ÖREBRO UNIVERSITY

Programming of Distributed Systems

Topic I – Introduction to Distributed Systems

Dr.-Ing. Dipl.-Inf. Erik Schaffernicht

General Information

E-Mail

erik.schaffernicht@oru.se

Drop-in office hours

None. Drop me an email.

Blackboard ULTRA

Lecture slides, Exercises

Schedule

Lectures (week 44-50)

2 sessions per week (varying times on Mo, Tu or We → Kronox)

Seminar (week 46, 47, 49, 50)

1 session per week (varying times – signed with est in Kronox)

1 reserved lab time per week (signed with eget arbete in Kronox)

Written examination (week 2)

10th of January, 15:15 – 18:15 (prelim.)

General Information

Course credits: 7.5 HP/ETCS → **200 hours** spent on the topic

14 lectures	→ 21 hours
4 seminars	→ 12 hours
examination	→ 3 hours

To pass the seminar part (3 points)
- Participate in group work and present to your fellow students

To pass the theory part (4.5 points)
- Pass the exam
  (A-F ← course grade)

Remaining **~160** hours: self/group study / reading preparation and wrap-up of lectures / seminars / labs & exam

Four seminars

Purpose:
Presentation and discussion of group assignments

Assignments are given two weeks in advance of every session
(➔ maximum of two assignments active, first one today)

1st: repetition of concepts from Computer Communication & Networks

2nd: mini lectures given by you

3rd – 4th: Programming tasks in distributed systems

Exam

Time: 3 hours

- focus is not on memorized knowledge
- understanding and ability to apply knowledge is

More details in the assessment criteria found on Blackboard & towards the end of the course.

Course Literature

Steen, Tanenbaum:

Distributed Systems

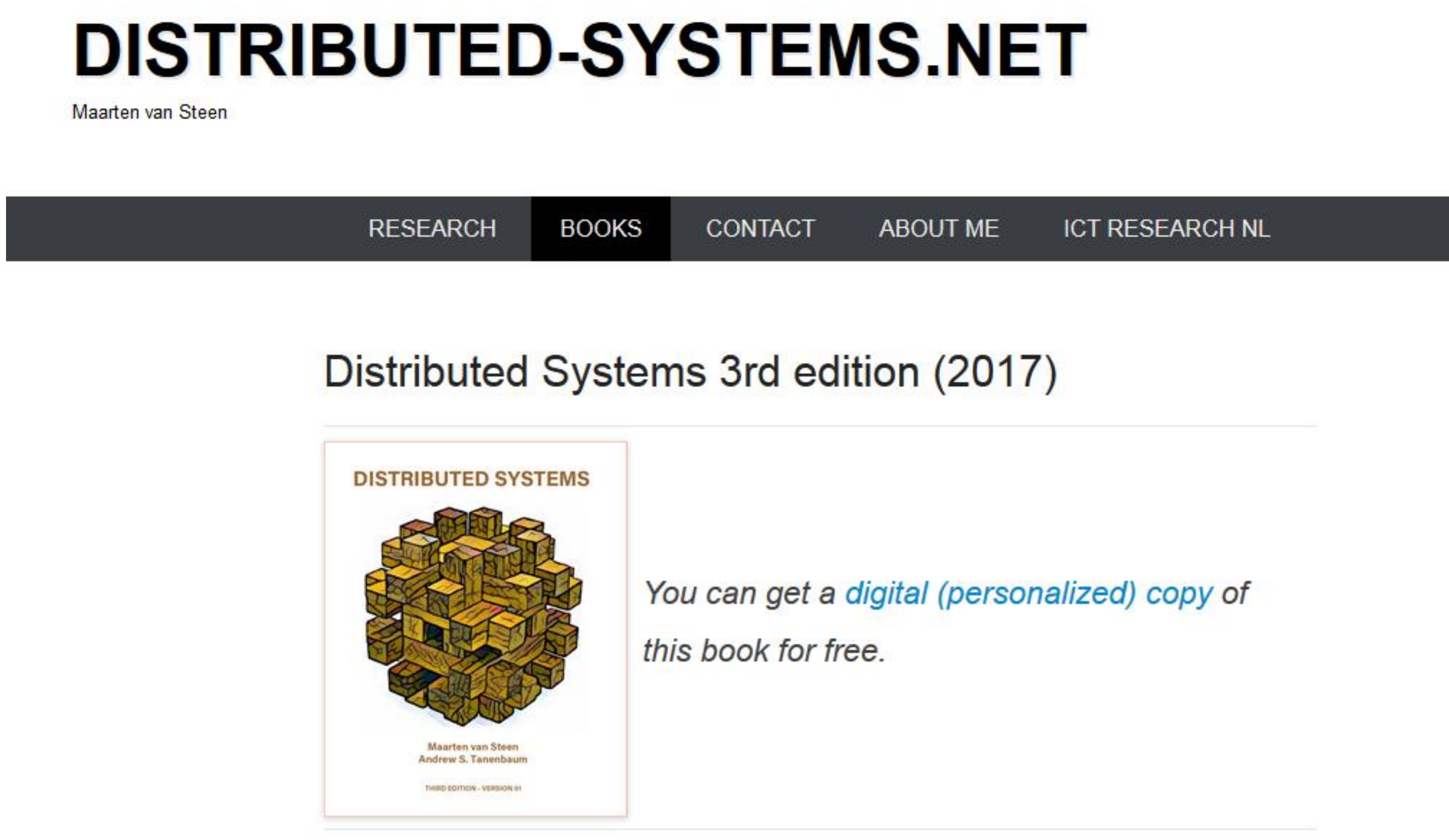
https://www.distributed-systems.net/

→ Books

→ Distributed Systems 3rd edition (2017)

→ Reference Version 3.02

Link is also on Blackboard



DISTRIBUTED-SYSTEMS.NET

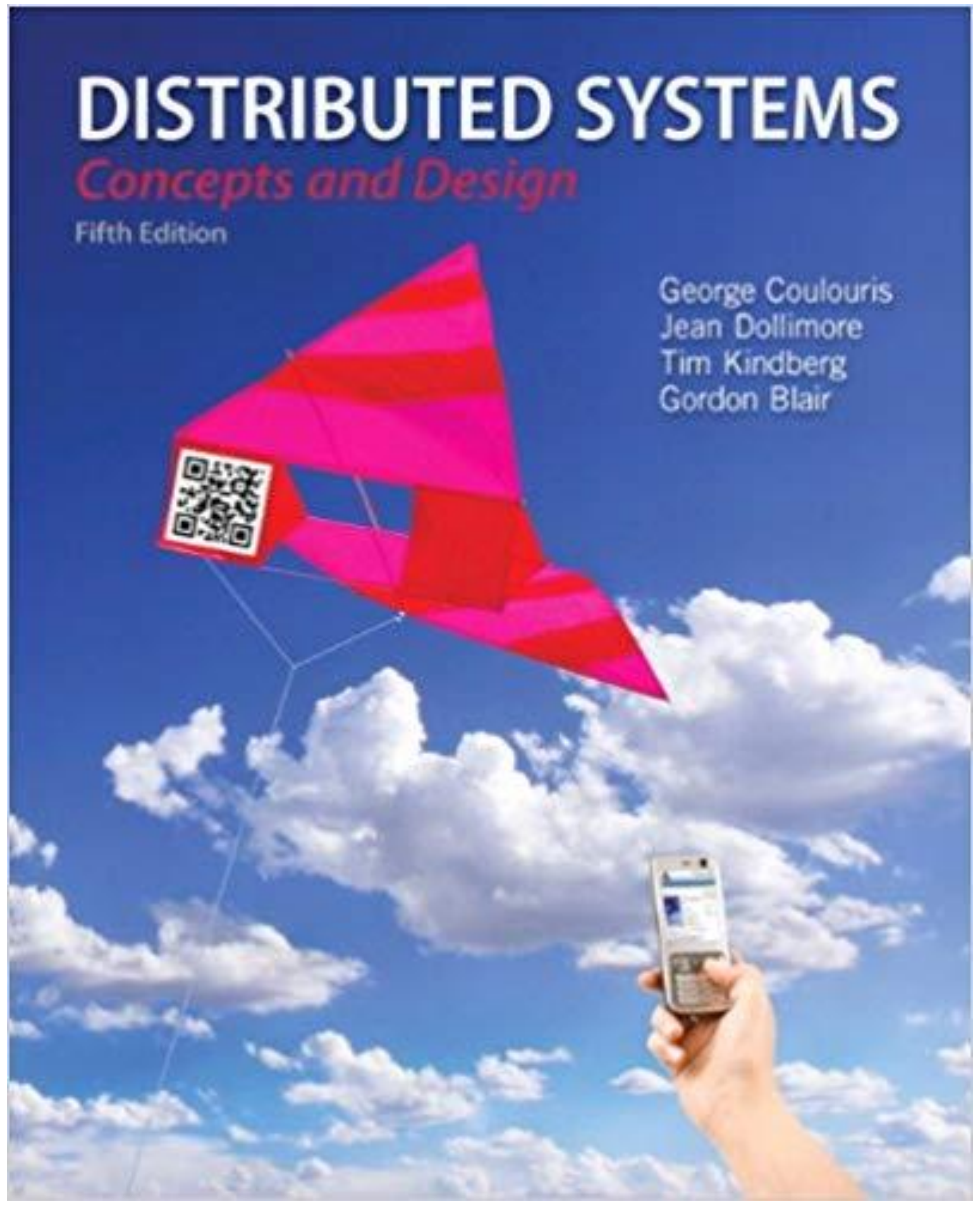
Maarten van Steen

Distributed Systems 3rd edition (2017)

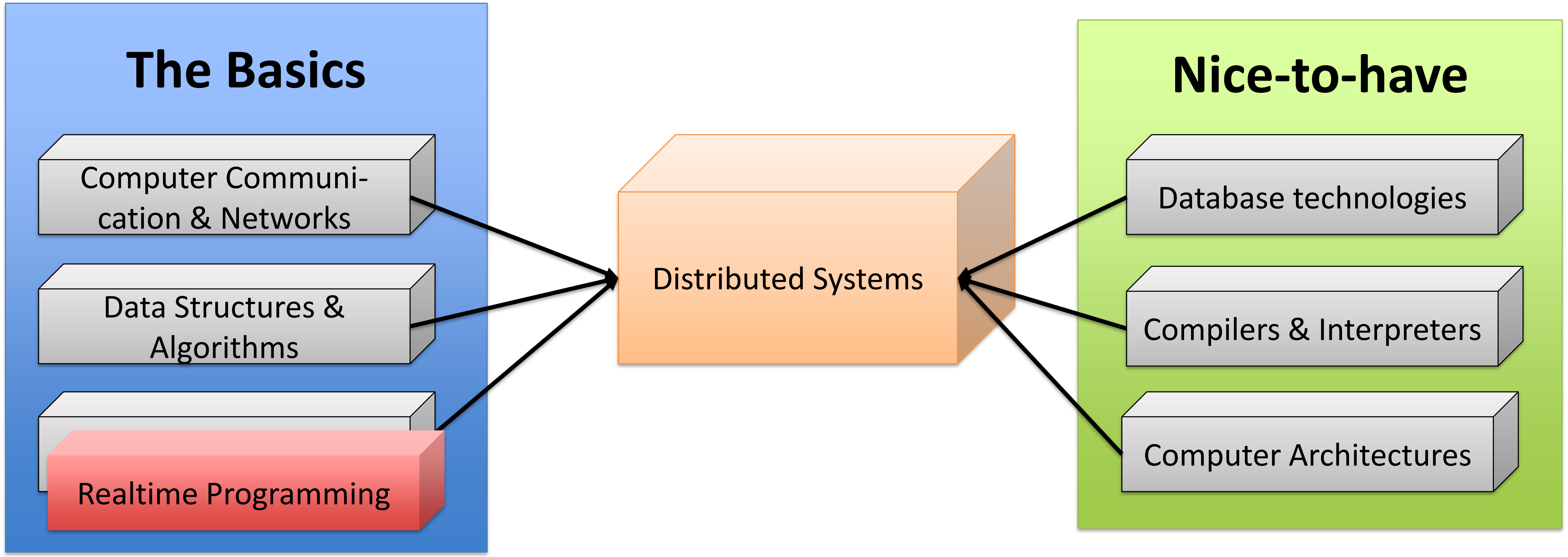
You can get a digital (personalized) copy of this book for free.

Additional reading

Coulouris, Dollimore, Kindberg and Blair:
Distributed Systems: Concepts and Design (5th Ed.)
Addison Wesley (2012)



Place in the program



Questions?

Reading Remarks

Reading Task:
Chapter 1 of the course book!

What is a distributed system?

A distributed system is a collection of **autonomous** computing elements that **appears** to its users as a single coherent system.

From Steen, Tanenbaum. Distributed Systems (2017), p.2

A distributed system is one in which components located at **networked computers** communicate and coordinate their actions **only** by passing **messages**.

From: Coulouris, Dollimore, Kindberg and Blair.
Distributed Systems: Concepts and Design (2012), p.1

What is a distributed system?

> Distributed programming is the art of solving the same problem that you can solve on a single computer using multiple computers.

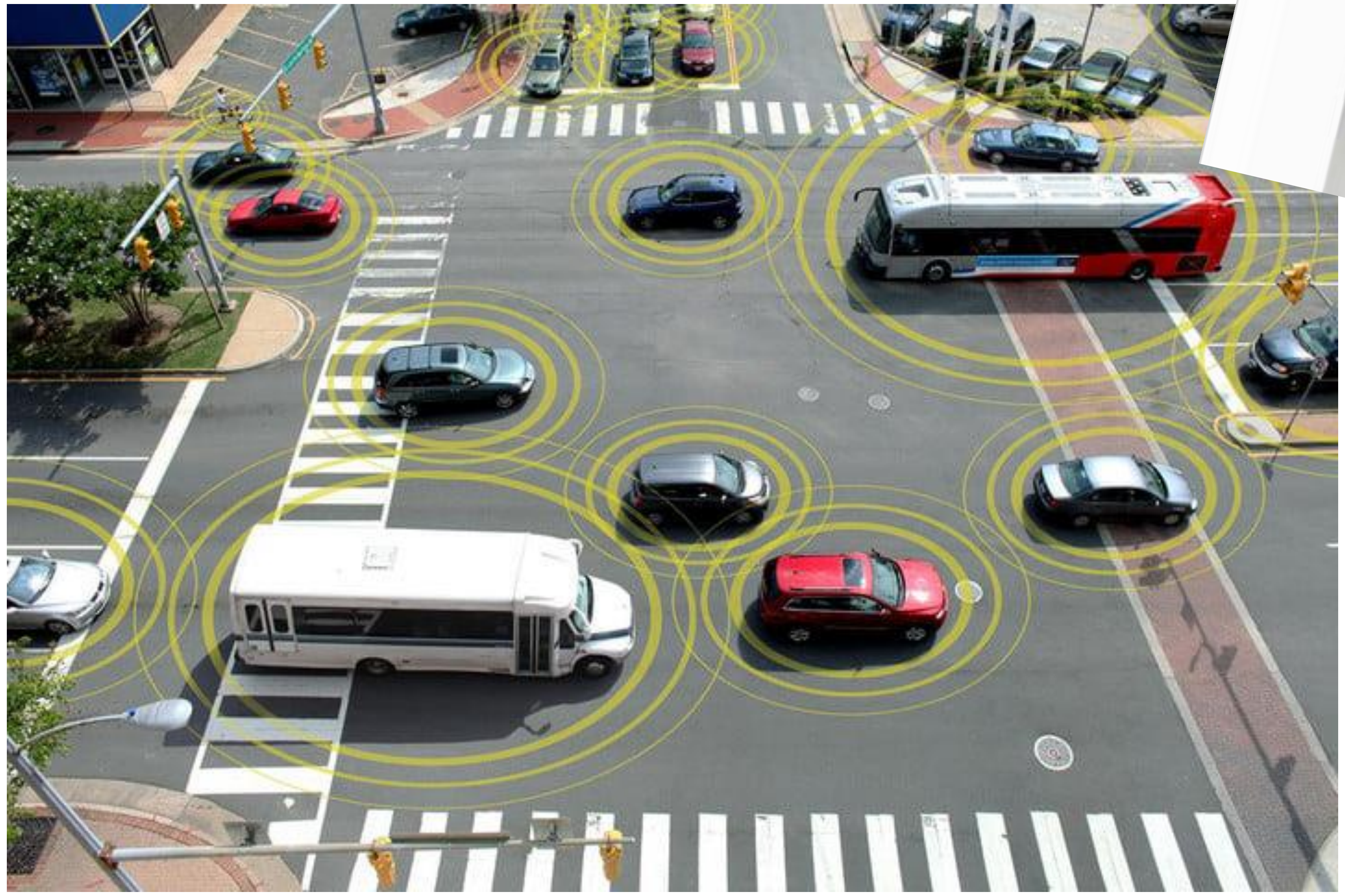
From Takada, Distributed Systems for Fun and Profit

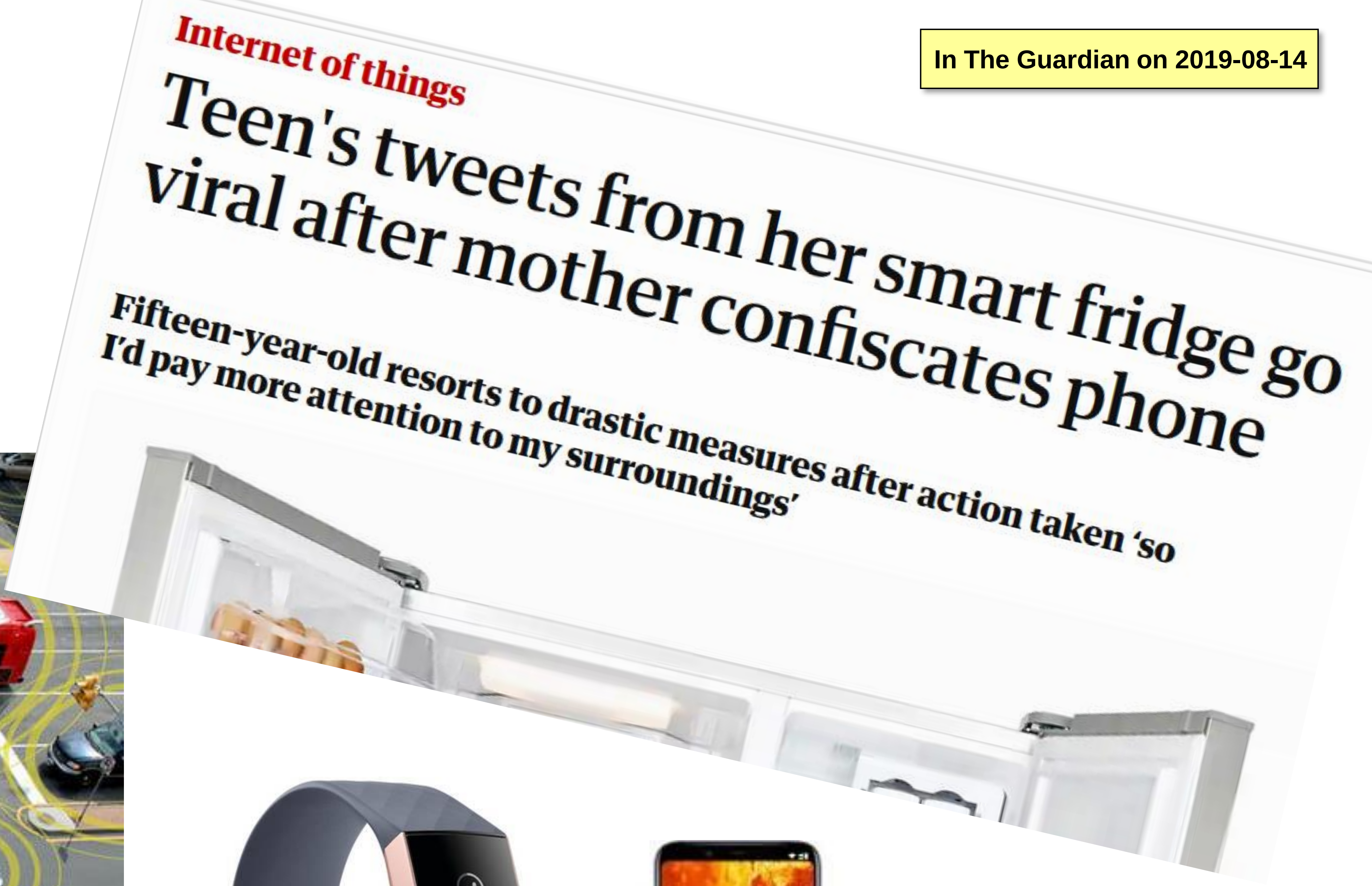
> A distributed system is one in which the failure of a computer you didn't even know existed can render your own computer unusable.

Leslie Lamport

In The Guardian on 2019-08-14

Examples



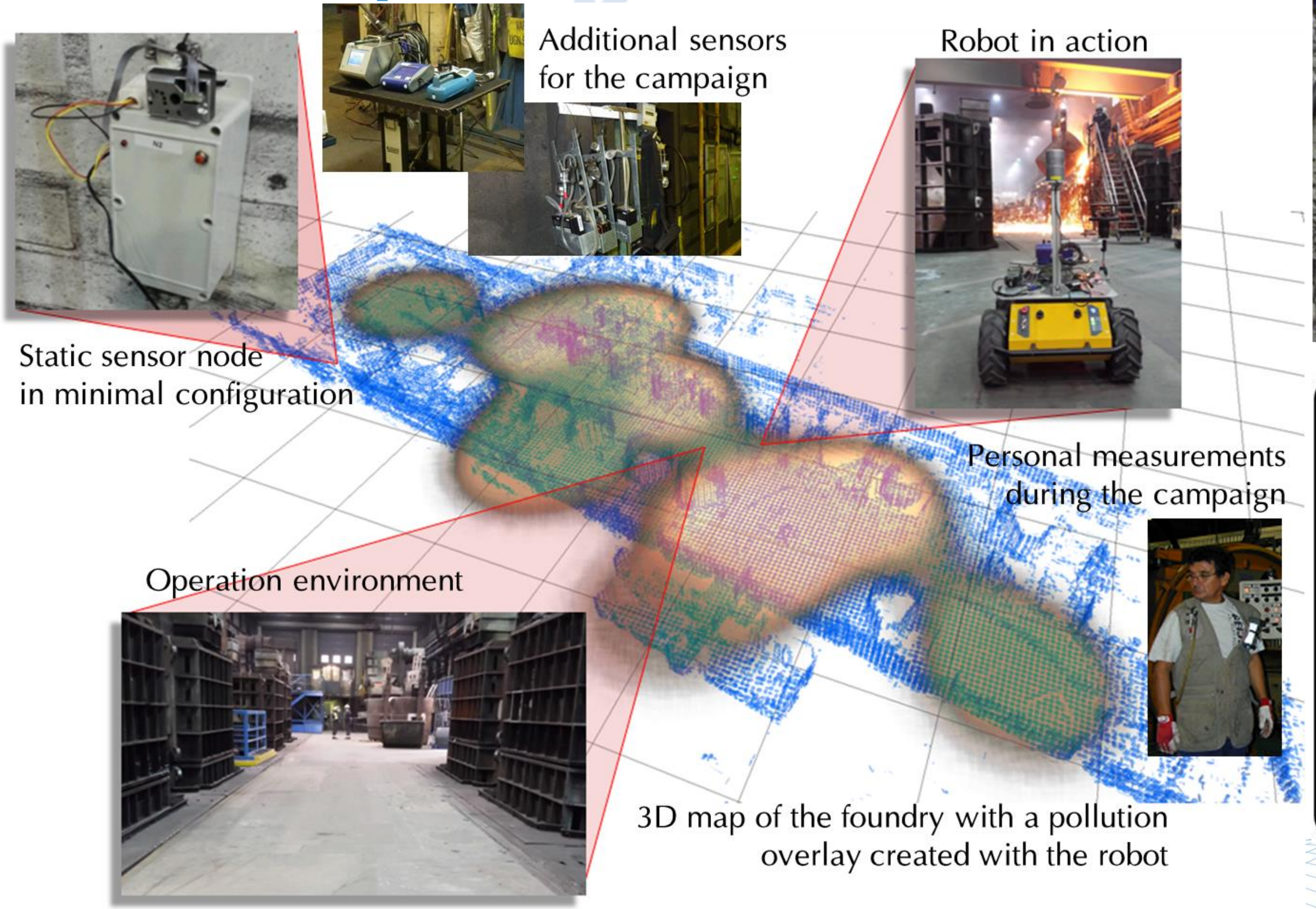

Internet of things

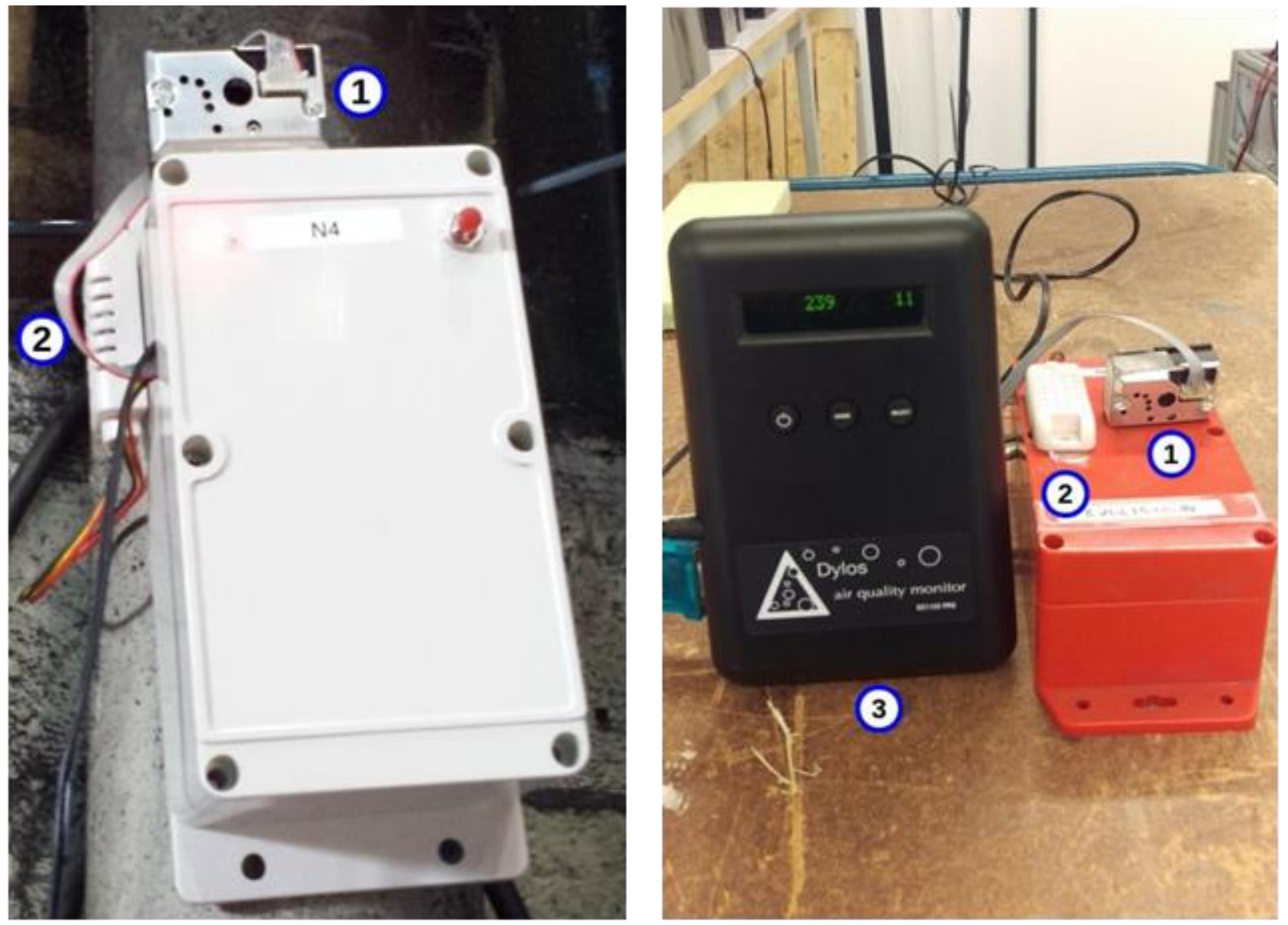
Teen's tweets from her smart fridge go viral after mother confiscates phone

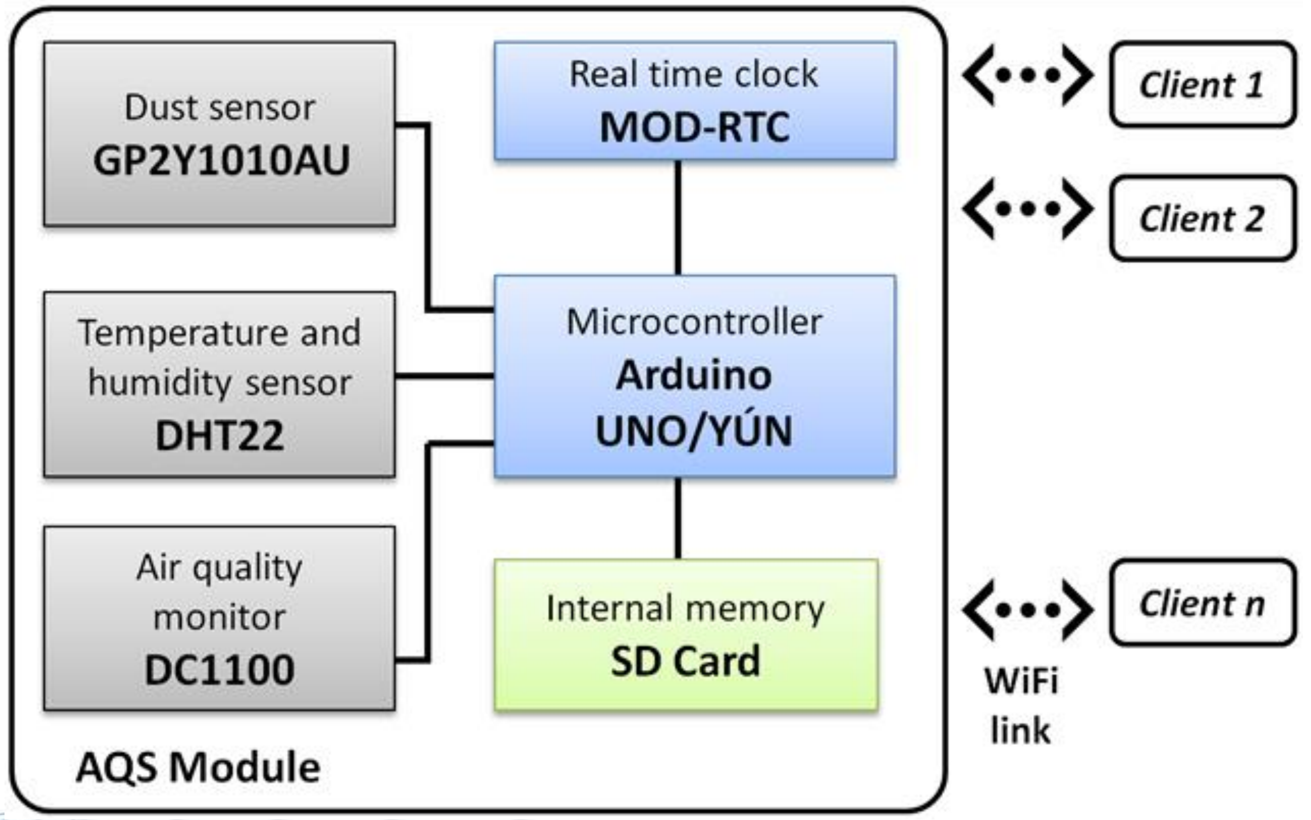
Fifteen-year-old resorts to drastic measures after action taken 'so I'd pay more attention to my surroundings'

Examples

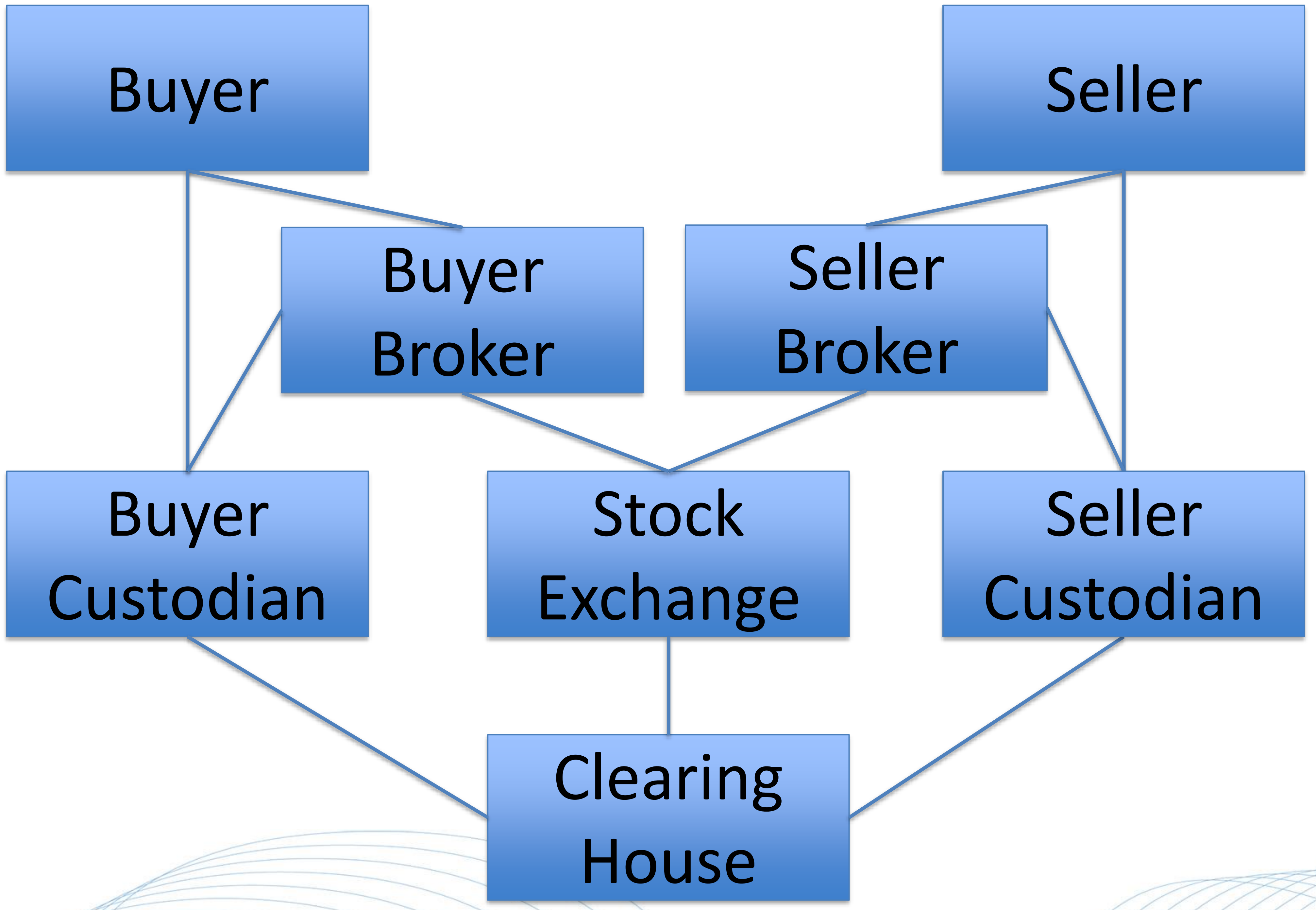
RAISE Robotic System for Air Quality Assessment in Industrial Environments

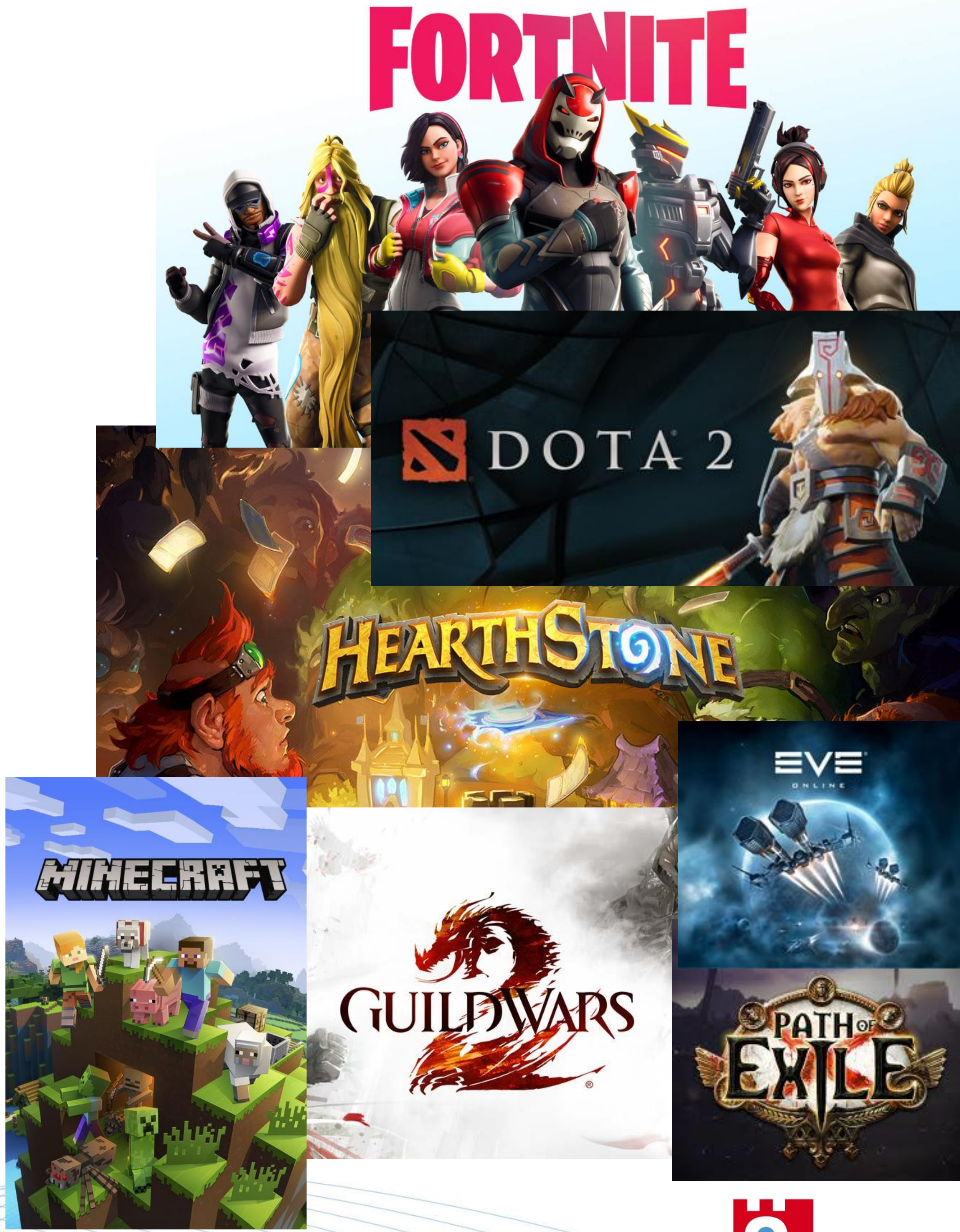

Additional sensors for the campaign
Robot in action
Static sensor node in minimal configuration
Personal measurements during the campaign
Operation environment
3D map of the foundry with a pollution overlay created with the robot



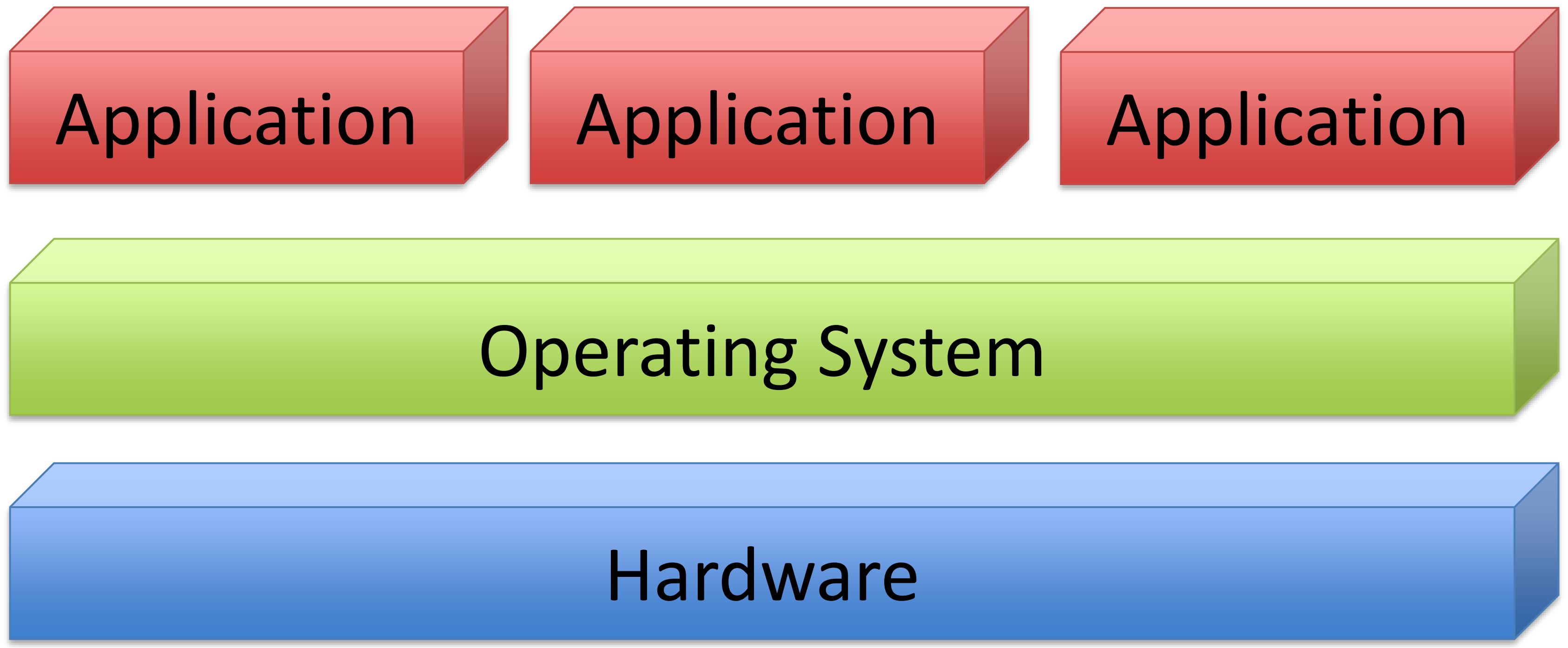


More examples





In the beginning ...



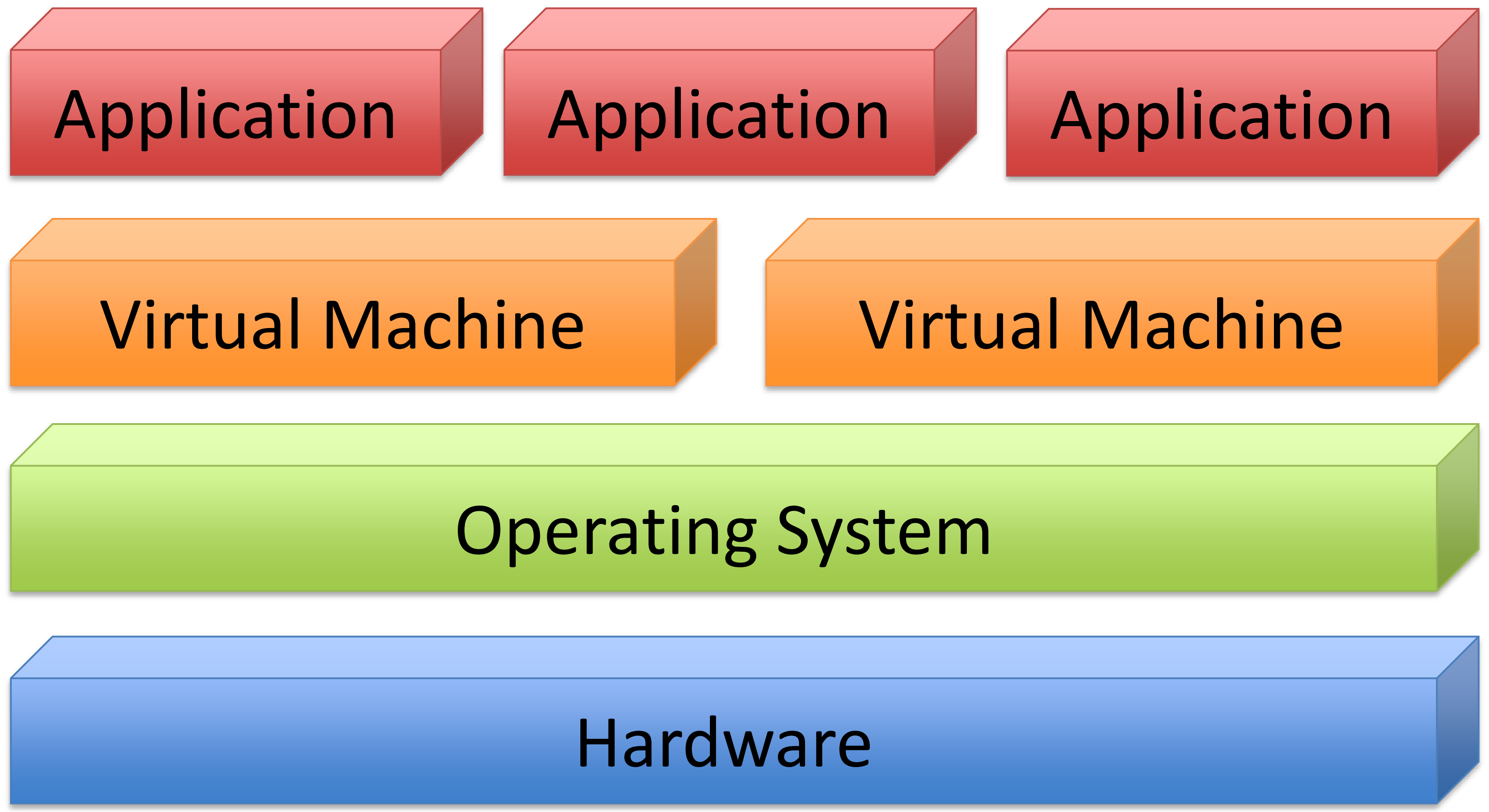
Virtualization of

- CPU, memory, I/O system, file system

Interprocess Communication (IPC)

Task: If you need a refresher on the concepts mentioned here from realtime programming, you can start with Chapter 3.1 + 3.2 of the course book!

... Yesterday ...

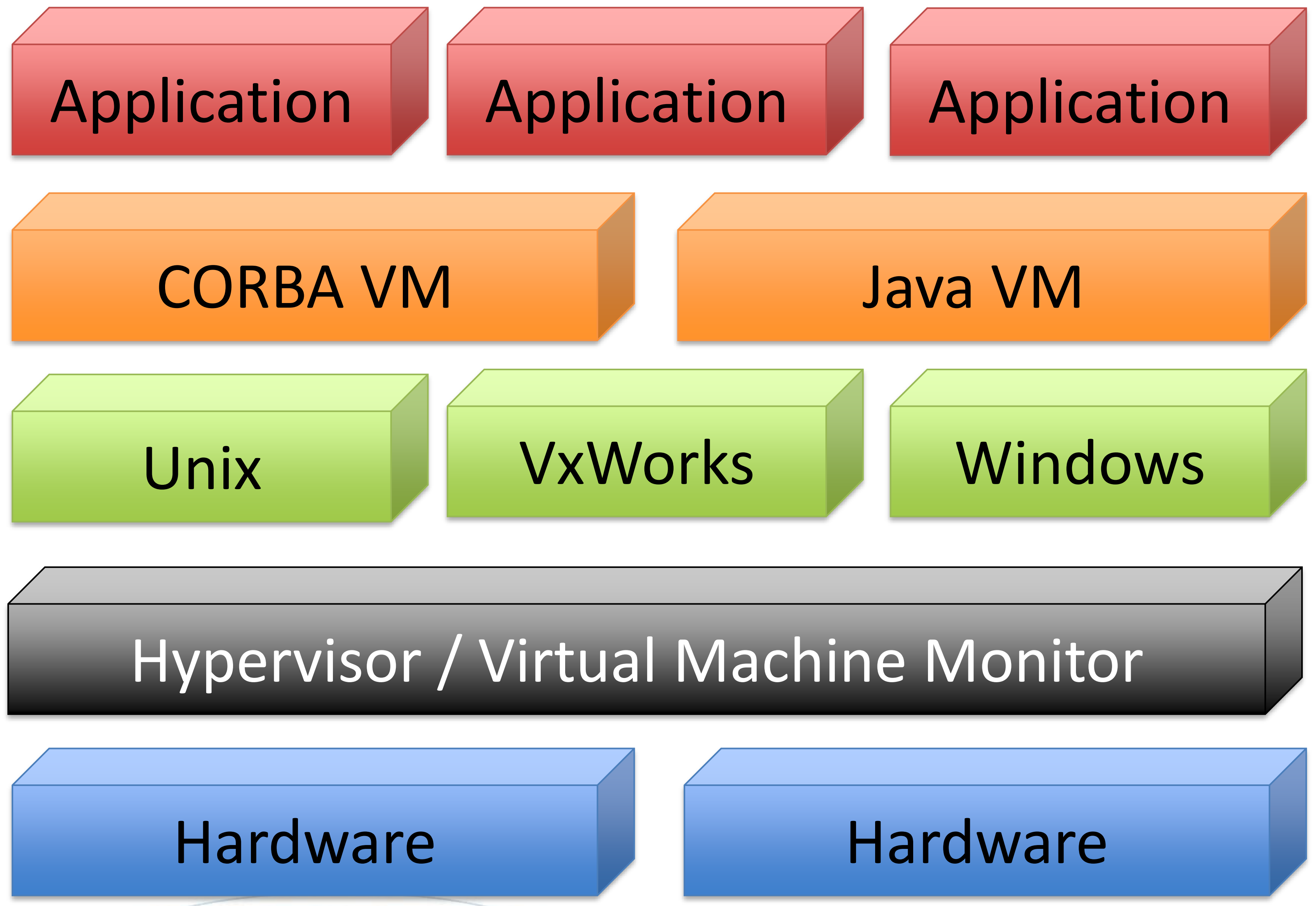


Programming paradigms for distributed systems

- Client/Server
- Peer-to-Peer (P2P)
- Remote Procedure Calls (RPC)

Distributed Services

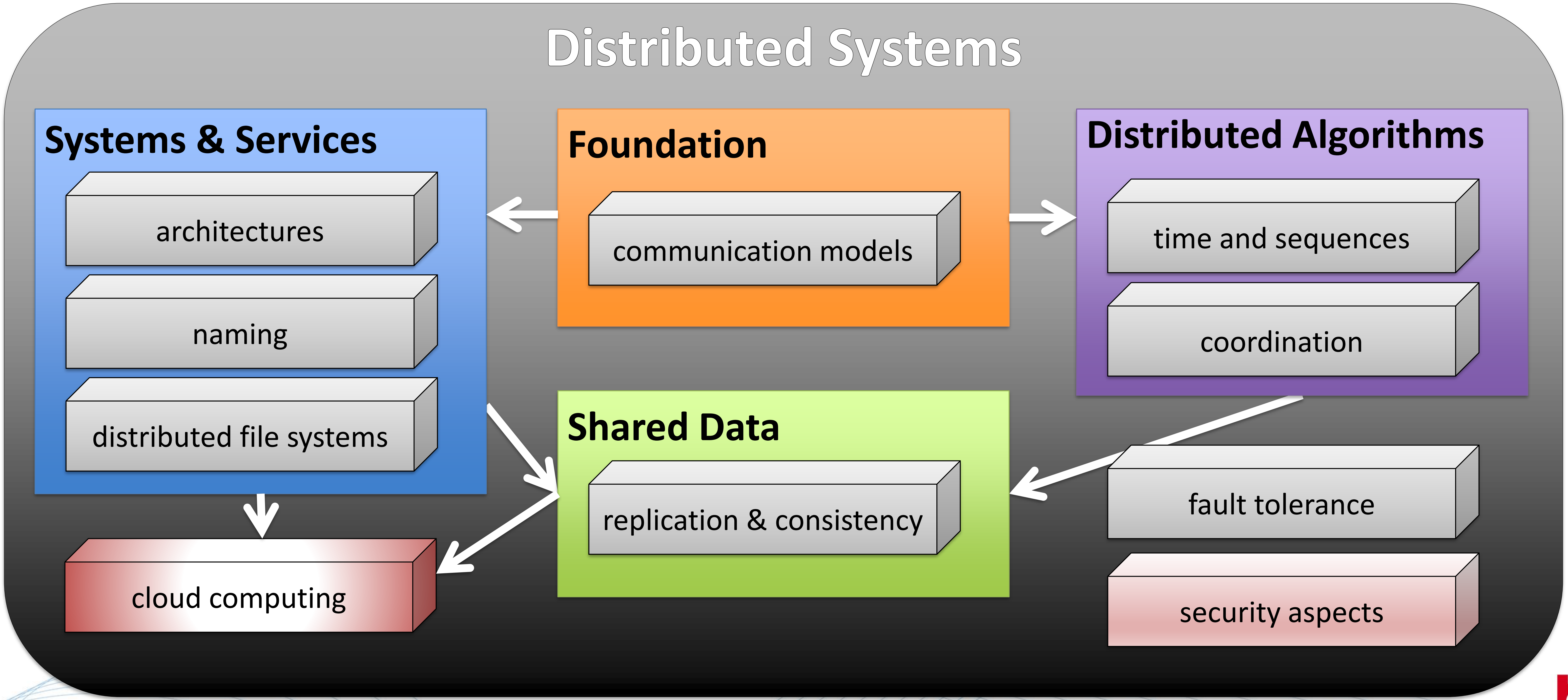
... Today



Scalable & failure resistant hardware

Choice of operating system properties

Overview of the topics covered



Learning Goals

- Ability to describe and explain important aspects of distributed systems
- Chose an appropriate design for problems involving distributed components
- Be aware of the pitfalls and challenges inherent to designing and operating distributed systems
- Get your hands dirty with some distributed systems programming

Distributed systems: A closer look

A distributed system is a collection of **autonomous computing elements** that appears to its users as a single coherent system.

From Steen, Tanenbaum. Distributed Systems (2017), p.2

Autonomous nodes

Independent behaviour & failures

➔ individual notion of time

➔ sychronization and coordination needs

Distributed systems: A closer look

A distributed system is **a collection** of autonomous computing elements that appears to its users as a single coherent system.

From Steen, Tanenbaum. Distributed Systems (2017), p.2

Collection

Membership & Communication in the collection

➔ Open vs. Closed groups

➔ Authentification of members & trust

Distributed systems: A closer look

A distributed system is a collection of autonomous computing elements that **appears** to its users **as a single coherent system**.

From Steen, Tanenbaum. Distributed Systems (2017), p.2

Hidden from the end user is

- where the computation is taking place
- where the data is stored and whether it was replicated

Desired properties of distributed systems

1. Resource sharing
2. Distribution transparency
3. Openness
4. Scalabilty
5. Redundancy

1) Resource sharing

Resource (data storage, computational resources, sensors, etc.) available on demand provided by specialists, instead of maintaining local infrastructure

- cloud storage of data (e.g. Dropbox)
- P2P streams and torrents
- shared web hosting (content distribution/delivery networks)

2) Distribution transparency

Distributed systems should operate independent of

- HW & OS specifics of nodes → **Access** transparency
- physical location of nodes → **(Re-)Location & Migration** transparency
- use of copies → **Replication** transparency
- multiple users per resource → **Concurrency** transparency
- partial system failures → **Failure** transparency

2) Distribution transparency (2)

Full distribution transparency is not possible to achieve, due to

- Laws of physics → latencies in communication
  (Location, Migration & Replication)
- Impossibility to distinguish a failing node from a slow node
  (Failure)
- Performance costs
  (Replication, Concurrency)

2) Distribution transparency (3)

Distribution transparency might even not be desirable

- Location-based services
- Users in different time zones
- Understanding system behaviour in partial system failures

3) Openness

Open system: can be used by or integrated into other systems

- Interfaces
- Interoperability
- Portabilty
- Extensibility

4) Scalability – Size

Size scalability refers to the number of users and/or processes

Problems are caused by

- Computational capacity
- Storage and data transfer capacity
- Network capacity

Solution: Buy / rent more hardware!

TASK: Analyzing service capacity (p. 16, Note 1.6)

Assuming a service capacity of $\mu=17$ requests per second and an average arrival rate of $\lambda=\{7,12,15,16\}$ requests per second calculate the expected response times!

Does a Denial-of-Service attack require $\lambda>=\mu$? Why (not)?

4) Scalability – Geography

Does the distributed system scale from local area networks (LAN) to wide area systems (WAN)?

- Latency in client-server interactions
- Different levels of reliability of the networks
- Different communication possibilities (e.g. broadcasts)

Possible solution:
Asynchronous communication
Replication & Caching, but creates consistency issues

4) Scalability – Administration

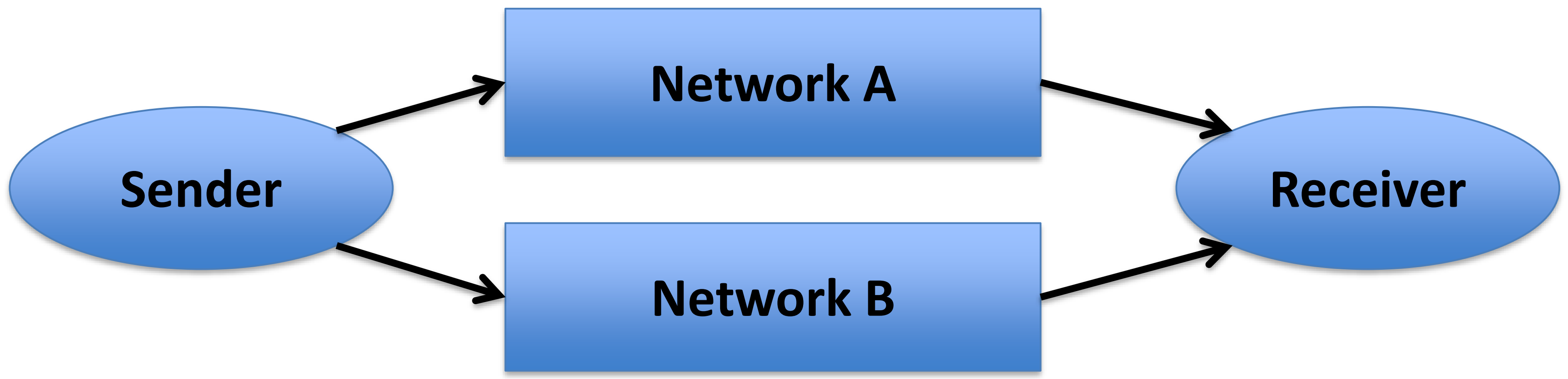
Distributed systems may involve components managed by different parties (e.g. ISPs, cloud service providers, specialized hardware)

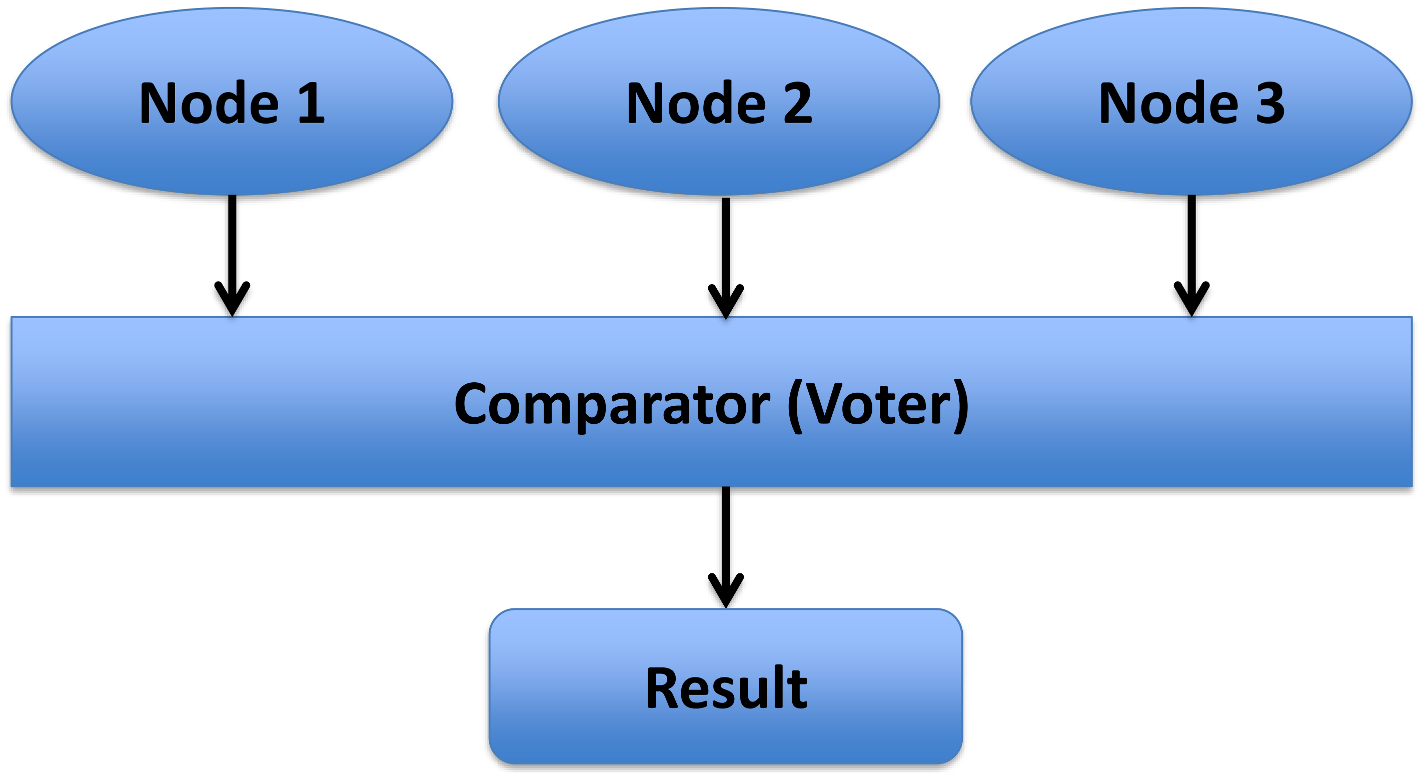
- Usage policies (e.g. restrictions for certain types of traffic)
- Security issues for the different parties (e.g. access to computational resources or storage)

Possible solution:
P2P systems (e.g. torrents, P2P telephony, P2P streaming) try to avoid these particular scaling issues

5) Redundacies

Goal: Creating tolerance towards **errors**, **faults** and **failures**





Common pitfalls

- The network is reliable
- The network is secure
- The network is homogeneous
- The topology does not change
- Latency is zero
- Bandwidth is infinite
- Transport cost is zero
- There is one administrator

Core message of the introduction

Distributed systems allow

- Resource sharing
  → economic efficiency
- Parallel processes
  → perfomance
- Redundancies
  → availibility, fault resistance
- Adaptivity
  → scalable, open

But this doesn’t come for free

- partial failures
- heterogenity
- realtime behaviour
- distributed algorithms
- IT security