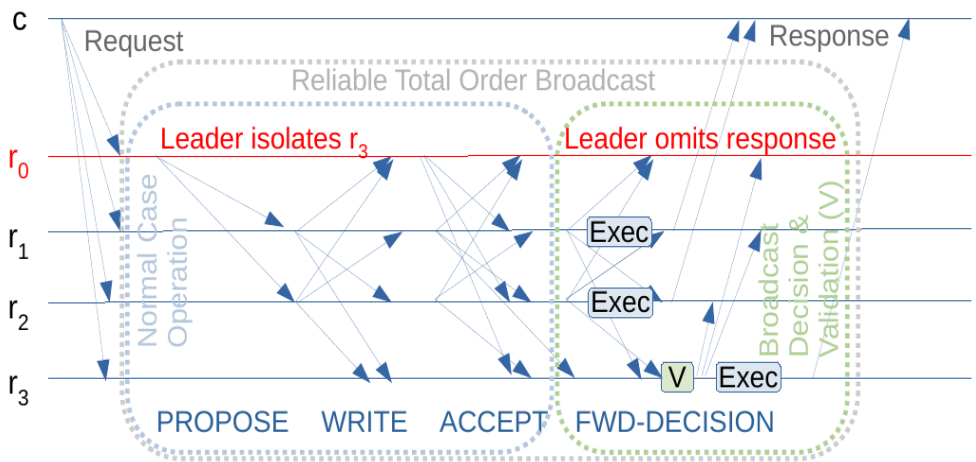
UNIVERSITÄT PASSAU
Fakultät für Informatik und Mathematik

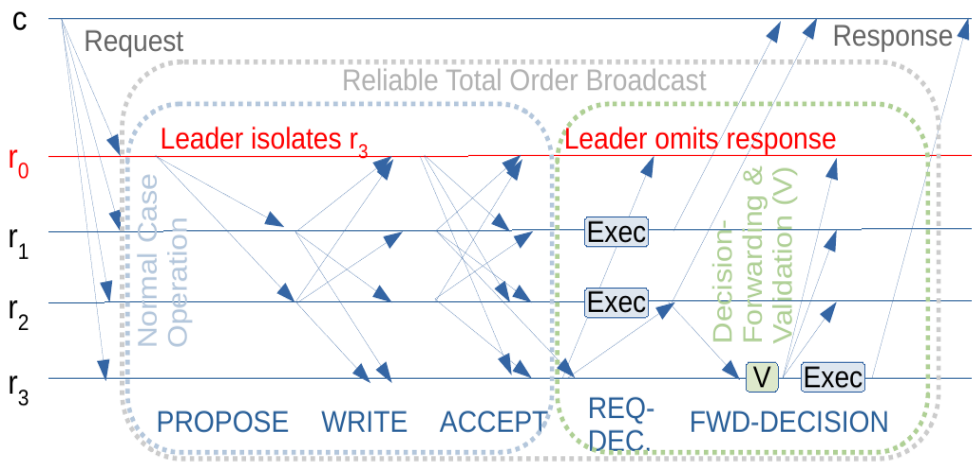
Dependable Distributed Systems – 5880V/UE

Part 1a: Organization, Introduction – 2021-10-04

Prof. Dr. Hans P. Reiser | WS 2021/22

UNIVERSITÄT PASSAU





Basic information

- This year: 2-week block lecture, Oct 4 - Oct 14
- Lecture 9-12h, exercises 13-16h
  (but we will flexibly adjust this if needed)
- Master PO2016: DDS $\rightarrow$ “IT security and reliability”
- Prerequisite:
  - Good basic understanding of computers, networks, operating systems and distributed systems
- Teaching language: English (unless *all* participants agree differently)

Basic information: Exam

- Oral exam (can be done online)
- First exam: planned for 03.03. and 04.03. (register before 12.12.2021)
- Second exam: planned for 04.04. (register before 25.03.2022)
- Registration for exams on HISQIS
- We will assign time slots in early Februray
  - Please pay attention to communication on Stud.IP. We may ask you for yours preferences for a specific day, online vs. on campus exam, etc.

Basic information: Lecture material

- Lecture material (slides as PDF document) available on Stud.IP
- Exercise questions will be published on Stud.IP
  - In the exercise classes, YOU shall try to answer all questions on your own
  - . . . and then present your solutions

Basic information

- `about:me`
  - Hans P. Reiser
  - Office: ITZ 137
  - `hr@sec.uni-passau.de`

  - Secretary: Siglinde Böck, ITZ 130
  - `sis-office@sec.uni-passau.de`

- Use Stud.IP forum for questions and discussions of potentially public interest!

Goals of this lecture

In-depth understanding of distributed systems which . . .

- . . . consist of many participants
- . . . have a high level of complexity
- . . . process large amounts of data
- . . . must be fault-tolerant
- . . . must be secure

Goals of this lecture

In-depth understanding of distributed systems which . . .

- . . . consist of many participants
- . . . have a high level of complexity
- . . . process large amounts of data
- . . . must be fault-tolerant
- . . . must be secure

Main focus: **distributed algorithms**

- A very interesting subject ☺

Literature

Main material:

- Material available on Stud.IP and ILIAS
- Scientific articles on specific topics

Recommended books:

- C. Cachin, R. Guerraoui, L. Rodrigues:
  Introduction to Reliable and Secure Distributed Programming, Springer, 2011
- M. Raynal:
  Concurrent Programming: Algorithms, Principles, and Foundations, 2013
- P. Veríssimo, L. Rodrigues:
  Distributed Systems for System Architects, Kluwer Academic Publishers, 2001. Part I (Distributed systems) and Part II (Fault tolerance)
- Vijay K. Garg: Elements of Distributed Computing, Wiley, 2002.
- Israel Koren, C. Mani Krishna: Fault-Tolerant Systems, Morgan Kaufmann, 2007.

Semester overview (preliminary plan)

1. Basic concepts of dependable distributed systems
2. Models, logical and physical clocks and time synchronization
3. Network Time Protocol and Precision Time Protocol

4. Fault tolerant group communication
5. Consensus and failure detectors
6. Replicated systems
7. Paxos algorithm
8. Byzantine fault-tolerant replication

9. Blockchain / Distributed Ledger Technology
10. Distributed coordination services (Zookeeper) and data storage (GFS etc.)
11. Decentralized self-organizing systems (P2P systems)

Additional remarks regarding the lecture

Feedback and questions:

- Ask question any time (including during lecture!)
- Point out errors!
- Give feedback on the material!
- Feel free to contact me:
  - personally (after lecture, or with appointment in my office)
  - E-mail (hr@sec.uni-passau.de)

Basic information

Questions regarding organization?

Overview

1 Organisation

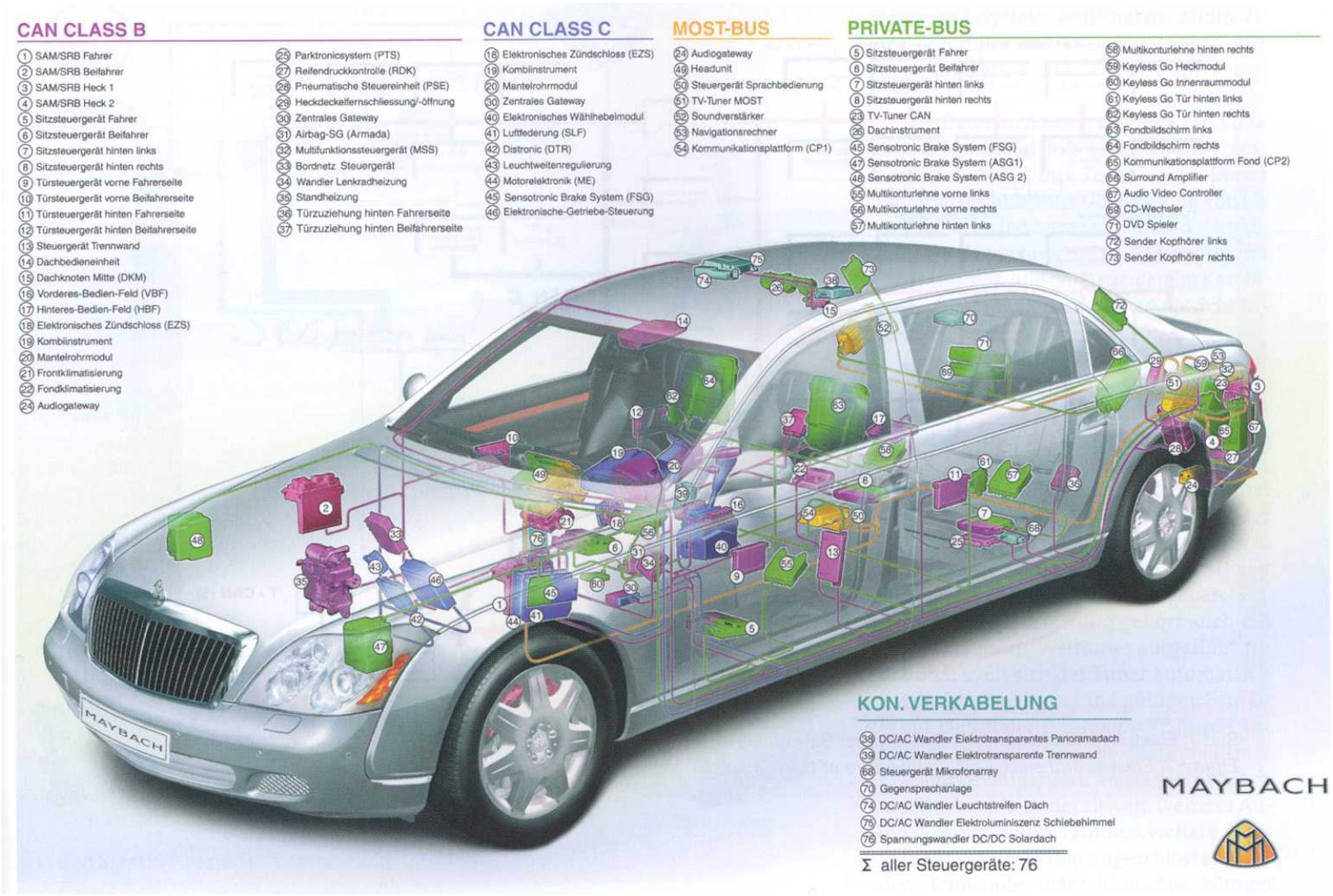
2 Introduction
- Distributed systems
- Basics of fault tolerance
- Ways towards dependability

Distributed Systems: Fundamentals

Overview

- What is a distributed system?
- Properties of a distributed system

What is a distributed system? (Example)

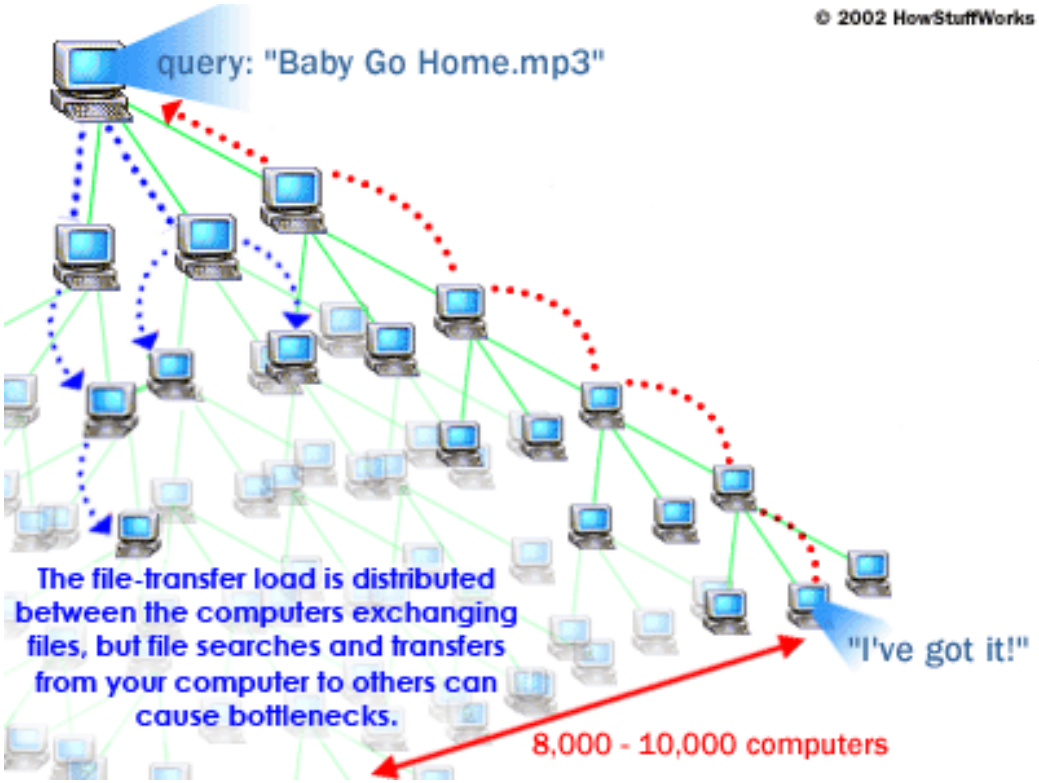


(source: Der neue Maybach, ATZ/MTZ Sonderheft September 2002, Seite 125)

What is a distributed system? (Example)

(source: NapoliRoma, CC BY-SA 3.0, https://commons.wikimedia.org/wiki/File:Sun_Modular_Datacenter.jpg)

What is a distributed system? (Example)

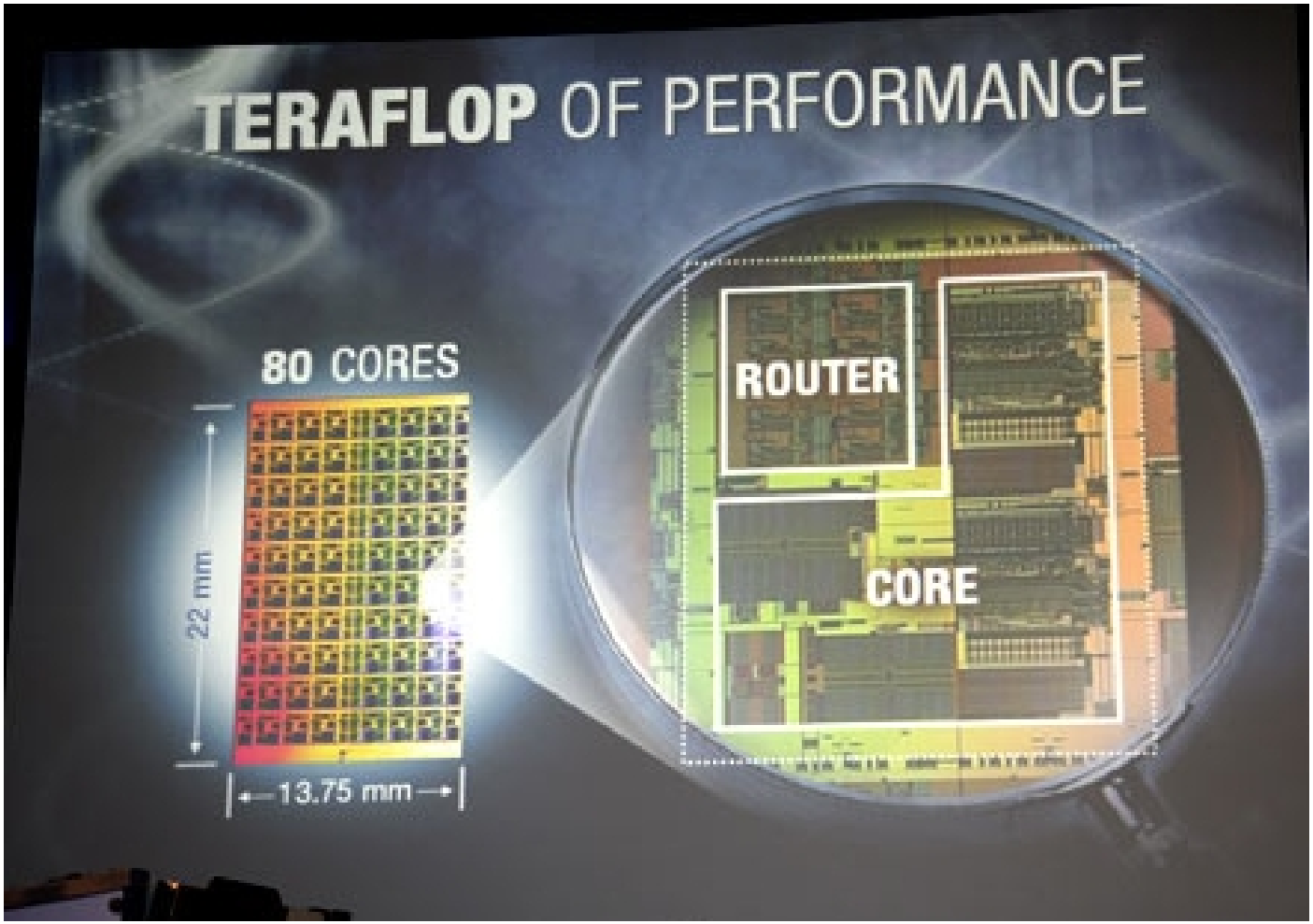


(source: Carmen Carmack "How BitTorrent Works" 26 March 2005. `https://computer.howstuffworks.com/bittorrent.htm`)

(source: Sandvine, 2012 (North America, Fixed Access))

	Upstream		Downstream		Aggregate	
Rank	Application	Share	Application	Share	Application	Share
1	BitTorrent	36.8%	Netflix	33.0%	Netflix	28.8%
2	HTTP	9.83%	YouTube	14.8%	YouTube	13.1%
3	Skype	4.76%	HTTP	12.0%	HTTP	11.7%
4	Netflix	4.51%	BitTorrent	5.89%	BitTorrent	10.3%
5	SSL	3.73%	iTunes	3.92%	iTunes	3.43%
6	YouTube	2.70%	MPEG	2.22%	SSL	2.23%
7	PPStream	1.65%	Flash Video	2.21%	MPEG	2.05%
8	Facebook	1.62%	SSL	1.97%	Flash Video	2.01%
9	Apple PhotoStream	1.46%	Amazon Video	1.75%	Facebook	1.50%
10	Dropbox	1.17%	Facebook	1.48%	RTMP	1.41%

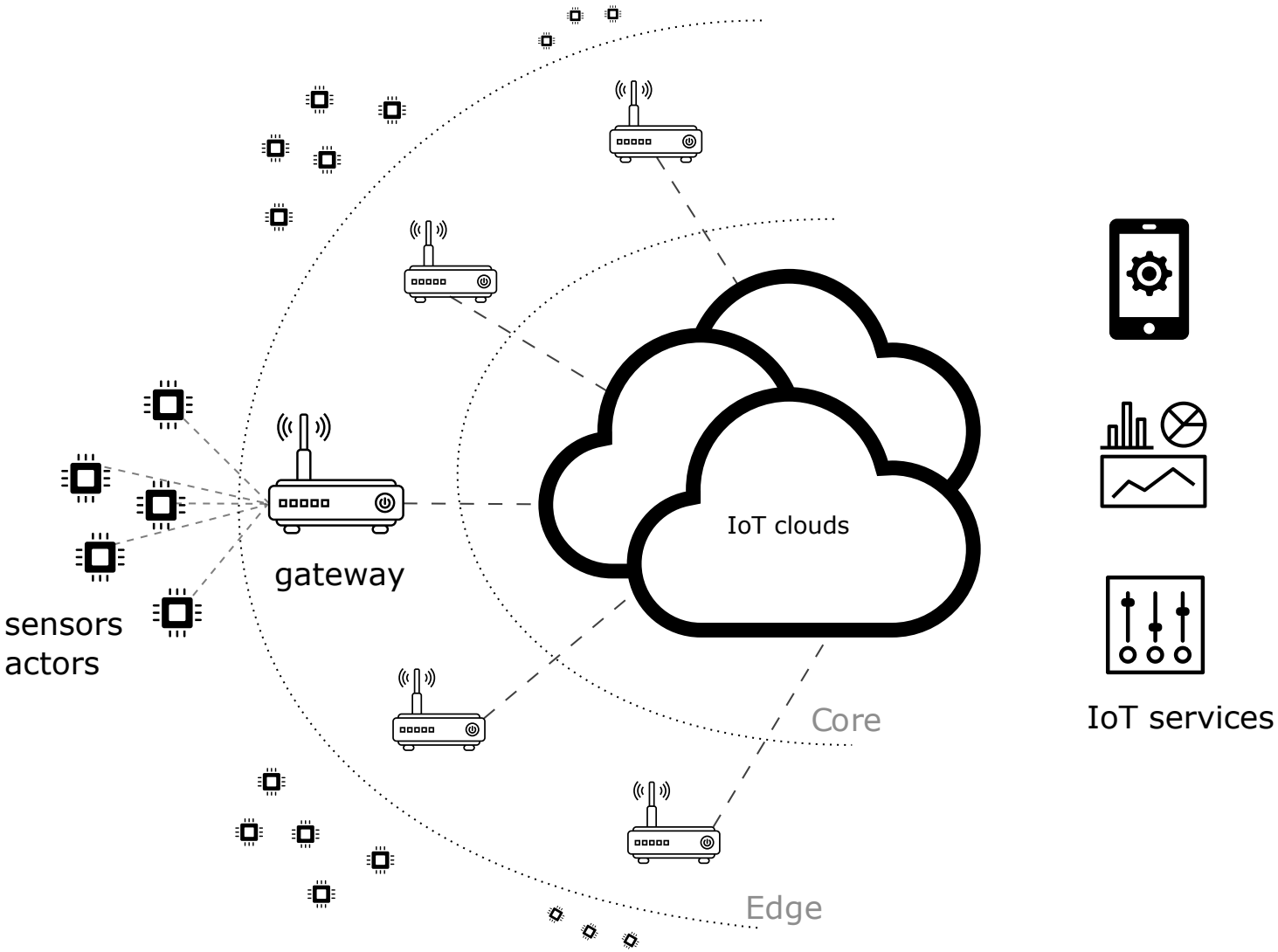
What is a distributed system? (Example)



(Completely different dimension, but problems similar to the other examples)

(source: Intel, Intel Developer Forum Fall 2006)

What is a distributed system? (Example)



What is a distributed system?

- A computer network is **not** distributed system

What is a distributed system?

- A computer network is **not** distributed system
- Computer network
  - Infrastructure that enables communication between machines
  - May have multiple media and topologies
  - Common communication protocols
- Distributed system
  - Composed of multiple computer systems
  - Communicate over a computer network
  - Set of processes that
    - use a common set of distributed protocols
    - to support a coherent execution
    - of distributed activities

Properties of a distributed system

- Several computers
- Connected by communication network
- Distributed global state

$\Rightarrow$

Properties of a distributed system

- Several computers
- Connected by communication network
- Distributed global state

$\Rightarrow$

- Independent failures

Properties of a distributed system

- Several computers
- Connected by communication network
- Distributed global state

$\Rightarrow$

- Independent failures
- Unreliable communication

Properties of a distributed system

- Several computers
- Connected by communication network
- Distributed global state

$\Rightarrow$

- Independent failures
- Unreliable communication
- Variable delays

Properties of a distributed system

- Several computers
- Connected by communication network
- Distributed global state

$\Rightarrow$

- Independent failures
- Unreliable communication
- Variable delays
- Moderate communication speed

Properties of a distributed system

- Several computers
- Connected by communication network
- Distributed global state

$\Rightarrow$

- Independent failures
- Unreliable communication
- Variable delays
- Moderate communication speed
- Only partial ordering of events

Properties of a distributed system

- Several computers
- Connected by communication network
- Distributed global state

$\Rightarrow$

- Independent failures
- Unreliable communication
- Variable delays
- Moderate communication speed
- Only partial ordering of events
- Global state difficult to determine

Properties of a distributed system

- Several computers
- Connected by communication network
- Distributed global state

$\Rightarrow$

- Independent failures
- Unreliable communication
- Variable delays
- Moderate communication speed
- Only partial ordering of events
- Global state difficult to determine
- Higher administrative costs

Properties of a distributed system

- Several computers
- Connected by communication network
- Distributed global state

$\Rightarrow$

- Independent failures
- Unreliable communication
- Variable delays
- Moderate communication speed
- Only partial ordering of events
- Global state difficult to determine

- Higher administrative costs
- Is often less expensive than centralized supercomputers

Distributed system: definitions

- “A distributed system is a collection of independent computers that appear to the users of the system as a single computer.”

  Andrew Tanenbaum

Distributed system: definitions

- No common clock
  - Distributed local clocks can be synchronized with only limited accuracy
- No shared memory
  - Common state not stored centrally;
    observing global system properties difficult
- No exact failure detection
  - No upper bounds on communication delays;
    failure not distinguishable from slow node

Vijay Garg

Centralized vs. distributed systems

Centralized:

- Easy accessibility
  - of resources and information
- Homogeneity
  - of methods and technologies
- Manageability
- Consistency
  - a global state
- Security
  - easier to protect

Distributed:

- Scalability, by
  - size
  - modularity
- Reliability and availability
  - redundancy and replication
  - graceful degradation
- Cheaper
- Security
  - by being able to compensate the impact of security problems
  - not really easy

Large Data Center (I)

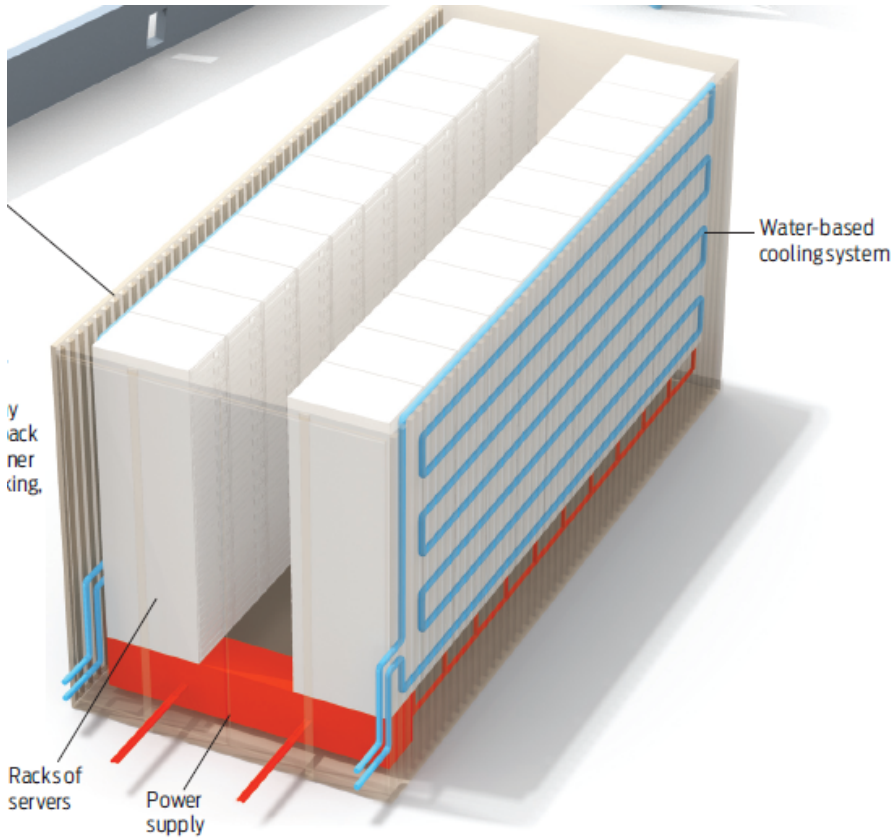
Google data center on the banks of the Columbia River. The site, with 2 server-packed buildings and space for a 3rd, houses tens of thousands of computers. Microsoft, Yahoo, and Amazon are also building data centers in the region.

- Randy Katz, Tech Titans Building Boom, IEEE Spectrum, Feb. 09

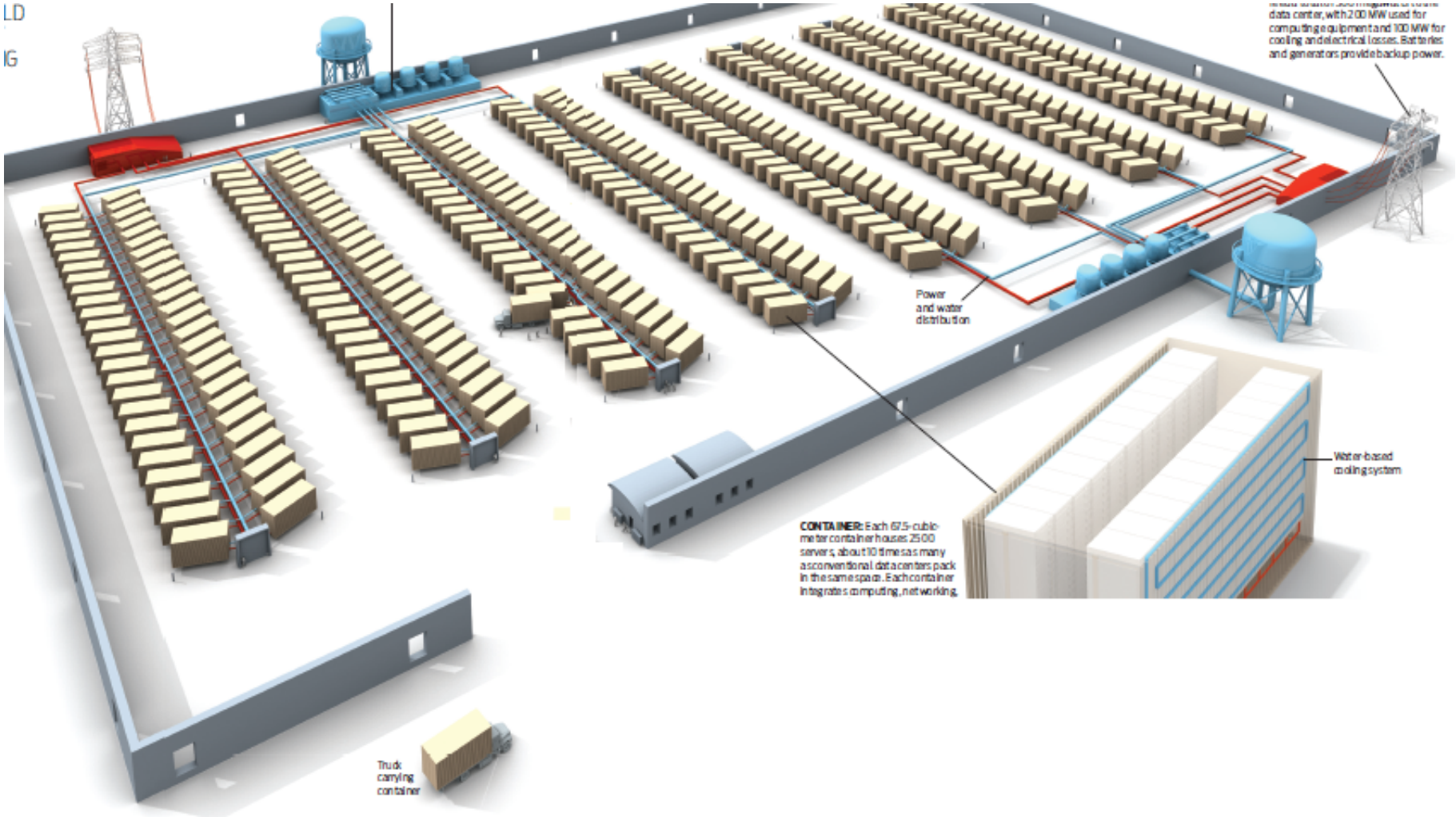
Large Data Center (II)

CONTAINER:

- Each 67.5-cubic-meter container houses 2500 servers, about 10 times as many as conventional data centers pack in the same space.
- Each container integrates computing, networking, power, and cooling systems.



Large Data Center (III)



Large Data Center (IV)

The Joys of Real Hardware

Typical first year for a new cluster:

~0.5 overheating (power down most machines in <5 mins, ~1-2 days to recover)
~1 PDU failure (~500-1000 machines suddenly disappear, ~6 hours to come back)
~1 rack-move (plenty of warning, ~500-1000 machines powered down, ~6 hours)
~1 network rewiring (rolling ~5% of machines down over 2-day span)
~20 rack failures (40-80 machines instantly disappear, 1-6 hours to get back)
~5 racks go wonky (40-80 machines see 50% packetloss)
~8 network maintenances (4 might cause ~30-minute random connectivity losses)
~12 router reloads (takes out DNS and external vips for a couple minutes)
~3 router failures (have to immediately pull traffic for an hour)
~dozens of minor 30-second blips for dns
~1000 individual machine failures
~thousands of hard drive failures
slow disks, bad memory, misconfigured machines, flaky machines, etc.

Long distance links: wild dogs, sharks, dead horses, drunken hunters, etc.

source: Jeffrey Dean, Google, https://research.google/people/jeff/

Overview

1 Organisation

2 Introduction
- Distributed systems
- Basics of fault tolerance
- Ways towards dependability

Fault Tolerance: Basics

Why fault tolerance?

Computer failures

Why do computers stop and what can be done about it? (J. Gray, 1985)

Computer failures

Why do computers stop and what can be done about it? (J. Gray, 1985)

- Because:
  - Everything that works will fail at some time
  - Hardware and software is generally overestimated
- Better prevent errors instead of repairing later
  - In a predictable and repeatable manner
- We need:
  - a scientific way to *quantify, predict, prevent and tolerate* disturbances that disrupt the normal operation of a system

Definition

Laprie 1993

"Computer system dependability is the quality of a delivered service such that reliance can justifiably be placed on this service"

Definition

Laprie 1993

"Computer system dependability is the quality of a delivered service such that reliance can justifiably be placed on this service"

A reliable system is a system that with high probability works according to its specification

- A simple definition . . .

Definition

Laprie 1993

"Computer system dependability is the quality of a delivered service such that reliance can justifiably be placed on this service"

A reliable system is a system that with high probability works according to its specification

- A simple definition . . .
  . . . but some open questions:
  - How is the service specified?
  - What exactly is specified?
    (the service itself? the environment?)
  - What is a "high probability"?

Impact of distribution on dependability?

> "A distributed system is one in which the failure of a computer you did not even know existed can render your own computer unusable."
>
> *Leslie Lamport*

Impact of distribution on dependability?

> "A distributed system is one in which the failure of a computer you did not even know existed can render your own computer unusable."
>
> *Leslie Lamport*

- A failure of some component might influence other components!
- Communication usually happens
  - over unreliable networks
  - with unpredictable delays

Impact of distribution on dependability?

- Failure of a group of computers:
  - If a single components fails on average once per year. . .
    (and all components fail independently)
  - . . . how many failures per year will you have in a datacenter using 1000 of these components?

Malicious faults (attackers!) make it even worse

- Errors are no longer independent
- Errors are more serious
- Error models are inaccurate
  (. . . stochastic model of an attacker?)

Dependability attributes

Basic definitions: (source: DSSA)

- Reliability
- Maintainability
- Availability

Dependability attributes

Basic definitions: (source: DSSA)

- Reliability
  - Measure of the continuous delivery of correct service
  - Metrics:
    - “Mean Time To Failure” (MTTF)
    - “Mean Time Between Failures” (MTBF)
    - Probability that the system does not fail during mission period
    - Probabilistic failure rate (e.g., 10^{-9} failures/hour)
- Maintainability
- Availability

Dependability attributes

Basic definitions: (source: DSSA)

- Reliability
  - Measure of the continuous delivery of correct service
  - Metrics:
    - “Mean Time To Failure” (MTTF)
    - “Mean Time Between Failures” (MTBF)
    - Probability that the system does not fail during mission period
    - Probabilistic failure rate (e.g., 10^{-9} failures/hour)
- Maintainability
  - Measure of the time to restoration of correct service
  - Metric: “Mean Time To Repair (MTTR)”
- Availability

Dependability attributes

Basic definitions: (source: DSSA)

- Reliability
  - Measure of the continuous delivery of correct service
  - Metrics:
    - “Mean Time To Failure” (MTTF)
    - “Mean Time Between Failures” (MTBF)
    - Probability that the system does not fail during mission period
    - Probabilistic failure rate (e.g., 10^{-9} failures/hour)

- Maintainability
  - Measure of the time to restoration of correct service
  - Metric: “Mean Time To Repair (MTTR)”

- Availability
  - Measure of the delivery of correct service
    with respect to the alternation
    between correct and incorrect service
  - Availability = MTBF / (MTBF+MTTR)

Other properties

- Integrity
  - Absence of improper system alterations
- Security
  - Ability to guarantee confidentiality, integrity and availability in service provision
- Resilience
  - Ability to recover from external faults
  - Frequently used when considering applications that require interactions through their surrounding environment

Faults, errors and failures

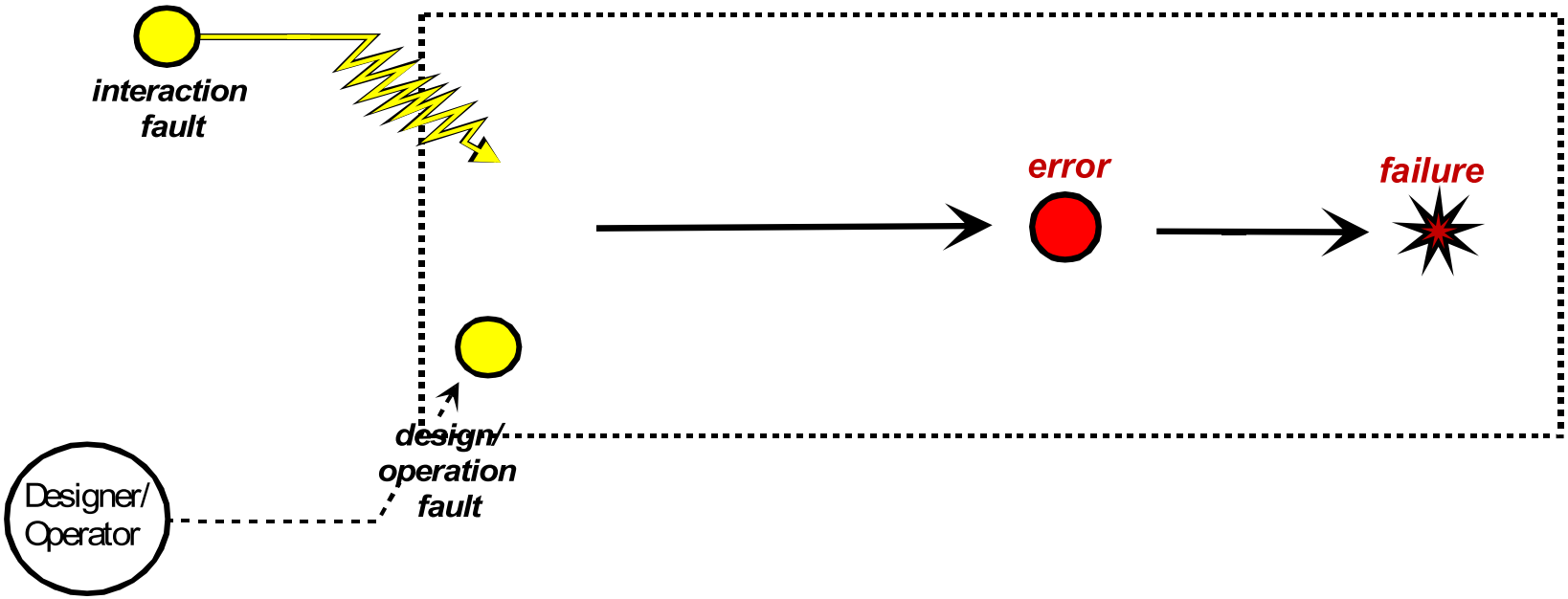
- A system **failure** occurs when the delivered service deviates from fulfilling the system function
- An **error** is that part of the system state which is liable to lead to subsequent failure
- The adjudged cause of an error is a **fault**
- These notions are for a system!

Faults, errors and failures

- A system **failure** occurs when the delivered service deviates from fulfilling the system function
- An **error** is that part of the system state which is liable to lead to subsequent failure
- The adjudged cause of an error is a **fault**
- These notions are for a system!

- Example (system = RAM memory)
  - Fault — stuck-at-'0' RAM memory register
  - Error — state after '1' is written to RAM
  - Failure — read returns wrong value '0'

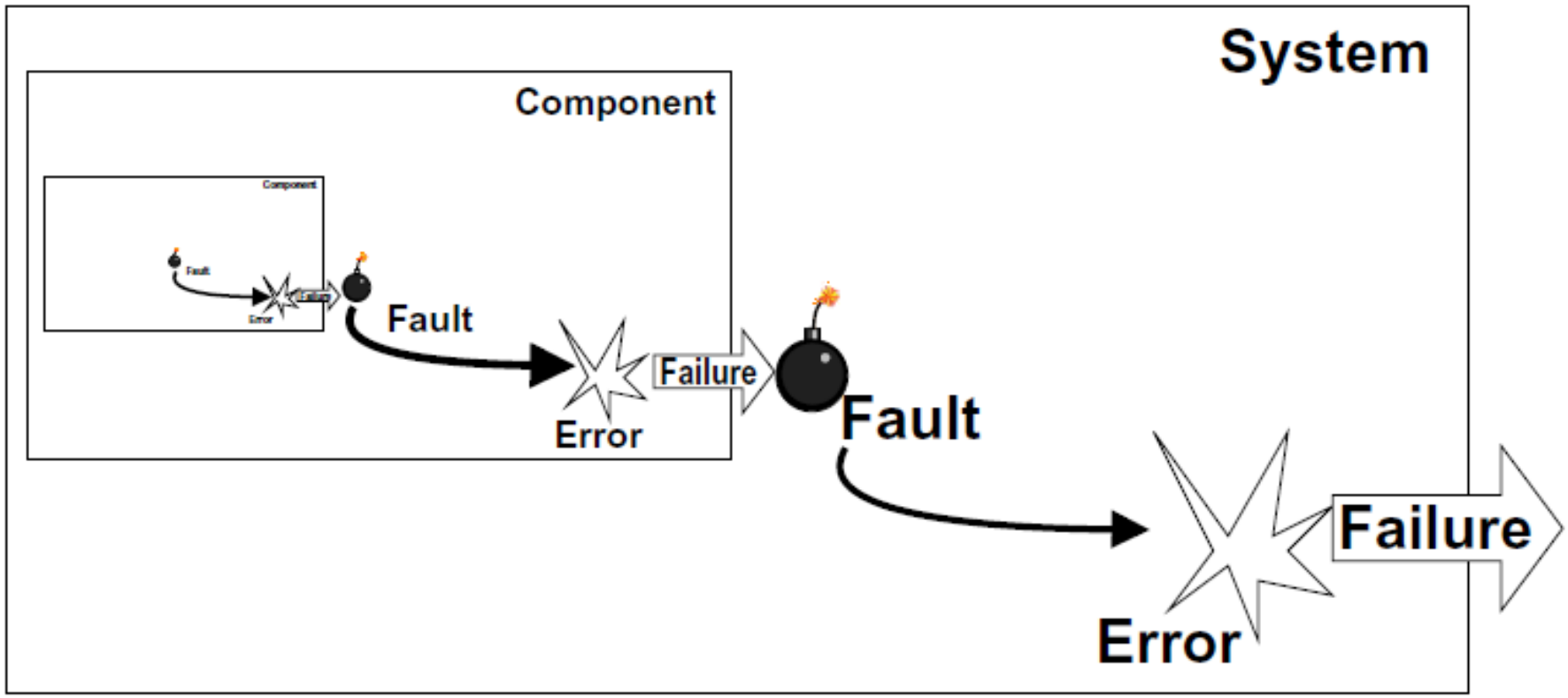
Sequence fault – error – failure



(source: Paulo Veríssimo)

- The relation is recursive!
  . . . What is a system? What is a component?

Fault – error – failure recursiveness



(source: Paulo Veríssimo)

Today's lecture

1. Organisation

2. Introduction
   - Distributed systems
   - Basics of fault tolerance
   - Ways towards dependability

Ways towards dependability

- **Fault prevention** — how to prevent the occurrence of faults?
  - Avoid design defects (good process; tools; etc.)
  - Shielding, robust design, good operational procedures etc. for runtime faults
- **Fault tolerance** — how to deliver correct service in the presence of faults?
- **Fault removal** — how to reduce the number/severity of faults?
  - Verification & validation (design phase)
  - Corrective & preventive maintenance (operations)
- **Fault forecasting** — how to estimate the future incidence of faults?
  - Simulation, modeling, prediction (data from models)
  - Historical data, accelerated life testing, etc. (data from real systems)

Ways towards dependability

- **Fault prevention** — how to prevent the occurrence of faults?
  - Avoid design defects (good process; tools; etc.)
  - Shielding, robust design, good operational procedures etc. for runtime faults
- **Fault tolerance** — how to deliver correct service in the presence of faults?
  - $\Longrightarrow$ **Main topic of lecture**
- **Fault removal** — how to reduce the number/severity of faults?
  - Verification & validation (design phase)
  - Corrective & preventive maintenance (operations)
- **Fault forecasting** — how to estimate the future incidence of faults?
  - Simulation, modeling, prediction (data from models)
  - Historical data, accelerated life testing, etc. (data from real systems)

Fault tolerance

Elements of fault tolerance:

- Error detection
- Recovery
  - Error handling
    - Backward recovery
    - Forward recovery
    - Compensation/masking
  - Fault handling

Error detection

- Reactive detection
  - Detects and reports errors at run-time (during operation)
  - Parity bits on memory, data buses, ALU, etc.
  - Coverage and flow-checking
    - Did the software hit all required checkpoints?
    - Multi-version comparison and majority voting
    - Compute multiple times and compare

Error detection

- Reactive detection
  - Detects and reports errors at run-time (during operation)
  - Parity bits on memory, data buses, ALU, etc.
  - Coverage and flow-checking
    - Did the software hit all required checkpoints?
    - Multi-version comparison and majority voting
    - Compute multiple times and compare
- Proactive detection
  - Detects and reports errors at development-time (offline)
  - Idea is to activate any faults before they can affect real computations
  - Execute self-tests, data consistency checks
  - Exercise system to activate faults
    - Memory scrubbing daemon that reads/resets all memory locations periodically

Error/fault recovery

- The objective is to restore a system state with errors and (possibly) faults into one without errors and faults
- Recovery can be done by:
  - Fault handling
  - Error handling
    - Backward recovery
    - Forward recovery
    - Error compensation or error masking

Fault handling

Record faults to avoid later reactivation

- Fault diagnosis: identify cause
  - Once diagnosed, avoid using that resource/performing that operation again
  - Also known as the “Don’t do that” solution
- Fault isolation: eliminate access to source of fault
  - Exclude faulty component from use
  - Shut down defective server in server farm
- System reconfiguration
  - Switch in spares or reassign tasks
  - Route traffic to backup component
- System reinitialization
  - Rebooting after repair/reconfiguration
  - Rebooting to perform “software rejuvenation”

Error handling

- Roll-back: return system to its state before the error occurred
  - Database commit (all-or-nothing execution semantics)
    - If error, undo all partial results to restore clean system state
  - Checkpoint/rollback
    - Periodically snapshot system state and restore upon error to avoid failure
    - Reboot is a very drastic rollback (reboot) to a checkpoint (freshly system image)
- Roll-forward: move the system forward into an error-free state
  - Retry to obtain error-free result
  - Proceed on and let error flush itself out of system
    - Replication where a backup takes over after the working replica fails

Error masking

- How?
  - The system state has enough **redundancy** that the correct service can be provided without any noticeable glitch
  - Recovery is achieved without explicit error detection
- What's good?
  - User is blissfully unaware of anything bad happening
  - Fault/failure transparency — great way to build systems!
- What's bad?
  - Independent failure assumption
  - Does not deal adequately with design faults

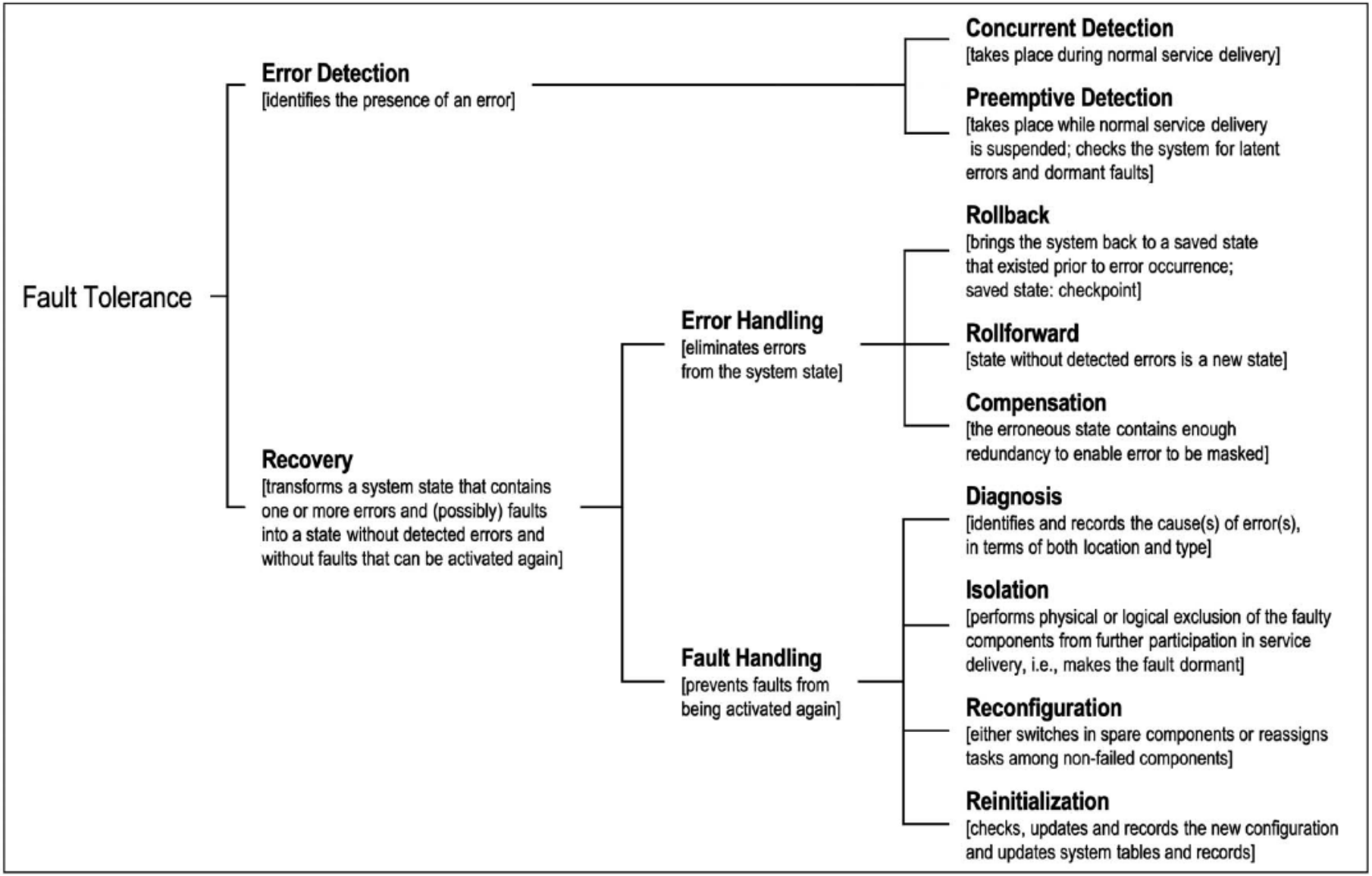
Forms of redundancy

- Space redundancy
  - Several copies of the same component
  - Same information stored in several disks
  - Different nodes compute same result in parallel
  - Messages disseminated along different paths
- Time redundancy
  - Doing the same thing more than once, in same or different ways
  - Retransmission of lost messages
  - Repeating computations that have aborted
- Value redundancy
  - Adding extra information about the data being stored or sent
  - Codes that allow the detection or correction of integrity errors
  - Parity bit or ECC added to memory chips or disk structures
  - Frame check sequences or cyclic redundancy in transmitted data cryptographic message signatures

The “Dependability Tree”

IMPAIRMENTS	• **FAULTS** • **ERRORS** • **FAILURES**
MEANS	• **FAULT PREVENTION** • **FAULT TOLERANCE** • **FAULT REMOVAL** • **FAULT FORECASTING**
ATTRIBUTES	• **RELIABILITY** • **MAINTAINABILITY** • **AVAILABILITY** • **SAFETY** • **CONFIDENTIALITY** • **INTEGRITY** • **AUTHENTICITY**

The “Fault Tolerance Tree”



Avizzienis et al.: Basic concepts and taxonomy of dependable and secure computing, TDSC v1(1), March 2004

Resilience vs. dependability

Resilience definition (Laprie)

The persistence of service delivery that can justifiably be trusted, when facing changes.

J.-C. Laprie: “From dependability to resilience”, DSN, 2008

Resilience definition (Berger et al.)

Resilience is the property of preserving the dependability and security of a system when the system encounters changes, thus withstanding or recovering from impairments.

Berger et al., “A Survey on Resilience in the IoT: Taxonomy, Classification and Discussion of Resilience Mechanisms”, ACM CSUR, 2021

Summary

Basic concepts you should remember

- What characterizes a distributed system?
  - Properties and problems
- What is dependability?
  - Dependability attributes
  - Faults, errors, and failures
  - Ways towards dependability