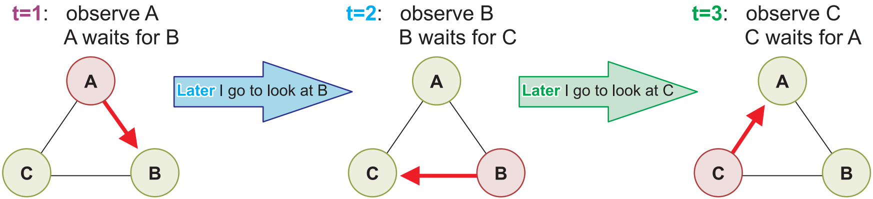
Uncertainty: No Global View

- Friedemann Mattern [3]: Uncertainty Principles of DS:
  1. Multiple processes can never be observed simultaneously



Uncertainty: No Global View

- Friedemann Mattern [3]: Uncertainty Principles of DS:
  1. Multiple processes can never be observed simultaneously
  2. It is difficult to make statements about the global system state



t=1: observe A
A waits for B

Later I go to look at B

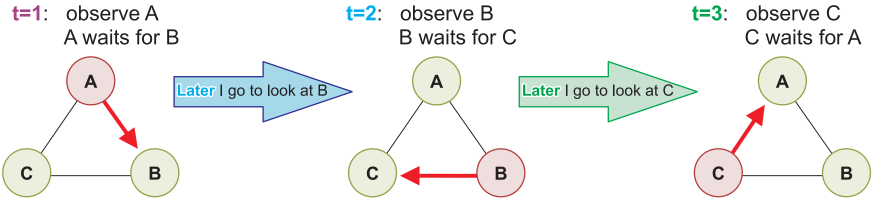
t=2: observe B
B waits for C

Later I go to look at C

t=3: observe C
C waits for A

Uncertainty: No Global View

- Friedemann Mattern [3]: Uncertainty Principles of DS:
  1. Multiple processes can never be observed simultaneously
  2. It is difficult to make statements about the global system state



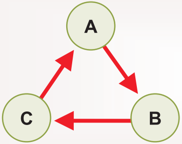
t=1: observe A
A waits for B

Later I go to look at B

t=2: observe B
B waits for C

Later I go to look at C

t=3: observe C
C waits for A

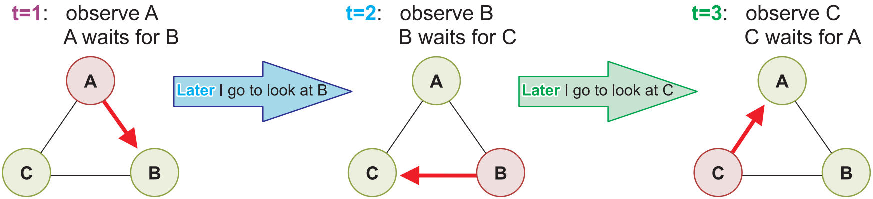


Phantom Deadlock

delays in observation lead to possibly wrong conclusion:
Deadlock
System may actually be OK.

Uncertainty: No Global View

- Friedemann Mattern [3]: Uncertainty Principles of DS:
  1. Multiple processes can never be observed simultaneously
  2. It is difficult to make statements about the global system state
  3. Nodes have no global information and act only based on local information → dangerous.



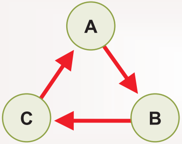
t=1: observe A
A waits for B

Later I go to look at B

t=2: observe B
B waits for C

Later I go to look at C

t=3: observe C
C waits for A



Phantom Deadlock

delays in observation lead to possibly wrong conclusion: Deadlock
System may actually be OK.

No Global Time

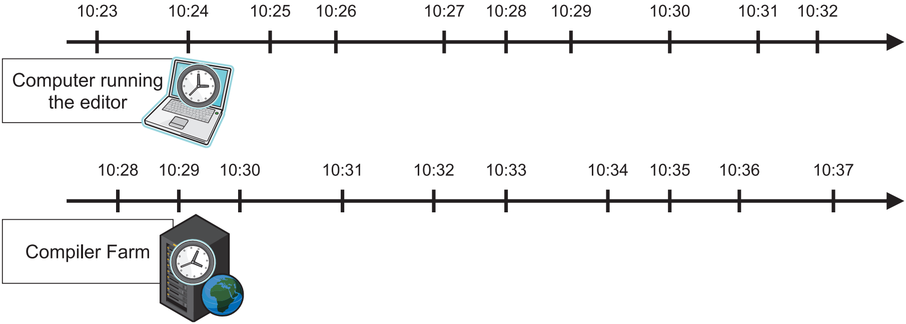
- From this, it follows that: We have no common, global time

No Global Time

- From this, it follows that: We have no common, global time
- Synchronization of clocks complicated because of message latency

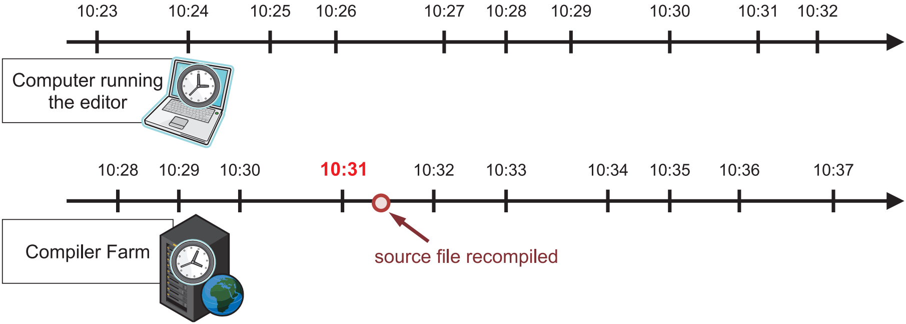
No Global Time

- From this, it follows that: We have no common, global time
- Synchronization of clocks complicated because of message latency



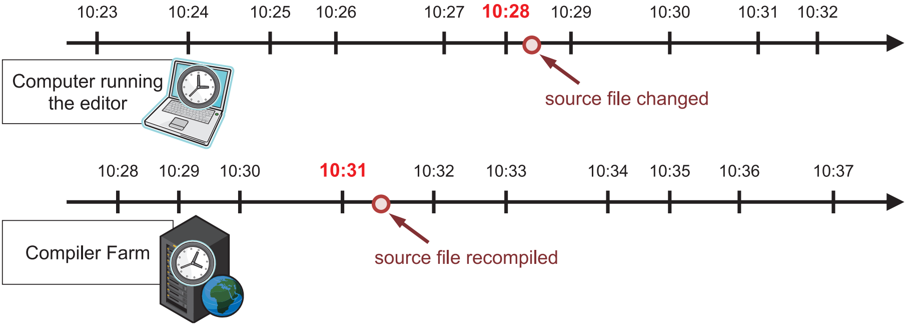
No Global Time

- From this, it follows that: We have no common, global time
- Synchronization of clocks complicated because of message latency



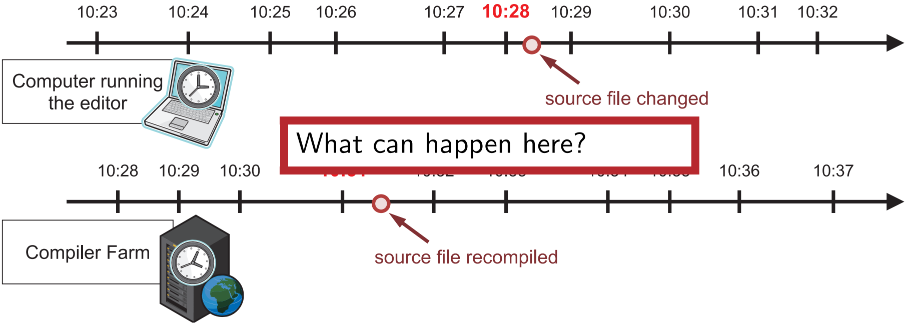
No Global Time

- From this, it follows that: We have no common, global time
- Synchronization of clocks complicated because of message latency



No Global Time

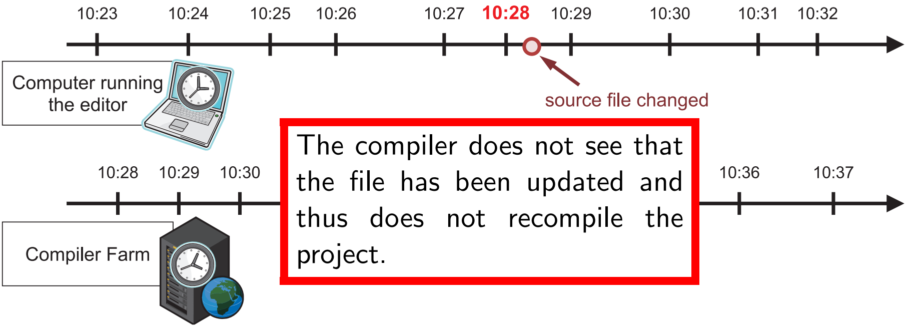
- From this, it follows that: We have no common, global time
- Synchronization of clocks complicated because of message latency



What can happen here?

No Global Time

- From this, it follows that: We have no common, global time
- Synchronization of clocks complicated because of message latency



The compiler does not see that the file has been updated and thus does not recompile the project.

General Features

- No globally shared memory

General Features

- No globally shared memory
    - no node knows the global state

General Features

- No globally shared memory
  - no node knows the global state
  - shared memory may be emulated, though that's not easy

General Features

- No globally shared memory
  - no node knows the global state
  - shared memory may be emulated, though that's not easy
- No accurate failure detection

General Features

- No globally shared memory
  - no node knows the global state
  - shared memory may be emulated, though that's not easy
- No accurate failure detection
  - in asynchronous distributed systems without upper bound for communication delay, it is impossible to distinguish failed from slow processes[6]

Pros and Cons

Ok, that was scary.

Pros and Cons

Ok, that was scary.

Surely, these are the only bad things that I have to consider when building a distributed application, right?

Pros and Cons

- Distributed systems are complex

Pros and Cons

- Distributed systems are complex:
    - heterogeneity of hardware/OSes/applications

Pros and Cons

- Distributed systems are complex:
  - heterogeneity of hardware/OSes/applications: documentation absolutely vital! [7]

Pros and Cons

- Distributed systems are complex:
  - heterogeneity of hardware/OSes/applications: documentation absolutely vital! [7]
  - consist of separate (autonomous) systems

Pros and Cons

- Distributed systems are complex:
  - heterogeneity of hardware/OSes/applications: documentation absolutely vital! [7]
  - consist of separate (autonomous) systems: true parallelism

Pros and Cons

- Distributed systems are complex:
    - heterogeneity of hardware/OSes/applications: documentation absolutely vital! [7]
    - consist of separate (autonomous) systems: true parallelism, indeterminism

Pros and Cons

- Distributed systems are complex:
  - heterogeneity of hardware/OSes/applications: documentation absolutely vital! [7]
  - consist of separate (autonomous) systems: true parallelism, indeterminism
- Programming them is more complex, too

Pros and Cons

- Distributed systems are complex:
    - heterogeneity of hardware/OSes/applications: documentation absolutely vital! [7]
    - consist of separate (autonomous) systems: true parallelism, indeterminism
- Programming them is more complex, too:
    - You have to deal with parallelism.

Pros and Cons

- Distributed systems are complex:
  - heterogeneity of hardware/OSes/applications: documentation absolutely vital! [7]
  - consist of separate (autonomous) systems: true parallelism, indeterminism
- Programming them is more complex, too:
  - You have to deal with parallelism.
  - You have to deal with communication.

Pros and Cons

- Distributed systems are complex:
  - heterogeneity of hardware/OSes/applications: documentation absolutely vital! [7]
  - consist of separate (autonomous) systems: true parallelism, indeterminism
- Programming them is more complex, too:
  - You have to deal with parallelism.
  - You have to deal with communication.
  - How to do testing and debugging?

Pros and Cons

- Distributed systems are complex:
  - heterogeneity of hardware/OSes/applications: documentation absolutely vital! [7]
  - consist of separate (autonomous) systems: true parallelism, indeterminism
- Programming them is more complex, too:
  - You have to deal with parallelism.
  - You have to deal with communication.
  - How to do testing and debugging?
  - Failures are often not reproducible.

Pros and Cons

- Security in distributed systems is complex

Pros and Cons

- Security in distributed systems is complex:
    - confidentiality

Pros and Cons

- Security in distributed systems is complex:
  - confidentiality
  - more attack points, DDOS attacks, intrusion detection, . . .

Pros and Cons

- Security in distributed systems is complex:
  - confidentiality
  - more attack points, DDOS attacks, intrusion detection, . . .
  - even professional organizations whose business it is to be secure. . . make terrible mistakes! [8–10]

Pros and Cons

- Security in distributed systems is complex:
  - confidentiality
  - more attack points, DDOS attacks, intrusion detection, . . .
  - even professional organizations whose business it is to be secure. . . make terrible mistakes! [8–10]
- Systems may span over country borders, different administrative domains

Pros and Cons

- Security in distributed systems is complex:
  - confidentiality
  - more attack points, DDOS attacks, intrusion detection, . . .
  - even professional organizations whose business it is to be secure. . . make terrible mistakes! [8–10]
- Systems may span over country borders, different administrative domains:
  - data security

Pros and Cons

- Security in distributed systems is complex:
  - confidentiality
  - more attack points, DDOS attacks, intrusion detection, . . .
  - even professional organizations whose business it is to be secure. . . make terrible mistakes! [8–10]
- Systems may span over country borders, different administrative domains:
  - data security
  - legislative security

Pros and Cons

- Security in distributed systems is complex:
  - confidentiality
  - more attack points, DDOS attacks, intrusion detection, . . .
  - even professional organizations whose business it is to be secure. . . make terrible mistakes! [8–10]
- Systems may span over country borders, different administrative domains:
  - data security
  - legislative security
  - central administration impossible

Pros and Cons

- Security in distributed systems is complex:
  - confidentiality
  - more attack points, DDOS attacks, intrusion detection, ...
  - even professional organizations whose business it is to be secure... make terrible mistakes! [8–10]
- Systems may span over country borders, different administrative domains:
  - data security
  - legislative security
  - central administration impossible
- Distributed systems are expensive

Pros and Cons

- Security in distributed systems is complex:
    - confidentiality
    - more attack points, DDOS attacks, intrusion detection, . . .
    - even professional organizations whose business it is to be secure. . . make terrible mistakes! [8–10]
- Systems may span over country borders, different administrative domains:
    - data security
    - legislative security
    - central administration impossible
- Distributed systems are expensive:
    - network infrastructure

Pros and Cons

- Security in distributed systems is complex:
  - confidentiality
  - more attack points, DDOS attacks, intrusion detection, . . .
  - even professional organizations whose business it is to be secure. . . make terrible mistakes! [8–10]
- Systems may span over country borders, different administrative domains:
  - data security
  - legislative security
  - central administration impossible
- Distributed systems are expensive:
  - network infrastructure
  - energy consumption

Leslie Lamport's Statement

Ok, let's see what Leslie Lamport — one of the *big guys* in distributed systems – has to say.

Leslie Lamport's Statement

Ok, let's see what Leslie Lamport — one of the *big guys* in distributed systems – has to say. He has done lots of useful research [11–13].

Leslie Lamport's Statement

Ok, let's see what Leslie Lamport — one of the *big guys* in distributed systems – has to say. He has done lots of useful research [11–13] (and also invented LaTeX [15]).

Leslie Lamport's Statement

Ok, let's see what Leslie Lamport — one of the *big guys* in distributed systems – has to say. He has done lots of useful research [11–13] (and also invented $\LaTeX$ [15], which sits on top of the $\TeX$ [16] system created by Knuth).

Leslie Lamport's Statement

Ok, let's see what Leslie Lamport — one of the *big guys* in distributed systems – has to say. He has done lots of useful research [11–13] (and also invented LaTeX [15], which sits on top of the TeX [16] system created by Knuth, who is the guy who brought you *The Art of Computer Programming* [17–19] and *Concrete Mathematics* [20]).

Leslie Lamport's Statement

Ok, let's see what Leslie Lamport — one of the *big guys* in distributed systems – has to say. He has done lots of useful research [11–13] (and also invented LaTeX [15], which sits on top of the TeX [16] system created by Knuth, who is the guy who brought you *The Art of Computer Programming* [17–19] and *Concrete Mathematics* [20]).

. . . Anyway, what does he have to say? [14]

Leslie Lamport's Statement

Received: by jumbo.dec.com (5.54.3/4.7.34) id AA09105; Thu, 28 May 87 12:23:29 PDT
Date: Thu, 28 May 87 12:23:29 PDT
From: lamport (Leslie Lamport)[14] To: src-t
Message-Id: <8705281923.AA09105@jumbo.dec.com>
Subject: distribution

There has been considerable debate over the years about what constitutes a distributed system. It would appear that the following definition has been adopted at SRC:

> A distributed system is one in which the failure of a computer you didn't even know existed can render your own computer unusable.

The current electrical problem in the machine room is not the culprit--it just highlights a situation that has been getting progressively worse. It seems that each new version of the nub makes my FF more dependent upon programs that run elsewhere.

Having to wait a few seconds for a program to be swapped in is a lot less annoying than having to wait an hour or two for someone to reboot the servers. I therefore propose a development project to make our system more robust. I am not proposing any particular approach (enabling stand-alone operation is just one possibility).

I will begin the effort by volunteering to gather some data on the problem. If you know of any instance of user's FF becoming inoperative through no fault of its own, please send me a message indicating the user, the time, and the cause (if known).
Leslie

Pros and Cons

- Leslie Lamport:
  “A distributed system is one in which the failure of a computer you didn’t even know existed can render your own computer unusable.” [14]

Pros and Cons

- Leslie Lamport:
  "A distributed system is one in which the failure of a computer you didn't even know existed can render your own computer unusable." [14]
- Veríssimo and Rodrigues:
  "If you do not need a distributed system, do not distribute." [5]

Pros and Cons

- Leslie Lamport:
  "A distributed system is one in which the failure of a computer you didn't even know existed can render your own computer unusable." [14]
- Veríssimo and Rodrigues:
  "If you do not need a distributed system, do not distribute." [5]

So why do we even bother?

Pros and Cons

- Leslie Lamport:
  “A distributed system is one in which the failure of a computer you didn’t even know existed can render your own computer unusable.” [14]
- Veríssimo and Rodrigues:
  “If you do not need a distributed system, do not distribute.” [5]

So why do we even bother?

Are there advantages??

Pros and Cons

- Leslie Lamport:
  "A distributed system is one in which the failure of a computer you didn't even know existed can render your own computer unusable." [14]
- Veríssimo and Rodrigues:
  "If you do not need a distributed system, do not distribute." [5]

So why do we even bother?

Are there advantages??

Why am I taking this course?

Pros and Cons

- Some services need to carried out at a specific location

Pros and Cons

- Some services need to carried out at a specific location, e.g.,
  - Online ordering/manufacturing systems

Pros and Cons

- Some services need to carried out at a specific location, e.g.,
    - Online ordering/manufacturing systems
    - Control of hard-to-reach systems (chemical plant, robotic moon mission)

Pros and Cons

- Some services need to carried out at a specific location, e.g.,
    - Online ordering/manufacturing systems
    - Control of hard-to-reach systems (chemical plant, robotic moon mission)
    - Drones in modern warfare

Pros and Cons

- Some services need to carried out at a specific location
- Some tasks are distributed by nature

Pros and Cons

- Some services need to carried out at a specific location
- Some tasks are distributed by nature, e.g.,
    - Team of robots in robotic soccer or robots in a warehous

Pros and Cons

- Some services need to carried out at a specific location
- Some tasks are distributed by nature, e.g.,
  - Team of robots in robotic soccer or robots in a warehous
  - Storage of photos/music/videos and access from multiple different devices

Pros and Cons

- Some services need to carried out at a specific location
- Some tasks are distributed by nature, e.g.,
    - Team of robots in robotic soccer or robots in a warehous
    - Storage of photos/music/videos and access from multiple different devices
    - Access of websites/information created at different locations

Pros and Cons

- Some services need to carried out at a specific location
- Some tasks are distributed by nature
- Modularity and flexibility

Pros and Cons

- Some services need to carried out at a specific location
- Some tasks are distributed by nature
- Modularity and flexibility, e.g.,
    - Services in a SOA are single modules which can be maintained and replaced separately

Pros and Cons

- Some services need to carried out at a specific location
- Some tasks are distributed by nature
- Modularity and flexibility, e.g.,
  - Services in a SOA are single modules which can be maintained and replaced separately
  - Computers can easily be added to clusters or grids

Pros and Cons

- Some services need to carried out at a specific location
- Some tasks are distributed by nature
- Modularity and flexibility, e.g.,
    - Services in a SOA are single modules which can be maintained and replaced separately
    - Computers can easily be added to clusters or grids
    - Elastic cloud services can be replicated to more computers/virtual machines added to applications when more computational power becomes necessary

Pros and Cons

- Some services need to carried out at a specific location
- Some tasks are distributed by nature
- Modularity and flexibility
- Incremental growth / scalability

Pros and Cons

- Some services need to carried out at a specific location
- Some tasks are distributed by nature
- Modularity and flexibility
- Incremental growth / scalability, e.g.,
    - New components can be added to a SOA

Pros and Cons

- Some services need to carried out at a specific location
- Some tasks are distributed by nature
- Modularity and flexibility
- Incremental growth / scalability, e.g.,
  - New components can be added to a SOA
  - Heterogeneous systems: legacy applications + new applications

Pros and Cons

- Some services need to carried out at a specific location
- Some tasks are distributed by nature
- Modularity and flexibility
- Incremental growth / scalability, e.g.,
  - New components can be added to a SOA
  - Heterogeneous systems: legacy applications + new applications
  - Purchase of more computing power in a cloud

Pros and Cons

- Some services need to carried out at a specific location
- Some tasks are distributed by nature
- Modularity and flexibility
- Incremental growth / scalability, e.g.,
    - New components can be added to a SOA
    - Heterogeneous systems: legacy applications + new applications
    - Purchase of more computing power in a cloud
    - Adding of sensors to an existing manufacturing surveillance plant system

Pros and Cons

- Some services need to carried out at a specific location
- Some tasks are distributed by nature
- Modularity and flexibility
- Incremental growth / scalability
- Resource sharing

Pros and Cons

- Some services need to carried out at a specific location
- Some tasks are distributed by nature
- Modularity and flexibility
- Incremental growth / scalability
- Resource sharing, e.g.,
    - Cloud and cluster computing

Pros and Cons

- Some services need to carried out at a specific location
- Some tasks are distributed by nature
- Modularity and flexibility
- Incremental growth / scalability
- Resource sharing, e.g.,
    - Cloud and cluster computing
    - Seti@home [21], BOINC [22]

Pros and Cons

- Some services need to carried out at a specific location
- Some tasks are distributed by nature
- Modularity and flexibility
- Incremental growth / scalability
- Resource sharing
- Load sharing / balancing / grid computing

Pros and Cons

- Some services need to carried out at a specific location
- Some tasks are distributed by nature
- Modularity and flexibility
- Incremental growth / scalability
- Resource sharing
- Load sharing / balancing / grid computing, e.g.,
  - Replication of web servers

Pros and Cons

- Some services need to carried out at a specific location
- Some tasks are distributed by nature
- Modularity and flexibility
- Incremental growth / scalability
- Resource sharing
- Load sharing / balancing / grid computing, e.g.,
    - Replication of web servers
    - Access node of cluster decides where the job should be done

Pros and Cons

- Some services need to carried out at a specific location
- Some tasks are distributed by nature
- Modularity and flexibility
- Incremental growth / scalability
- Resource sharing
- Load sharing / balancing / grid computing
- High availability, mobile accessibility

Pros and Cons

- Some services need to carried out at a specific location
- Some tasks are distributed by nature
- Modularity and flexibility
- Incremental growth / scalability
- Resource sharing
- Load sharing / balancing / grid computing
- High availability, mobile accessibility, e.g.,
    - Replication

Pros and Cons

- Some services need to carried out at a specific location
- Some tasks are distributed by nature
- Modularity and flexibility
- Incremental growth / scalability
- Resource sharing
- Load sharing / balancing / grid computing
- High availability, mobile accessibility, e.g.,
    - Replication
    - Maintenance centralized

Pros and Cons

- Some services need to carried out at a specific location
- Some tasks are distributed by nature
- Modularity and flexibility
- Incremental growth / scalability
- Resource sharing
- Load sharing / balancing / grid computing
- High availability, mobile accessibility, e.g.,
    - Replication
    - Maintenance centralized
    - Internet accessible from hand helds

Pros and Cons

- Some services need to carried out at a specific location
- Some tasks are distributed by nature
- Modularity and flexibility
- Incremental growth / scalability
- Resource sharing
- Load sharing / balancing / grid computing
- High availability, mobile accessibility, e.g.,
  - Replication
  - Maintenance centralized
  - Internet accessible from hand helds
  - Price checks from your mobile phone

Pros and Cons

- Some services need to carried out at a specific location
- Some tasks are distributed by nature
- Modularity and flexibility
- Incremental growth / scalability
- Resource sharing
- Load sharing / balancing / grid computing
- High availability, mobile accessibility
- Lower costs ($/MHz, $/(IO/s), . . .)

Pros and Cons

- Some services need to carried out at a specific location
- Some tasks are distributed by nature
- Modularity and flexibility
- Incremental growth / scalability
- Resource sharing
- Load sharing / balancing / grid computing
- High availability, mobile accessibility
- Lower costs (\$/MHz, \$/(IO/s), . . .), e.g.,
    - Large-scale cluster with central cooling and energy supply

Pros and Cons

- Some services need to carried out at a specific location
- Some tasks are distributed by nature
- Modularity and flexibility
- Incremental growth / scalability
- Resource sharing
- Load sharing / balancing / grid computing
- High availability, mobile accessibility
- Lower costs (\$/MHz, \$/(IO/s), . . .), e.g.,
    - Large-scale cluster with central cooling and energy supply
    - Shared infrastructure

Pros and Cons

- Some services need to carried out at a specific location
- Some tasks are distributed by nature
- Modularity and flexibility
- Incremental growth / scalability
- Resource sharing
- Load sharing / balancing / grid computing
- High availability, mobile accessibility
- Lower costs (\$/MHz, \$/(IO/s), . . .), e.g.,
    - Large-scale cluster with central cooling and energy supply
    - Shared infrastructure
    - Computers turned on/off on demand

Pros and Cons

- Some services need to carried out at a specific location
- Some tasks are distributed by nature
- Modularity and flexibility
- Incremental growth / scalability
- Resource sharing
- Load sharing / balancing / grid computing
- High availability, mobile accessibility
- Lower costs ($/MHz, $/(IO/s), . . .)
- Outsourcing / time zone advantages / cloud computing

Pros and Cons

- Some services need to carried out at a specific location
- Some tasks are distributed by nature
- Modularity and flexibility
- Incremental growth / scalability
- Resource sharing
- Load sharing / balancing / grid computing
- High availability, mobile accessibility
- Lower costs (\$/MHz, \$/(IO/s), . . .)
- Outsourcing / time zone advantages / cloud computing, e.g.,
    - US/Europe outsource many IT jobs to India, because it is cheap

Pros and Cons

- Some services need to carried out at a specific location
- Some tasks are distributed by nature
- Modularity and flexibility
- Incremental growth / scalability
- Resource sharing
- Load sharing / balancing / grid computing
- High availability, mobile accessibility
- Lower costs ($/MHz, $/(IO/s), . . .)
- Outsourcing / time zone advantages / cloud computing, e.g.,
    - US/Europe outsource many IT jobs to India, because it is cheap
    - US + China could work in shifts, everyone works at day time, but work is done 24h

Pros and Cons

- Some services need to carried out at a specific location
- Some tasks are distributed by nature
- Modularity and flexibility
- Incremental growth / scalability
- Resource sharing
- Load sharing / balancing / grid computing
- High availability, mobile accessibility
- Lower costs ($/MHz, $/(IO/s), ...)
- Outsourcing / time zone advantages / cloud computing, e.g.,
  - US/Europe outsource many IT jobs to India, because it is cheap
  - US + China could work in shifts, everyone works at day time, but work is done 24h
  - Infrastructure in US is rarely used at night → which is daytime in India

Pros and Cons

- Some services need to carried out at a specific location
- Some tasks are distributed by nature
- Modularity and flexibility
- Incremental growth / scalability
- Resource sharing
- Load sharing / balancing / grid computing
- High availability, mobile accessibility
- Lower costs (\$/MHz, \$/(IO/s), . . .)
- Outsourcing / time zone advantages / cloud computing
- Fault tolerance / safety / replication / component replacement

Pros and Cons

- Some services need to carried out at a specific location
- Some tasks are distributed by nature
- Modularity and flexibility
- Incremental growth / scalability
- Resource sharing
- Load sharing / balancing / grid computing
- High availability, mobile accessibility
- Lower costs ($/MHz, $/(IO/s), . . .)
- Outsourcing / time zone advantages / cloud computing
- Fault tolerance / safety / replication / component replacement, e.g.,
    - Large companies replicate their databases (e.g., with engineering designs) at different geographical locations

Pros and Cons

- Some services need to carried out at a specific location
- Some tasks are distributed by nature
- Modularity and flexibility
- Incremental growth / scalability
- Resource sharing
- Load sharing / balancing / grid computing
- High availability, mobile accessibility
- Lower costs (\$/MHz, \$/(IO/s), . . .)
- Outsourcing / time zone advantages / cloud computing
- Fault tolerance / safety / replication / component replacement, e.g.,
    - Large companies replicate their databases (e.g., with engineering designs) at different geographical locations
    - Critical computing infrastructure should be replicated

Pros and Cons

- Some services need to carried out at a specific location
- Some tasks are distributed by nature
- Modularity and flexibility
- Incremental growth / scalability
- Resource sharing
- Load sharing / balancing / grid computing
- High availability, mobile accessibility
- Lower costs ($/MHz, $/(IO/s), . . .)
- Outsourcing / time zone advantages / cloud computing
- Fault tolerance / safety / replication / component replacement
- Decentralization

Pros and Cons

- Some services need to carried out at a specific location
- Some tasks are distributed by nature
- Modularity and flexibility
- Incremental growth / scalability
- Resource sharing
- Load sharing / balancing / grid computing
- High availability, mobile accessibility
- Lower costs (\$/MHz, \$/(IO/s), . . .)
- Outsourcing / time zone advantages / cloud computing
- Fault tolerance / safety / replication / component replacement
- Decentralization, e.g.,
    - Content “closer” to destination (see Content Delivery Networks)

Pros and Cons

- Some services need to carried out at a specific location
- Some tasks are distributed by nature
- Modularity and flexibility
- Incremental growth / scalability
- Resource sharing
- Load sharing / balancing / grid computing
- High availability, mobile accessibility
- Lower costs (\$/MHz, \$/(IO/s), . . .)
- Outsourcing / time zone advantages / cloud computing
- Fault tolerance / safety / replication / component replacement
- Decentralization, e.g.,
    - Content “closer” to destination (see Content Delivery Networks)
    - Services can be carried out in different locations: less computational load on central hardware

Pros and Cons

- Some services need to carried out at a specific location
- Some tasks are distributed by nature
- Modularity and flexibility
- Incremental growth / scalability
- Resource sharing
- Load sharing / balancing / grid computing
- High availability, mobile accessibility
- Lower costs (\$/MHz, \$/(IO/s), . . .)
- Outsourcing / time zone advantages / cloud computing
- Fault tolerance / safety / replication / component replacement
- Decentralization
- Distribution over different jurisdictional and/or political areas

Design Principles

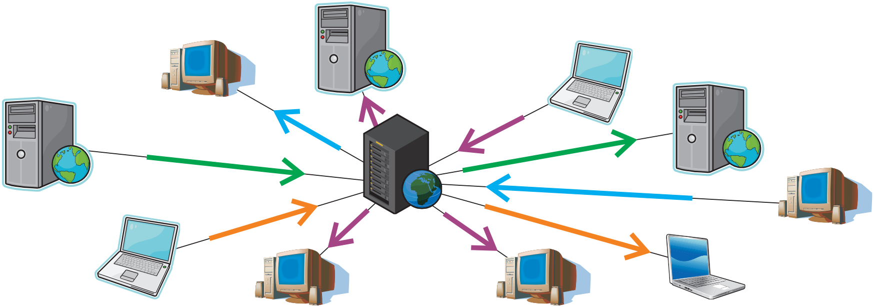
- Only distribute if there is a striking benefit, otherwise use centralized solutions.

Design Principles

- Only distribute if there is a striking benefit, otherwise use centralized solutions.
- What else?

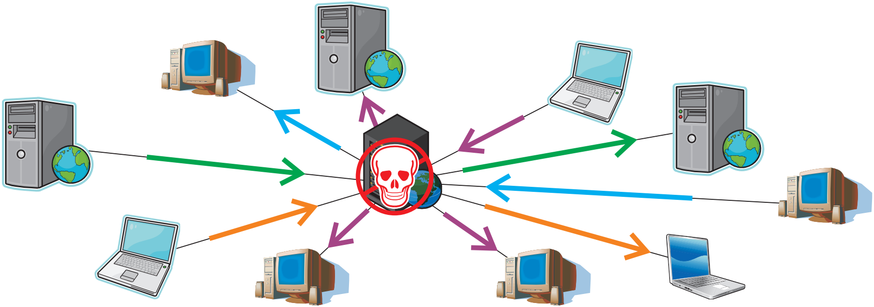
Single Point of Failure?

- One central server dealing with all requests and forwarding all messages between the nodes. Is this good or bad?



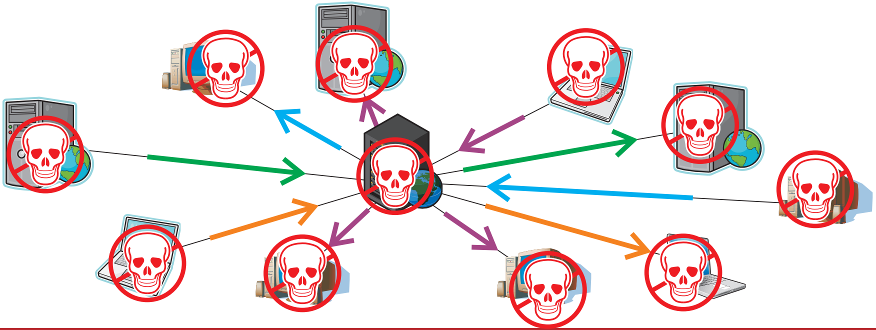
Single Point of Failure?

- Bad! You should avoid central/single points of failure!



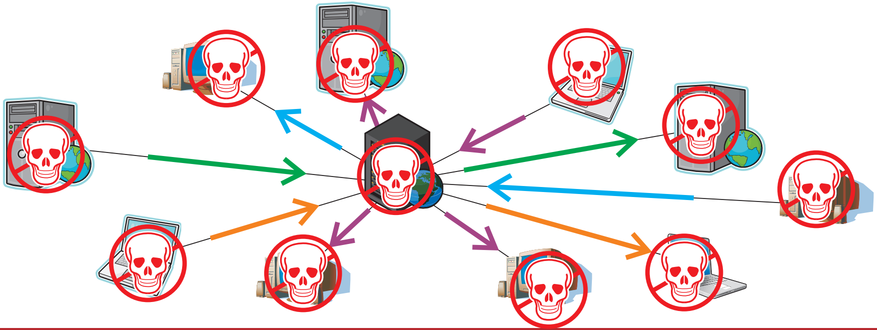
Single Point of Failure?

- Bad! You should avoid central/single points of failure!



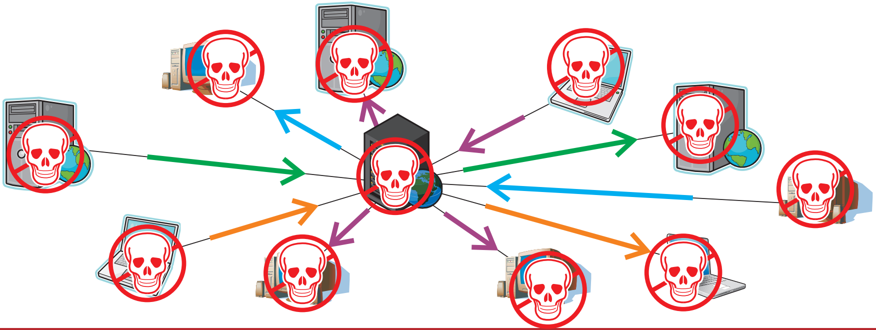
Single Point of Failure?

- Bad! You should avoid central/single points of failure!
  - regardless which node or connection fails, the system should remain intact



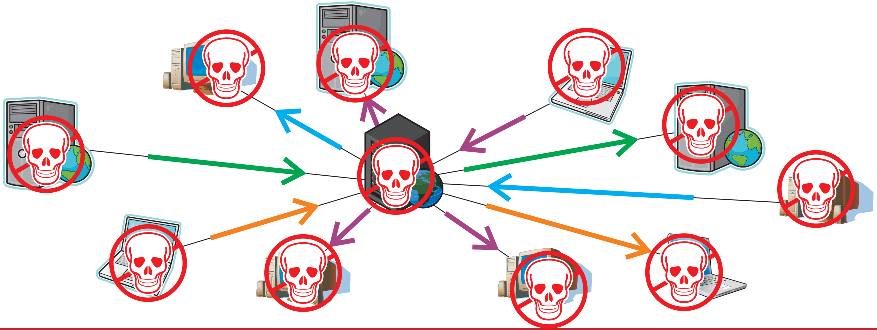
Single Point of Failure?

- Bad! You should avoid central/single points of failure!
  - regardless which node or connection fails, the system should remain intact
  - node failure/churn should lead to gentle performance degeneration rather than total failure



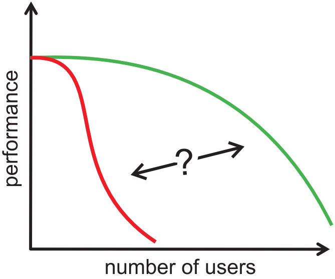
Single Point of Failure?

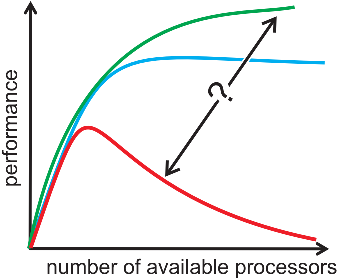
- Bad! You should avoid central/single points of failure!
  - regardless which node or connection fails, the system should remain intact
  - node failure/churn should lead to gentle performance degeneration rather than total failure
  - bottlenecks should be avoided



Scalability

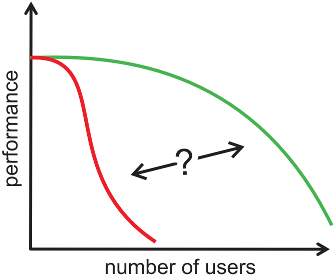
- How should a system behave when the numbers of users or processors increase?

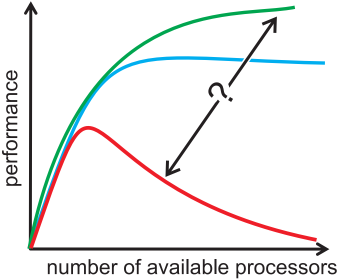




Scalability

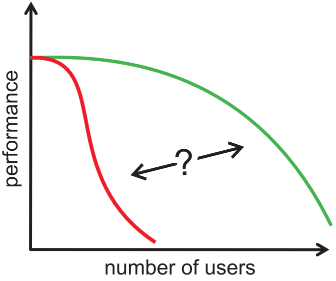
- Scalability

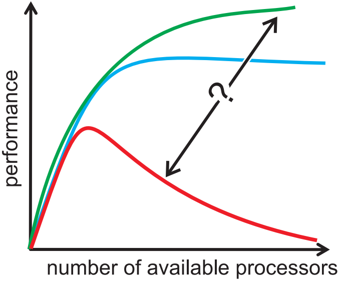




Scalability

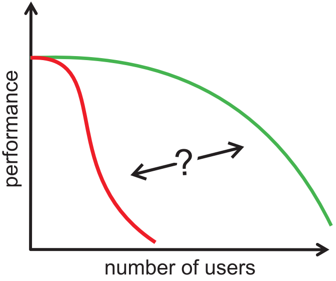
- Scalability
  - growth of number of tasks / users / work load / available nodes should be anticipated

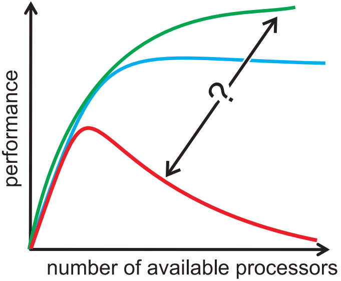




Scalability

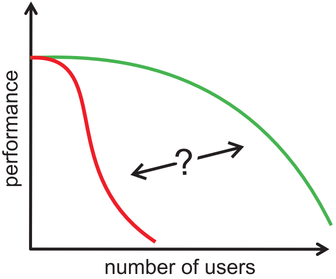
- Scalability
  - growth of number of tasks / users / work load / available nodes should be anticipated
  - system performance should rise in case of more computing power

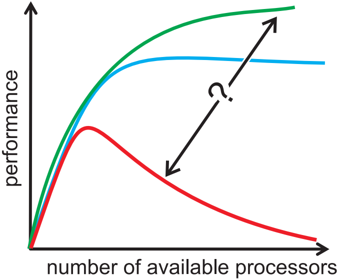




Scalability

- Scalability
  - growth of number of tasks / users / work load / available nodes should be anticipated
  - system performance should rise in case of more computing power
  - system performance should degenerate gently in case of higher work load





Security

- Security: Information hiding / need to know

Security

- Security: Information hiding / need to know
  - only the information absolutely need to solve a task is handed to the notes

Security

- Security: Information hiding / need to know
  - only the information absolutely need to solve a task is handed to the notes
  - security cannot be “integrated later”, security is no side dish

Security

- Security: Information hiding / need to know
  - only the information absolutely need to solve a task is handed to the notes
  - security cannot be "integrated later", security is no side dish
  - security by obfuscation does not work, use well-known algorithms and methods instead

Security

- Security: Information hiding / need to know
  - only the information absolutely need to solve a task is handed to the notes
  - security cannot be "integrated later", security is no side dish
  - security by obfuscation does not work, use well-known algorithms and methods instead

- Maybe you should also take an information systems security course as well...

Transparency

- Transparency of an aspect = the aspect is not visible to the app/user

Transparency

- Transparency of an aspect = the aspect is not visible to the app/user
  - hide complexity (for instance: protocol stack, middleware)

Transparency

- Transparency of an aspect = the aspect is not visible to the app/user
  - hide complexity (for instance: protocol stack, middleware)
  - many applications of transparency in distributed systems (distribution, location, machine, fault, replication, migration, . . .)

Transparency

- Transparency of an aspect = the aspect is not visible to the app/user
  - hide complexity (for instance: protocol stack, middleware)
  - many applications of transparency in distributed systems (distribution, location, machine, fault, replication, migration, . . .)
  - sometimes not appropriate (system management, context aware systems, adaptive systems, . . .)

Summary

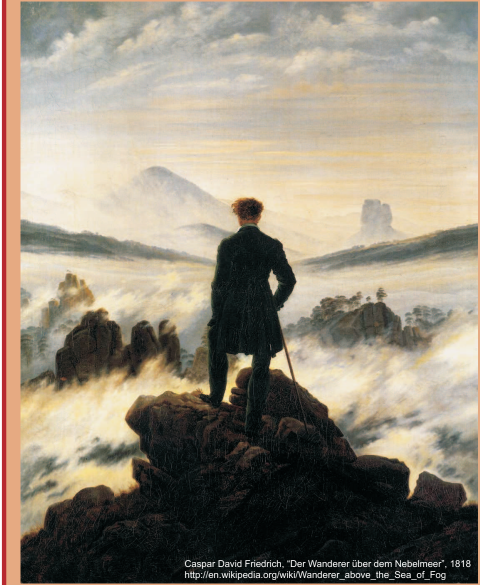
- Distributed System = autonomous nodes communicating via messages
- Several advantages and drawbacks: distributed computing only if necessary
- Several design principles to consider
- Scalability is always limited
- Message complexity should be low

谢谢

Thank you

Thomas Weise [汤卫思]
tweise@hfuu.edu.cn
http://www.it-weise.de

Hefei University, South Campus 2
Institute of Applied Optimization
Shushan District, Hefei, Anhui,
China



Caspar David Friedrich, “Der Wanderer über dem Nebelmeer”, 1818
http://en.wikipedia.org/wiki/Wanderer_above_the_Sea_of_Fog

Bibliography

Bibliography I

1. Andrew Stuart Tanenbaum. *Distributed Operating Systems*. Prentice-Hall International Editions. Upper Saddle River, NJ, USA: Prentice Hall International Inc., 1995. ISBN 0-13-143934-0, 0-13-219908-4, 978-0-13-143934-4, and 978-0-13-219908-7. URL http://books.google.de/books?id=MX8ZAQAAIAAJ.
2. George F. Coulouris, Jean Dollimore, and Tim Kindberg. *Distributed Systems: Concepts and Design*. Upper Saddle River, NJ, USA: Pearson Education and Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 4th rev. edition, June 2005. ISBN 0201180596, 0321263545, 9780201180596, and 9780321263544. URL http://books.google.de/books?id=d63sQPvBezgC.
3. Friedemann Mattern. *Verteilte Basisalgorithmen*, volume 226 of *Informatik-Fachberichte (IFB)*. Berlin, Germany: Springer-Verlag GmbH, October 1989. ISBN 0-387-51835-5, 3540518355, 978-0-387-51835-0, and 9783540518358. URL http://books.google.de/books?id=EUC4AAAAIAAJ. Based on his dissertation at Prof. Dr. J. Nehmer's group / Sonderforschungsbereich "VLSI-Entwurf und Parallelität" at the computer science department of the University of Kaiserslautern.
4. Andrew Stuart Tanenbaum and Maarten van Steen. *Distributed Systems. Principles and Paradigms*. Upper Saddle River, NJ, USA: Prentice Hall International Inc. and Upper Saddle River, NJ, USA: Pearson Education, March 2003. ISBN 0130888931, 0131217860, 0132392275, 8178087898, 9780130888938, 978-0131217867, 9780132392273, and 9788178087894. URL http://books.google.de/books?id=7R-RGQAACAAJ.
5. Paulo Veríssimo and Luís Rodrigues. *Distributed Systems for System Architects*, volume 1 of *Advances in Distributed Computing and Middleware*. Berlin, Germany: Springer-Verlag GmbH, 2001. ISBN 0792372662 and 9780792372660. URL http://books.google.de/books?id=oOzwLX1_bpkC.
6. Michael J. Fischer, Nancy A. Lynch, and Michael S. Paterson. Impossibility of distributed consensus with one faulty process. *Journal of the Association for Computing Machinery (JACM)*, 32(2):374–382, April 1985. doi: 10.1145/3149.214121. URL http://discolab.rutgers.edu/classes/cs519-old/papers/p374-fischer.pdf.
7. Barry M. Leiner, Vinton G. Cerf, David D. Clark, Robert E. Kahn, Leonard Kleinrock, Daniel C. Lynch, Jonathan B. Postel, Lawrence G. Roberts, and Stephen Wolff. Brief history of the internet. *ACM SIGCOMM Computer Communication Review*, 39(5):22–31, October 2009. doi: 10.1145/1629607.1629613. URL http://www.sigcomm.org/sites/default/files/ccr/papers/2009/October/1629607-1629613.pdf.
8. Adam Langley and David G. Andersen. *verify.go*. Mountain View, CA, USA: Google Inc. URL https://code.google.com/p/go/source/browse/src/pkg/crypto/x509/verify.go#128.

Bibliography II

9. Adam Goodman. Bypassing google's two-factor authentication. *The Duo Bulletin: Notes on security, from the desks of Duo Security*, February 25, 2013. URL `https://blog.duosecurity.com/2013/02/bypassing-googles-two-factor-authentication/`.
10. Michael Messner. Multiple vulnerabilities in d'link dir-600 and dir-300 (rev b). *s3cur1ty: no one is safe...*, February 4, 2013.
11. Leslie Lamport. Time, clocks, and the ordering of events in a distributed system. *Communications of the ACM (CACM)*, 21(7):558–565, July 1978. doi: 10.1145/359545.359563. URL `http://research.microsoft.com/users/lamport/pubs/time-clocks.pdf`. Also: Report CA-7603-2911, Massachusetts Comptuter Association, Wakefield, Massachusetts, USA, March 1976.
12. Leslie Lamport, Robert Shostak, and Marshall Pease. The byzantine generals problem. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 4(3):382–401, July 1982. doi: 10.1145/357172.357176. URL `http://www.cs.cornell.edu/courses/cs614/2004sp/papers/lsp82.pdf`.
13. Leslie Lamport. A new solution of dijkstra's concurrent programming problem. *Communications of the ACM (CACM)*, 17 (8):453–455, August 1974. doi: 10.1145/361082.361093. URL `http://research.microsoft.com/en-us/um/people/lamport/pubs/bakery.pdf`.
14. Leslie Lamport. Subject: distribution, May 28, 1987. URL `http://research.microsoft.com/en-us/um/people/lamport/pubs/distributed-system.txt`.
15. Leslie Lamport. *LaTeX: A Document Preparation System. User's Guide and Reference Manual*. Reading, MA, USA: Addison-Wesley Publishing Co. Inc., 1994. ISBN 0201529831 and 9780201529838. URL `http://books.google.de/books?id=19pzDwEACAAJ`.
16. Donald Ervin Knuth. *The TeXbook (Computers and Typesetting, Volume A)*. Reading, MA, USA: Addison-Wesley Publishing Co. Inc., 1984. ISBN 0-201-13448-9. URL `http://www.ctan.org/tex-archive/systems/knuth/dist/tex/texbook.tex`.
17. Donald Ervin Knuth. *Sorting and Searching*, volume 3 of *The Art of Computer Programming (TAOCP)*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2nd edition, 1998. ISBN 0-201-03803-X, 0-201-03809-9, 0-201-03822-6, 0-201-85394-9, 0-201-89685-0, 978-0-201-03803-3, 978-0-201-03809-5, 978-0-201-03822-4, 978-0-201-85394-0, and 978-0-201-89685-5. URL `http://books.google.de/books?id=jYNQAAAAMAAJ`. Original from the University of Michigan.
18. Donald Ervin Knuth. *Fundamental Algorithms*, volume 1 of *The Art of Computer Programming (TAOCP)*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., third edition, 1997. ISBN 0-201-89683-4 and 978-0-201-89683-1. URL `http://books.google.de/books?id=J_MySQAACAAJ`.

Bibliography III

19. Donald Ervin Knuth. *Seminumerical Algorithms*, volume 2 of *The Art of Computer Programming (TAOCP)*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1969. ISBN 0-201-89684-2, 8177583352, 978-0-201-89684-8, and 978-8177583359. URL `http://books.google.de/books?id=OtLNKNVh1XoC`.
20. Ronald Graham, Donald Ervin Knuth, and Oren Patashnik. *Concrete Mathematics: A Foundation for Computer Science*. Reading, MA, USA: Addison-Wesley Publishing Co. Inc., 1988. ISBN 0-201-14236-8 and 0-201-55802-5.
21. Eric Korpela, Dan Werthimer, David Anderson, Jeff Cobb, and Matt Lebofsky. Seti@home – massively distributed computing for seti. *Computing in Science & Engineering*, 3(1):78–83, January–February 2001. doi: 10.1109/5992.895191. URL `http://setiathome.ssl.berkeley.edu/~korpela/papers/CISE.pdf`.
22. Miguel A. Vega-Rodríguez, David Vega-Pérez, Juan A. Gómez-Pulido, and Juan M. Sánchez-Pérez. Radio network design using population-based incremental learning and grid computing with boinc. In Mario Giacobini, Anthony Brabazon, Stefano Cagnoni, Gianni A. Di Caro, Rolf Drechsler, Muddassar Farooq, Andreas Fink, Evelyne Lutton, Penousal Machado, Stefan Minner, Michael O'Neill, Juan Romero, Franz Rothlauf, Giovanni Squillero, Hideyuki Takagi, A. Şima Uyar, and Shengxiang Yang, editors, *Applications of Evolutionary Computing, Proceedings of EvoWorkshops 2007: EvoCoMnet, EvoFIN, EvoIASP, EvoINTERACTION, EvoMUSART, EvoSTOC and EvoTransLog (EvoWorkshops'07)*, volume 4448/2007 of *Lecture Notes in Computer Science (LNCS)*, pages 91–100, València, Spain, 2007. Berlin, Germany: Springer-Verlag GmbH. doi: 10.1007/978-3-540-71805-5_10.