Dependable Distributed Systems
Master of Science in Engineering in Computer Science

AA 2023/2024

INTRODUCTION

General Information

- 9 CFU all in the first semester
  - September 25th– December 22nd

- Instructors
  - Silvia Bonomi
  - Giovanni Farina

General Information

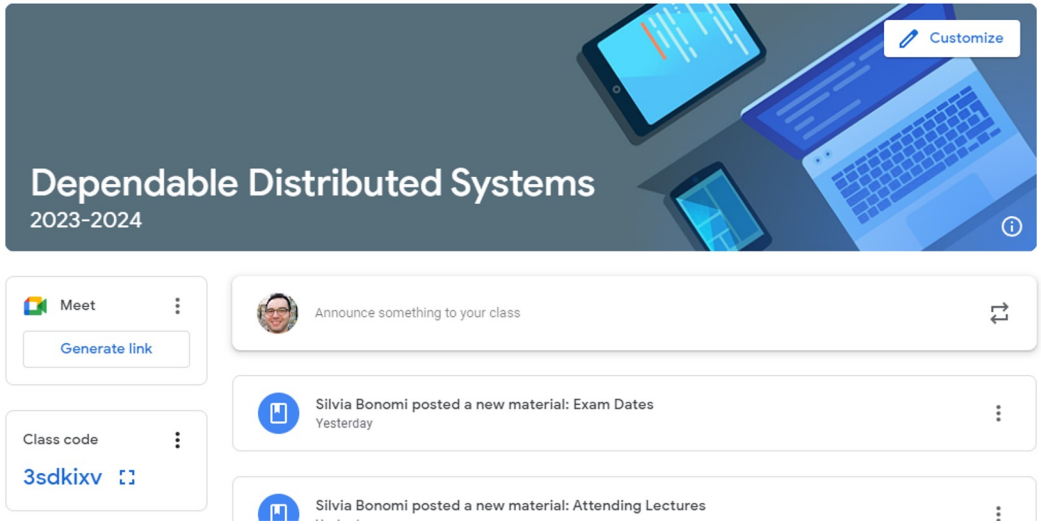
Schedule

- **Tuesday 12:00 - 14:00** Room 29 San Pietro in Vincoli, Via Eudossiana 18
- **Wednesday 10:00 - 14:00** Room 29 San Pietro in Vincoli, Via Eudossiana 18
- **Thursday 12:00 - 14:00** Room 29 San Pietro in Vincoli, Via Eudossiana 18

General Information

Course Material

- all the material will be made available on Google Classroom
- Classroom code: **3sdkixv**

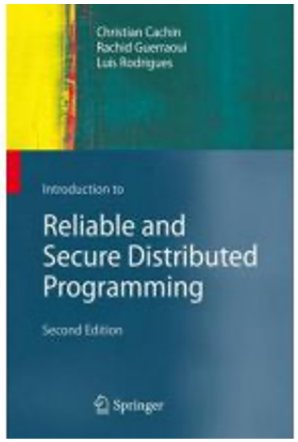


General Information

Material

- Main Textbooks
  - C. Cachin, R. Guerraoui and L. Rodrigues. Introduction to Reliable and Secure Distributed Programming, Springer, 2011

    (Available for reference at the DIAG library)



- Scientific papers
- Supporting Slides

General Information

Students' hours

- Asking for an appointment by sending an email
  - The appointment could be either face to face or remote and it will be in the week following your request
  - NOTE: use a proper object to write your email
  - NOTE2: reply-all button does not bite!

- For brief questions, when you are in presence
  - At the beginning/end of every lecture
  - During breaks

Where we are

- DIAG, Via Ariosto 25
  - Silvia Bonomi: Room B114, 1st floor, B wing
  - Giovanni Farina: Room B113, 1st floor, B wing

General Information

Exam

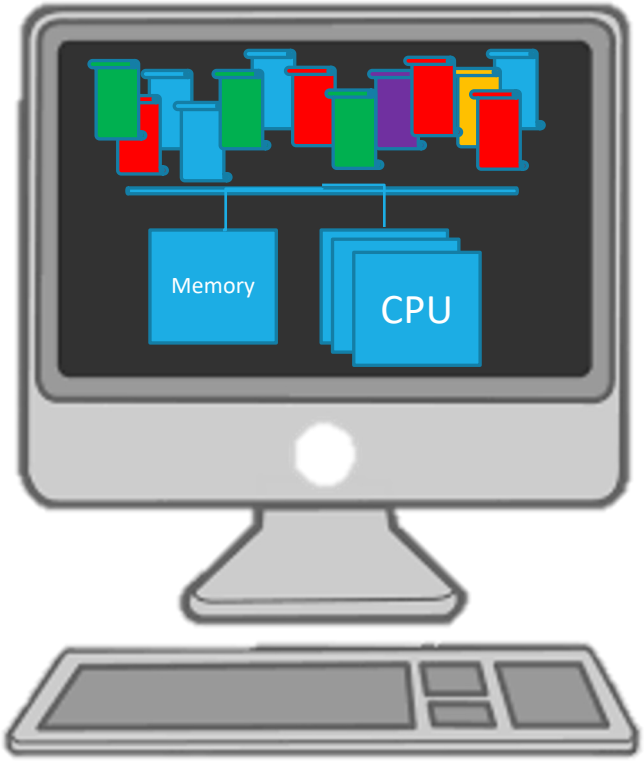
- Mandatory part
  - The exam is made of a written test
  - Questions may cover any topic contained in the final syllabus

- Optional part
  - Project assignment to develop in groups of 2-3 persons
  - **General rules:**
    - The project will be evaluated with a score between 0 and 3 that will be added to the mark of your written test
    - All projects must be requested, assigned, and set by the deadline (later defined, around end of November), they cannot be requested or modified later
    - The project can be delivered in any exam session until September (October/November) 2024
    - Further details will be available later in the course

Distributed Systems

From concurrent to distributed systems

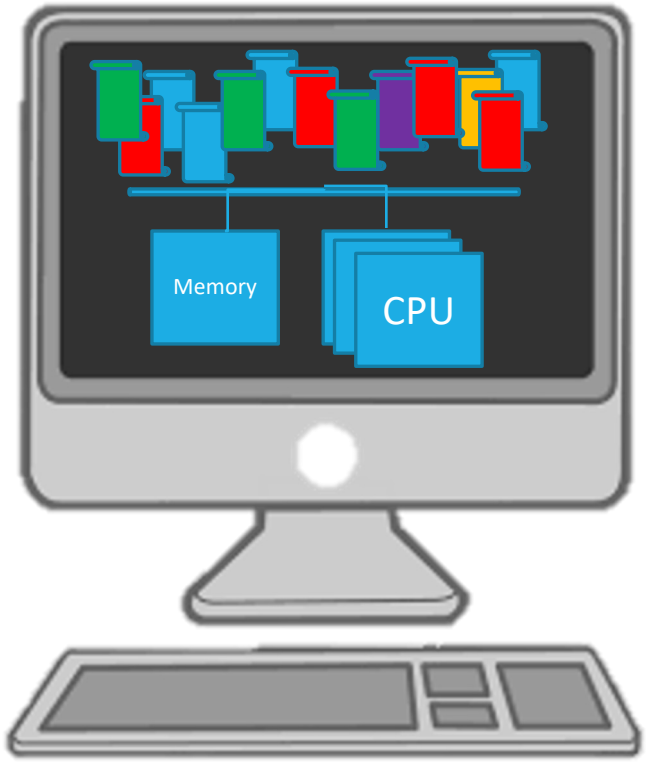
CONCURRENT SYSTEMS



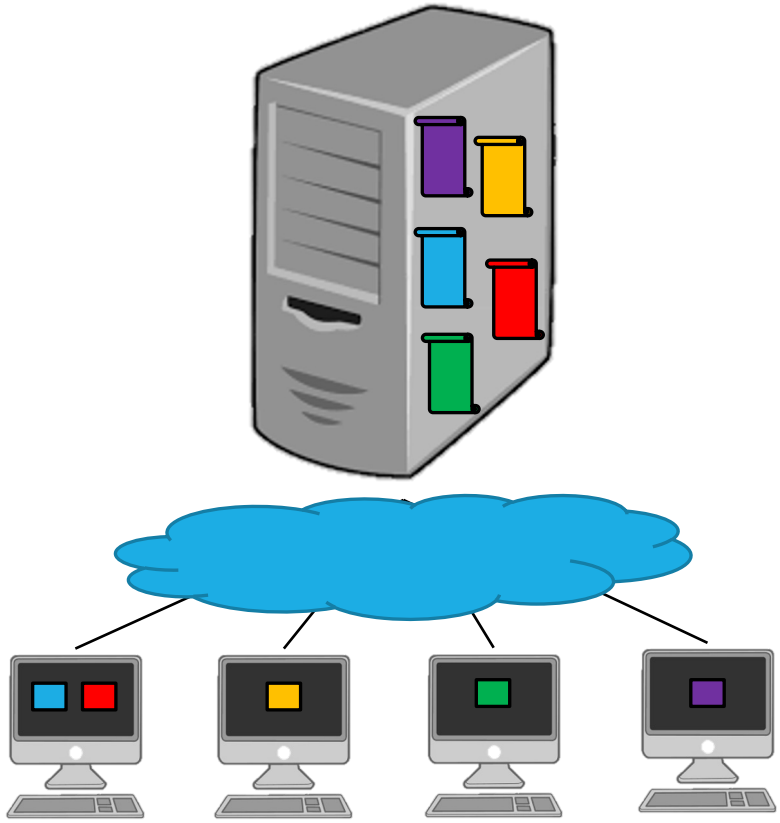
DISTRIBUTED SYSTEMS

From concurrent to distributed systems

CONCURRENT SYSTEMS

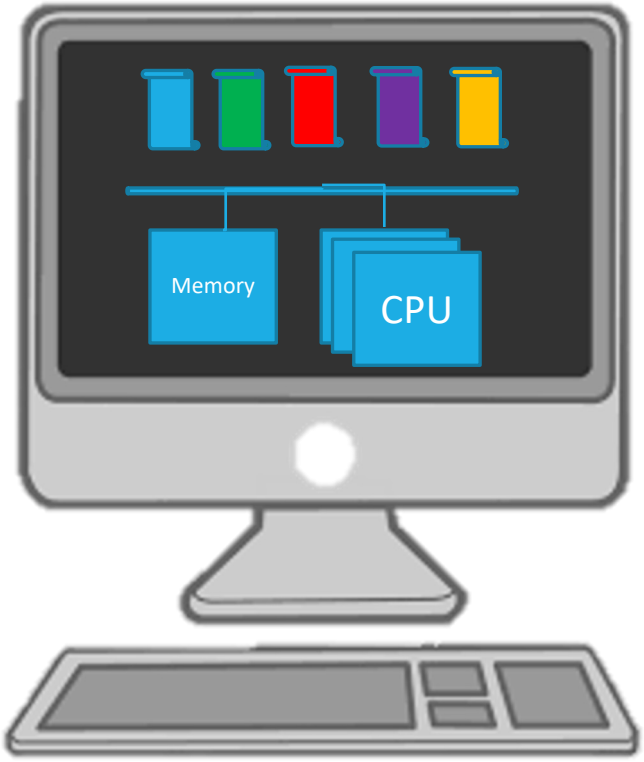


DISTRIBUTED SYSTEMS

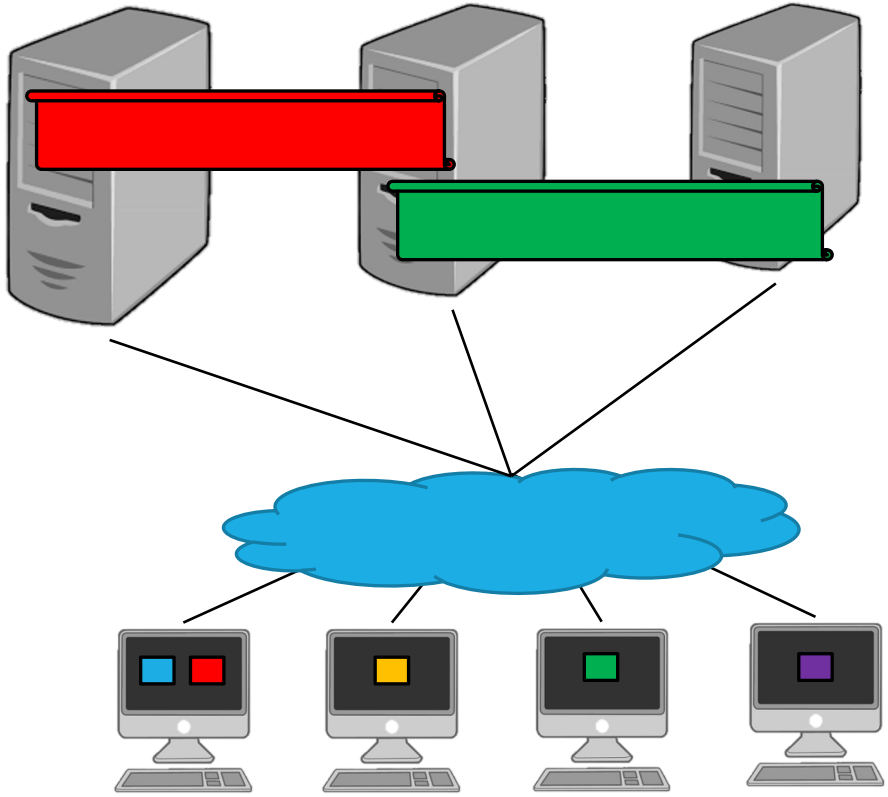


From concurrent to distributed systems

CONCURRENT SYSTEMS



DISTRIBUTED SYSTEMS



Definitions

A distributed system is a set of spatially separate entities, each of these with a certain computational power that are able to communicate and to coordinate among themselves for reaching a common goal

A distributed system consists of a collection of autonomous computers, connected through a network and distribution middleware, which enables computers to coordinate their activities and to share the resources of the system, so that users perceive the system as a single, integrated computing facility
(Wolfgang Emmerich)

A distributed system is a piece of software that ensures that a collection of independent computers appear to its users as a single coherent system
(Maarten van Steen)

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)

Common points across definitions

- Set of entities/computes/machines
- Communication, coordination, resource sharing
- Common Goal
- Appear as a single computing system

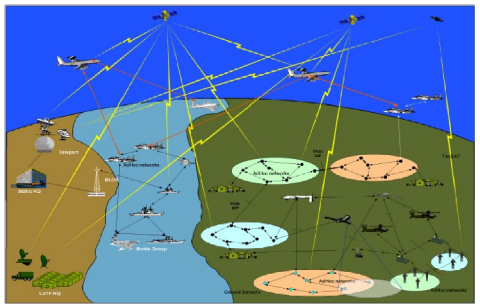
Why Distributed Systems?

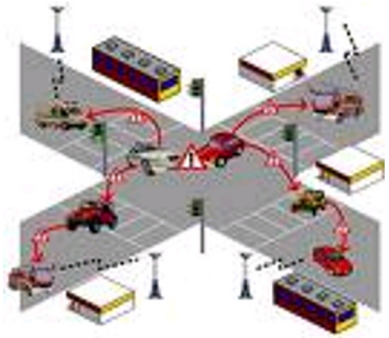
1. To Increase Performance
   - To cope with the increasing demand of users in both processing power and data storage
   - To reduce latency (users are spread all over the world and you want to provide them best user experience by reducing the response time)

2. To Build Dependable Services
   - To cope with failures

Distributed Systems: examples

Internet





Distributed Systems

CHARACTERISTIC 1: Collection of autonomous computing elements

CHALLENGES

- absence of a global clock
- group membership
  - open groups vs closed groups
- overlay network
  - structured vs unstructured

Distributed Systems

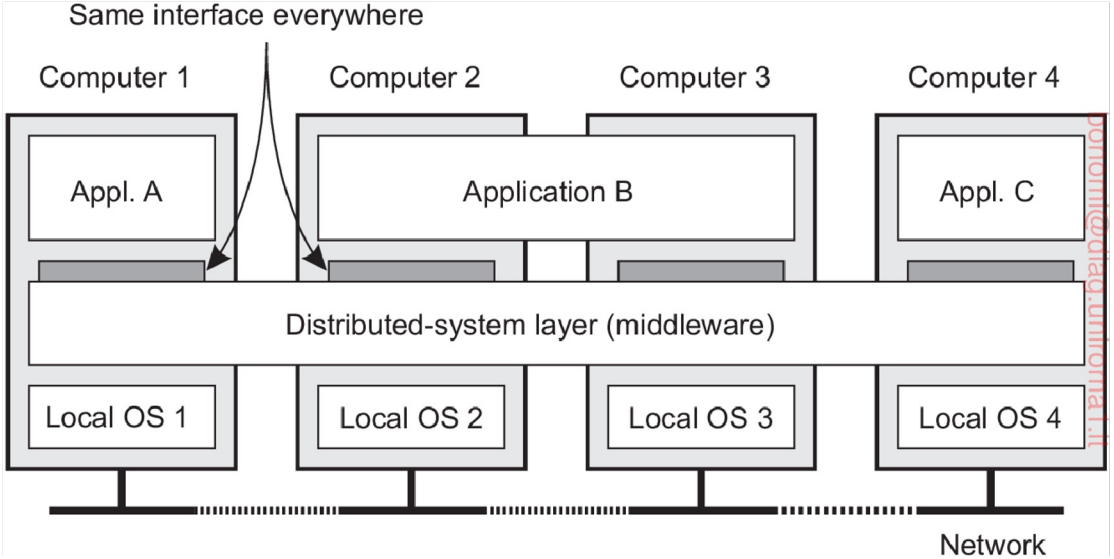
CHARACTERISTIC 2: Single coherent system

- a distributed system is coherent if it behaves according to the expectations of its users
  - the collection of nodes as a whole operates the same, no matter where, when, and how interaction between a user and the system takes place

CHALLENGE

- Failures

Middleware and distributed systems



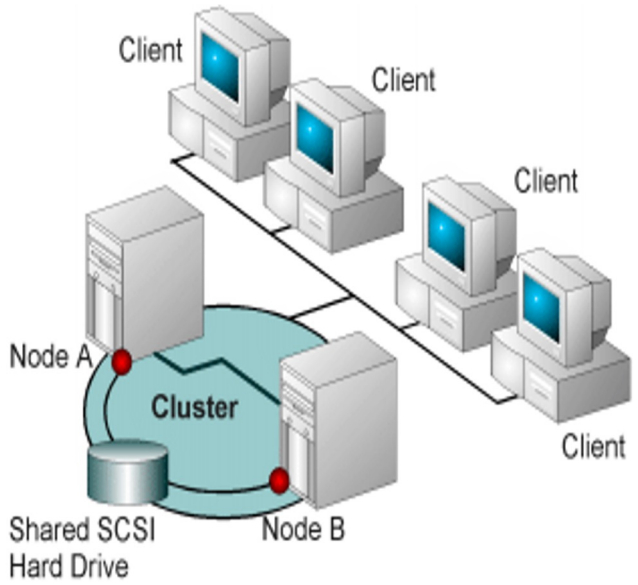
The distributed system

- Allows inter-process communication
  - to components of a single distributed application to communicate with each other
  - to different applications
- hides, as best and reasonably as possible, the differences in hardware and operating systems from each application

Primary Goal: Overcome the limitation of centralized environment

PROBLEMS

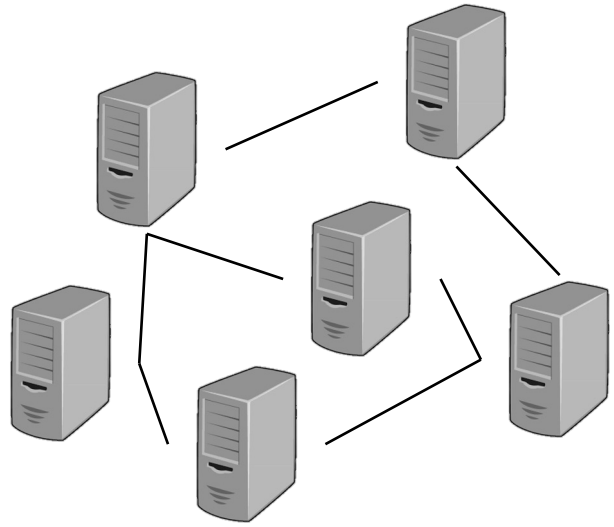
- Connectivity and Communication
- Synchronization
- Coordination



Primary Goal: Overcome the limitation of centralized environment

Coordination has to be implemented taking into account the following conditions that deviate from centralized systems:

1. Temporal and spatial concurrency
2. No global Clock
3. Failures
4. Unpredictable latencies



These limitations restrict the set of coordination problems we can be solve in a distributed setting

Trends in Distributed Systems

Distributed systems are undergoing a period of significant change, and this can be traced back to several influential trends:

- the emergence of pervasive networking technology
- the emergence of ubiquitous computing coupled with the desire to support user mobility in distributed systems
- the increasing demand for multimedia services
- the view of distributed systems as a utility

Pervasive networking and the modern Internet

CHARACTERISTICS

- Scale
- Heterogeneity in
  - Devices (e.g. servers, workstations, tiny devices)
  - Communication Protocols (wired vs wireless)
  - Available services
- Absence of time and space limitation to connection requests

Figure 1.3 A typical portion of the Internet

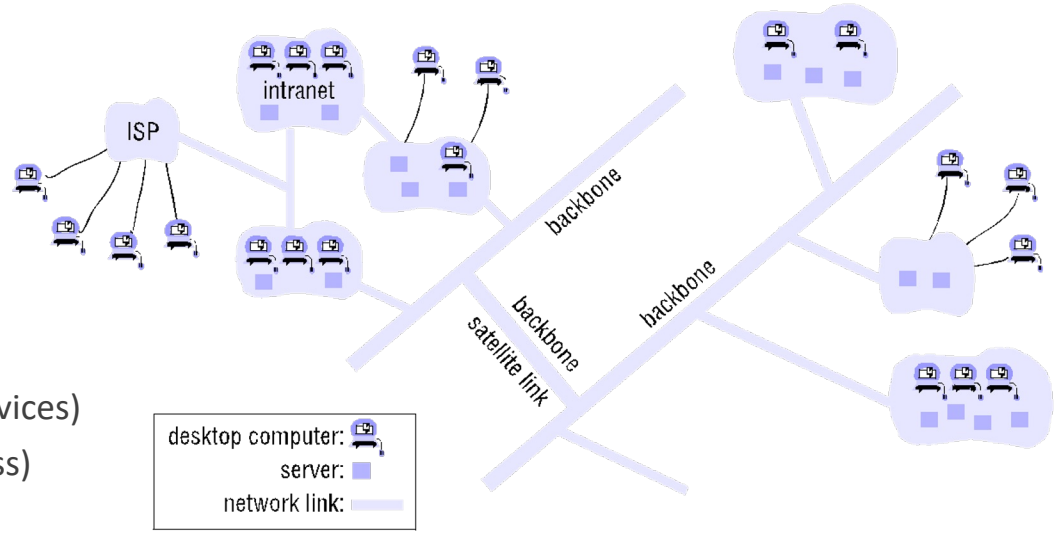


Figure taken from Couloris, Dollimore, Kindberg, Blair *Distributed Systems: Concepts and Design*, 5th Edition

Mobile and Ubiquitous Computing

Mobile computing is the performance of computing tasks while the user is on the move, or visiting places other than their usual environment

Ubiquitous computing is the harnessing of many small, cheap computational devices that are present in users' physical environments, including the home, office and even natural settings

COMMON PROBLEMS

- System scale
- Dynamicity in the system
- Heterogeneity of participants
- Security Issues

Distributed Multimedia Systems

Multimedia support is the ability of a system to support a range of media types in an integrated manner

- It should be able to perform the same functions for continuous media types such as audio and video

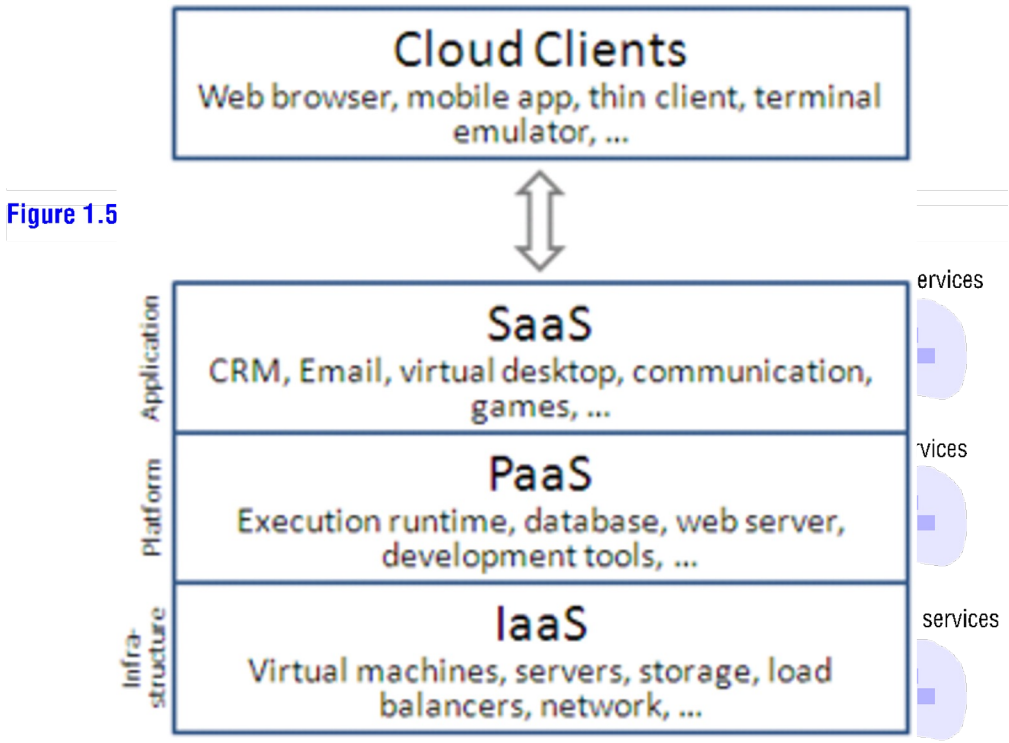
CHALLENGES

- Temporal dimension
- Quality of Service is a strong requirement

Distributed computing as a utility

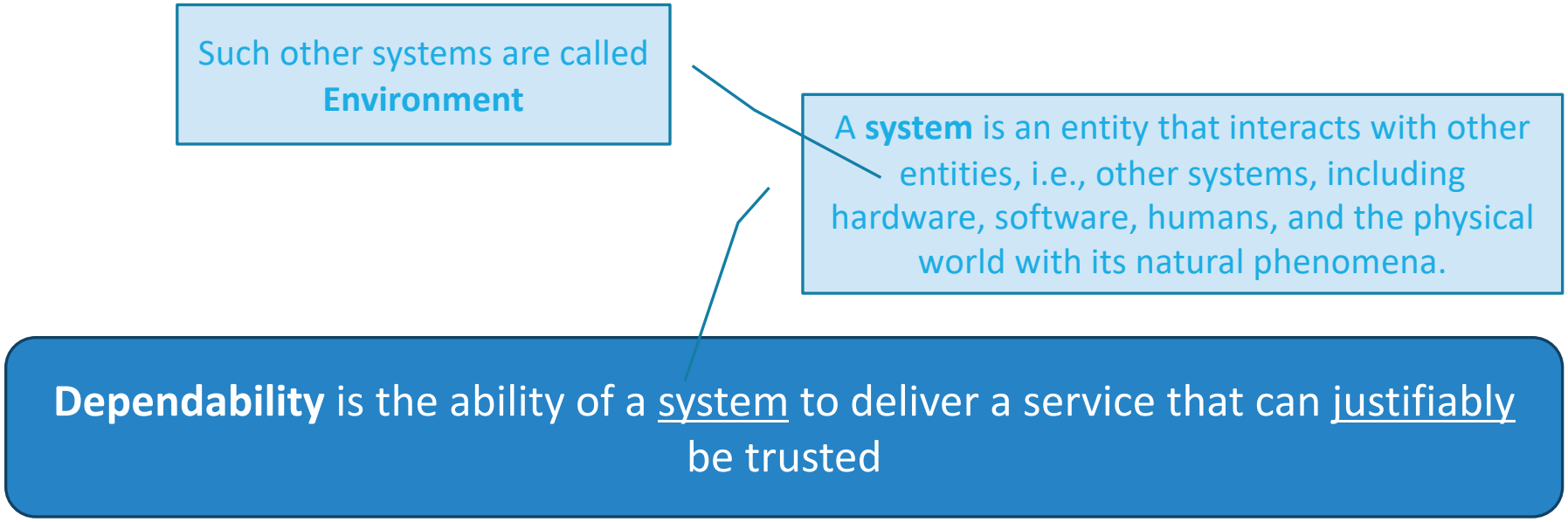
CLOUD COMPUTING

- IaaS
- PaaS
- SaaS



Dependability

Dependability Definition



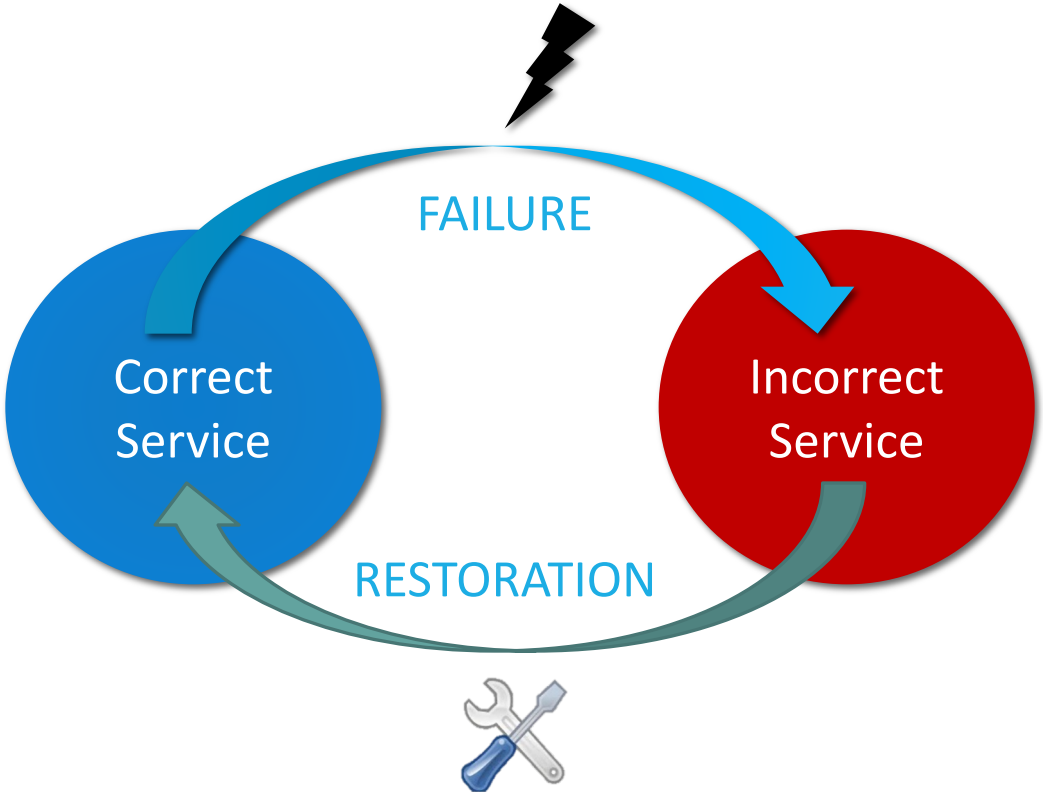
An Alternative Dependability Definition

Dependability *is the ability of a system to avoid service failures that are more frequent and more severe than is acceptable*

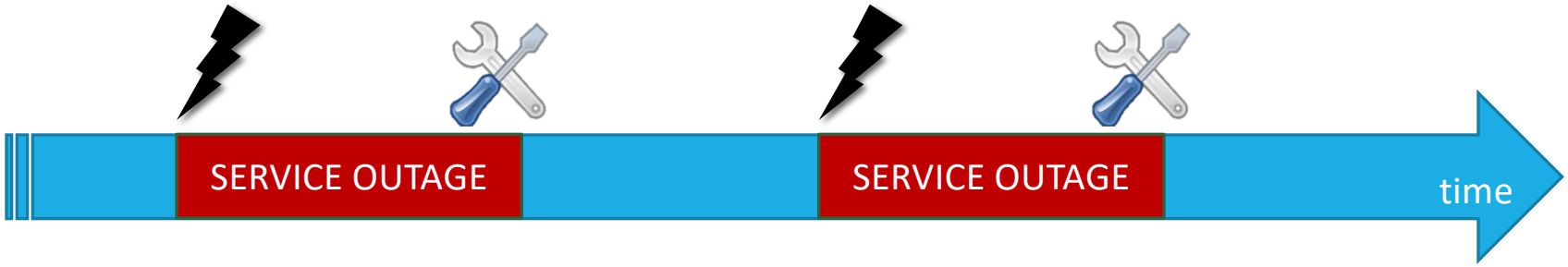
A **service failure** (or simply failure) is an event that occurs when the delivered service deviates from correct service

- A **correct** service is delivered when the service implements its functional specification in terms of
  - functionality
  - performance

Service Failure



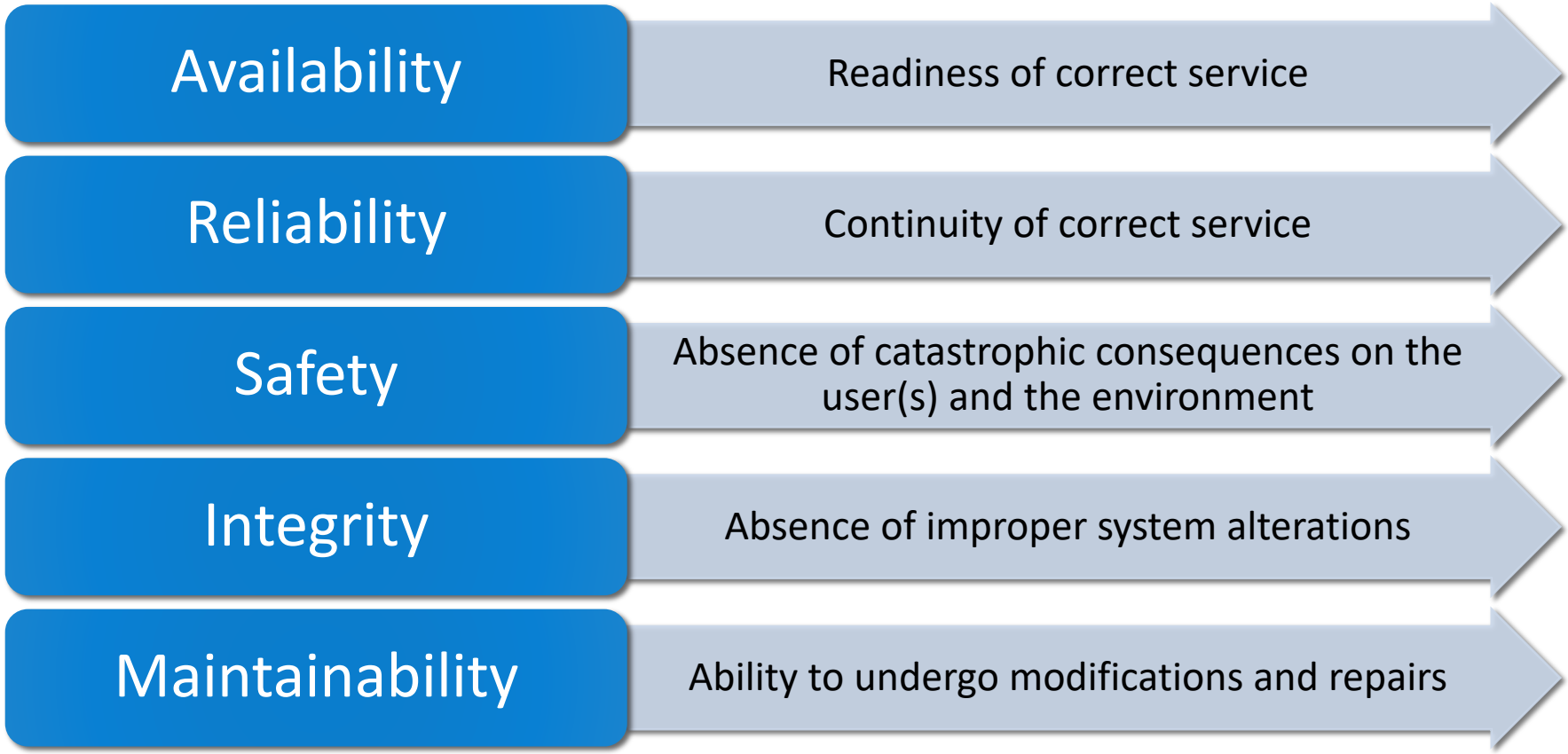
Service Failure



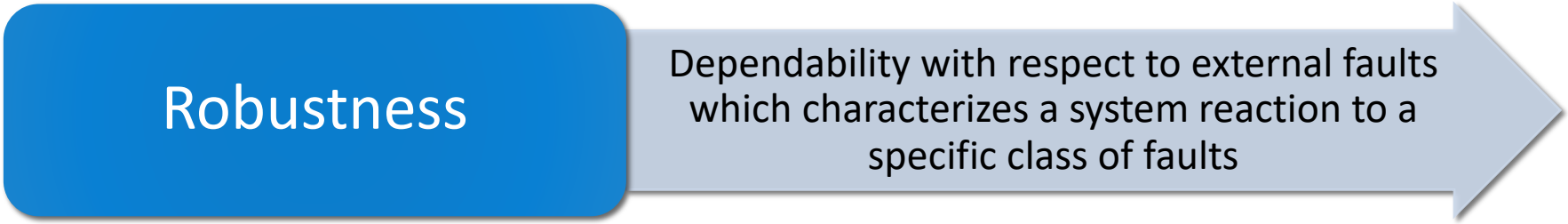
PROBLEM

How to design, develop and deploy a system that is dependable and secure?

Dependability Attributes (or requirements)



Secondary Dependability Attributes



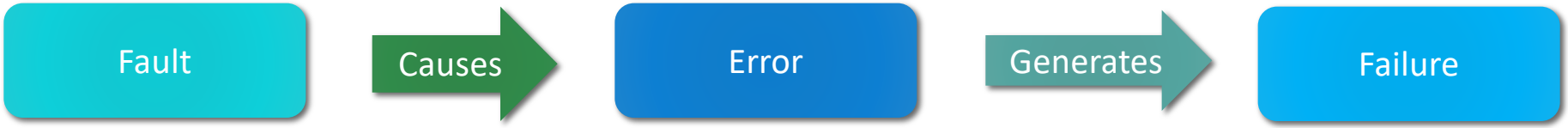
Failures, Errors and Faults

OBSERVATION: having a service failure means that there exists at least a deviation of the system behaviour from the correct service state.

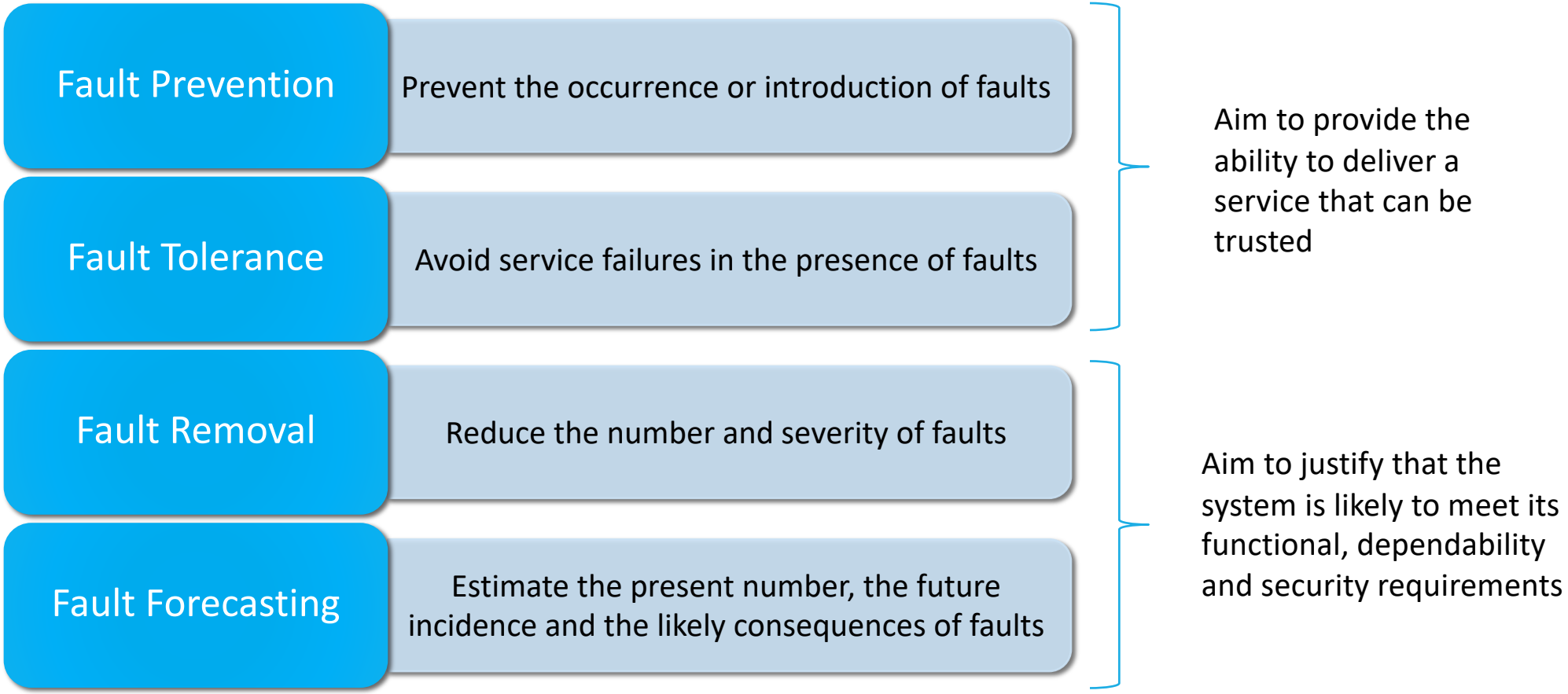
The deviation from the correct state is called an **error**

The adjudged or hypothesized cause of an error is called a **fault**

- A Fault can be internal or external of a system

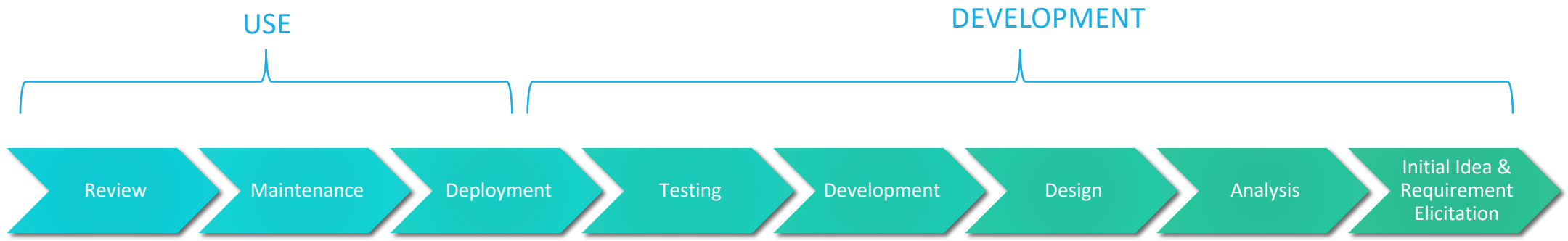


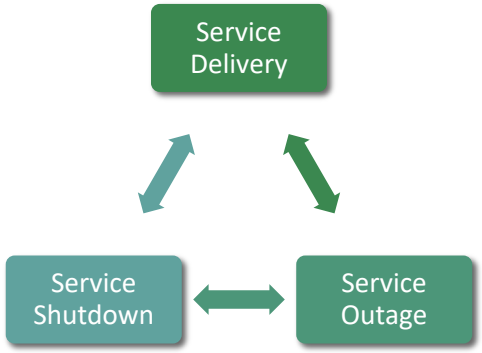
The Means to Attain Dependability



Threats to Dependability during its life cycle

THE LIFE CYCLE OF A SYSTEM CONSISTS OF TWO MAIN PHASES





DEVELOPMENT FAULTS MAY BE INTRODUCED INTO THE SYSTEM BY THE ENVIRONMENT

- **i.e., physical world, human developers, development tools, production and test facilities.**

Dependable Distributed Systems Characteristics and Challenges

Heterogeneity

Heterogeneity impacts at different layers

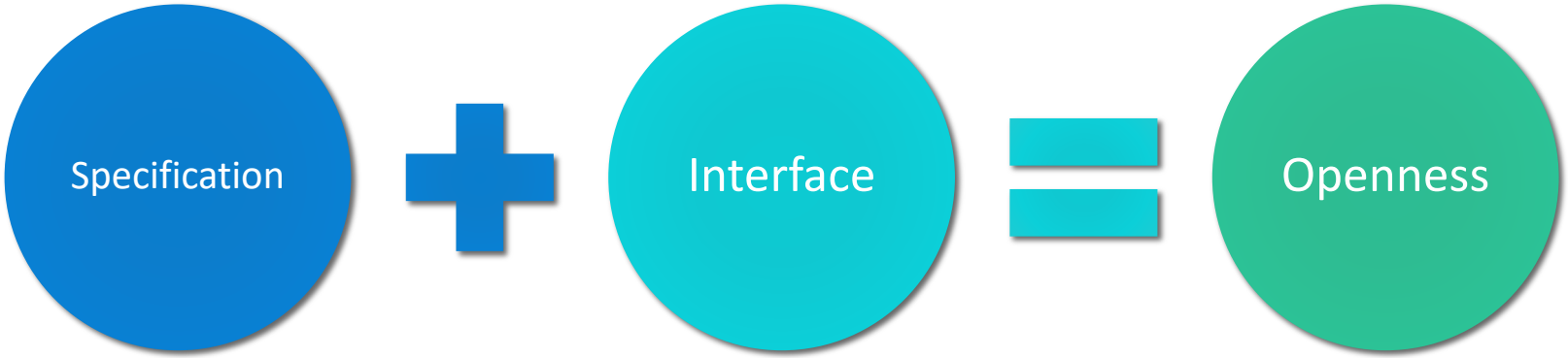
- Networks
- Hardware
- Operating Systems
- Programming Language
- Implementations from different Developers

SOLUTIONS

- Middleware (from RPC to Service oriented Architectures)
- Mobile code and Virtual Machine

Openness

Openness is the capability of a system to be extended and re-implemented



A specification of a service/component is well-formed if it is :

- *Complete* i.e., everything related to the behaviour has been specified
- *Neutral* i.e., it does not offer any detail on a possible implementation

How to work with specifications will be part of DDS

An Interface describes:

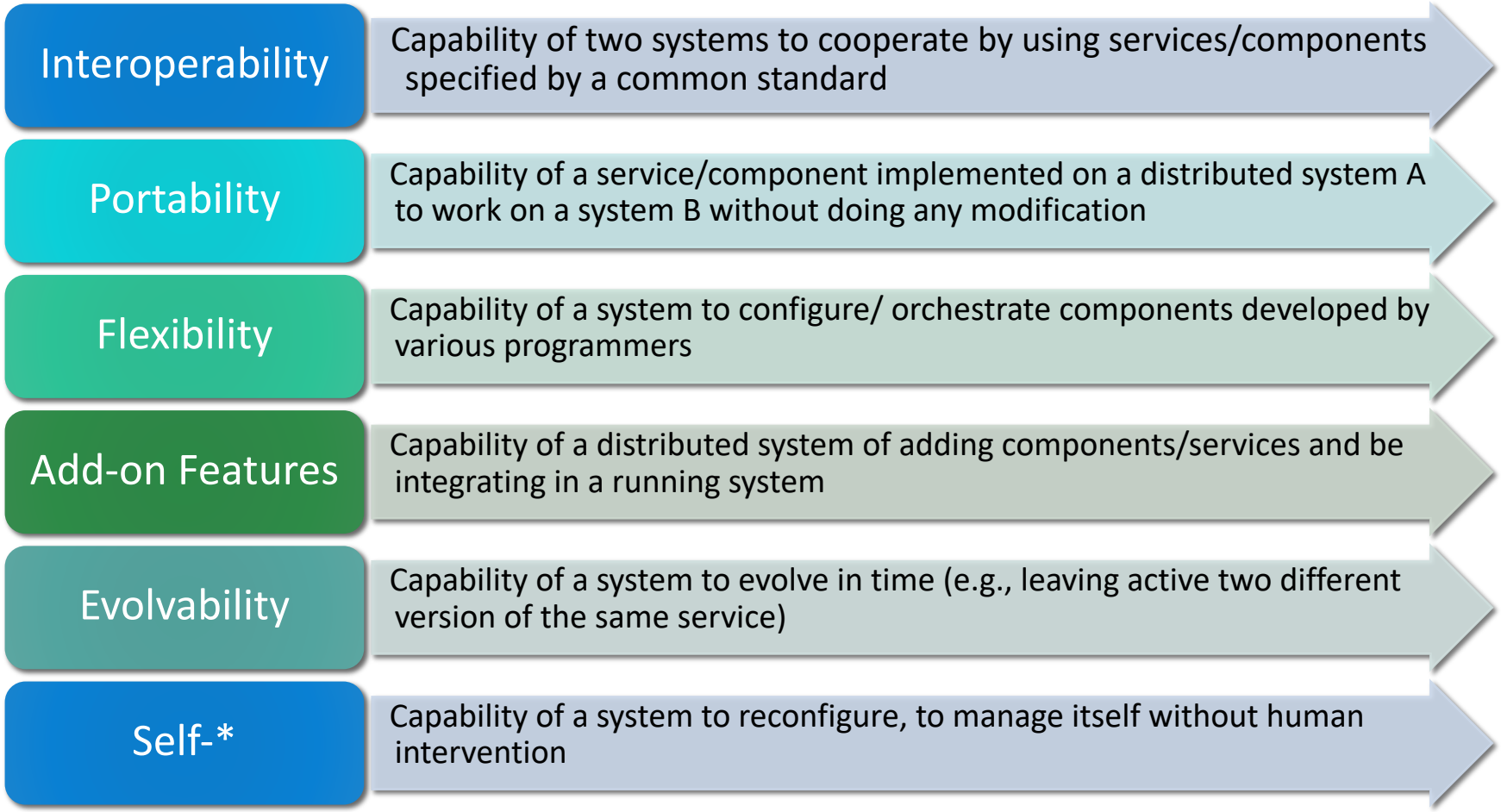
- the syntax and the semantic of a service/component
- available functions/services
- input parameters, exceptions

It can be realized through an IDL

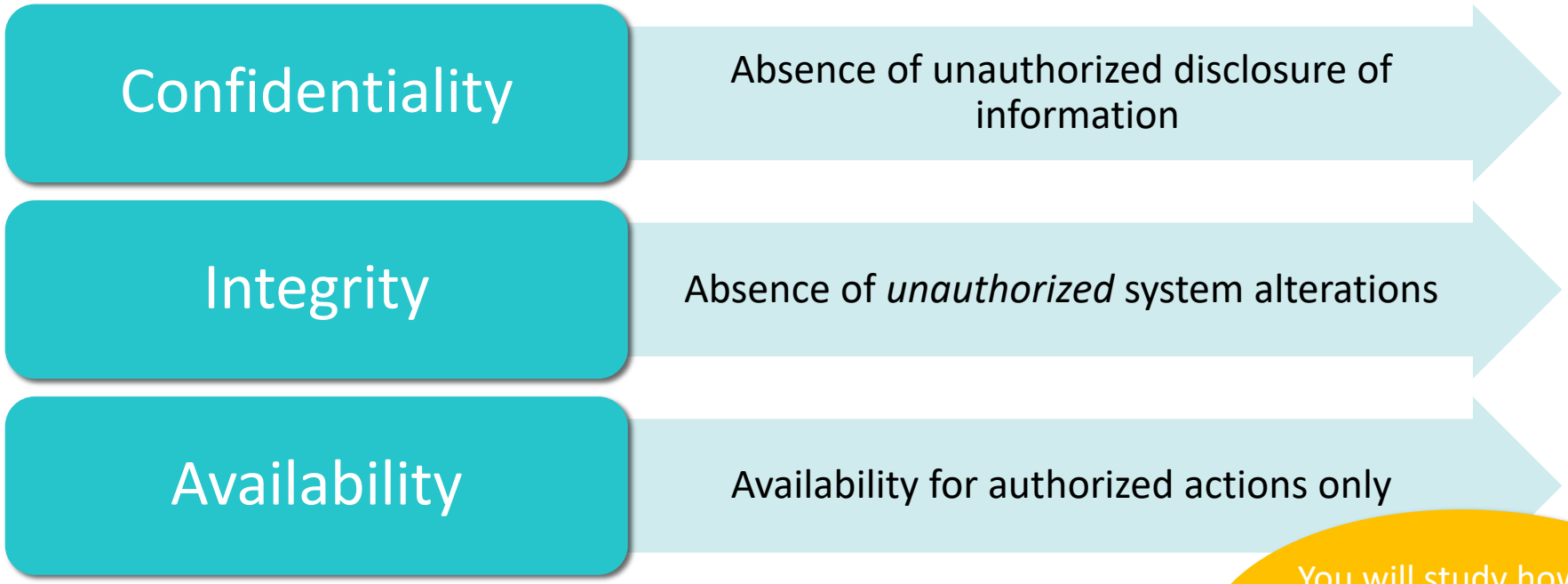
High-level interfaces will be part of DDS

Technological aspects will be Studied in Software Engineering

Openess


Interoperability
Capability of two systems to cooperate by using services/components specified by a common standard
Portability
Capability of a service/component implemented on a distributed system A to work on a system B without doing any modification
Flexibility
Capability of a system to configure/ orchestrate components developed by various programmers
Add-on Features
Capability of a distributed system of adding components/services and be integrating in a running system
Evolvability
Capability of a system to evolve in time (e.g., leaving active two different version of the same service)
Self-*
Capability of a system to reconfigure, to manage itself without human intervention

Security

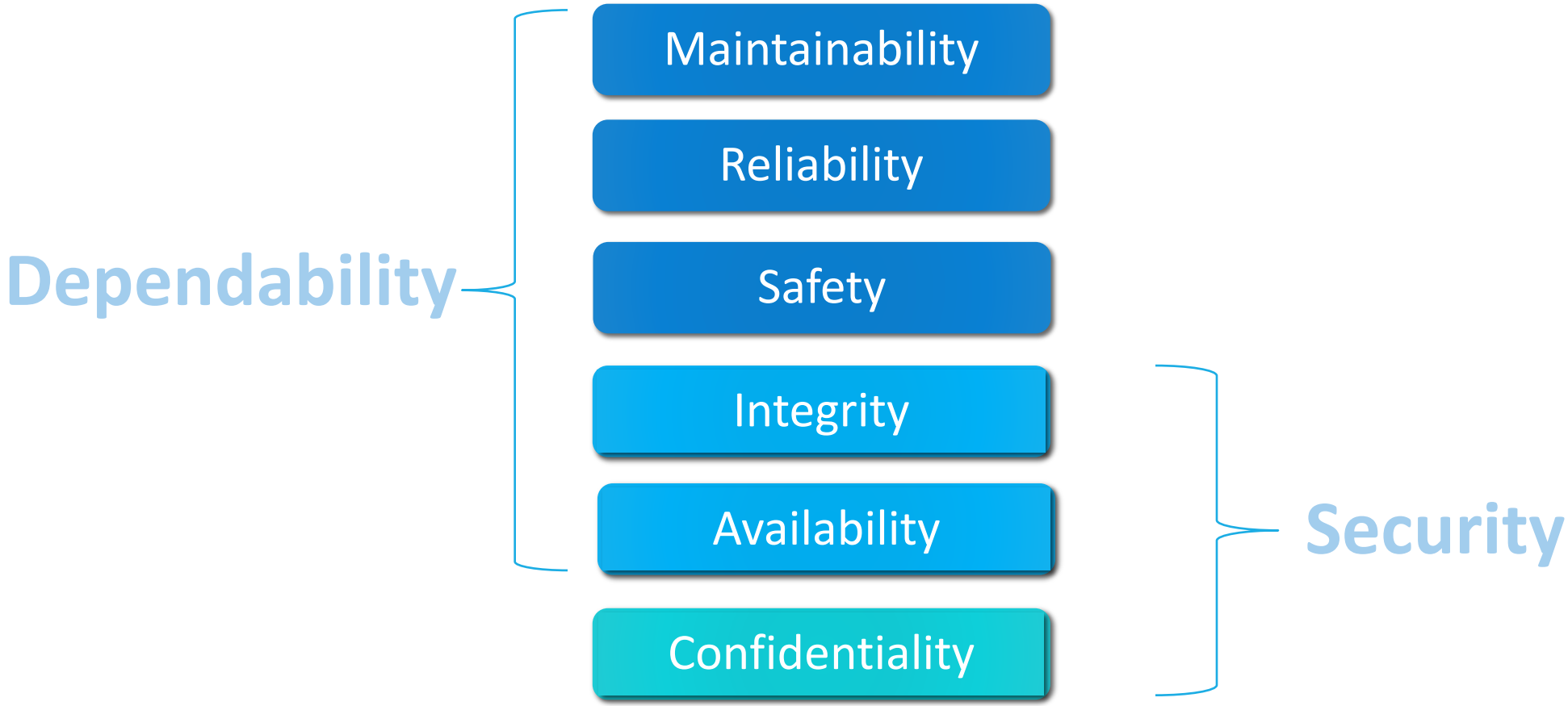

Confidentiality
Absence of unauthorized disclosure of information
Integrity
Absence of *unauthorized* system alterations
Availability
Availability for authorized actions only
DDS will look at these attributes from the design point of view
You will study how to technically manage security in Cybersecurity course

Secondary Security Attributes

Accountability	Availability and integrity of the identity of the person who performed an operation
Authenticity	Integrity of a message content and origin, and possibly of some other information, such as the time of emission
Non-repudiability	availability and integrity of the identity of the sender of a message or of the receiver

You will study how to technically manage security in Cybersecurity course

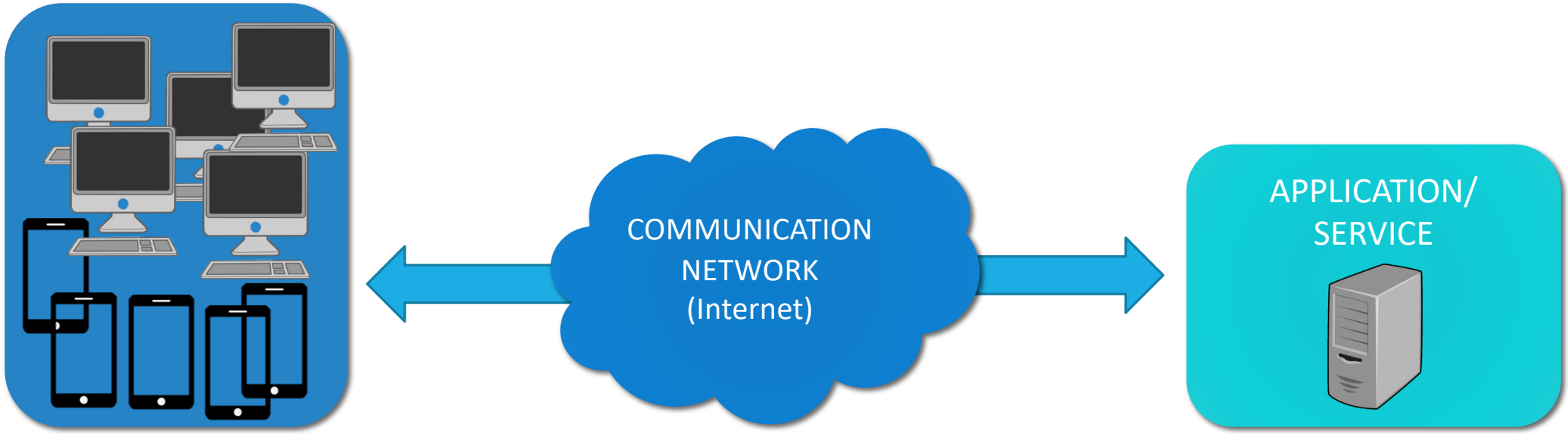
Dependability & Security



Scalability

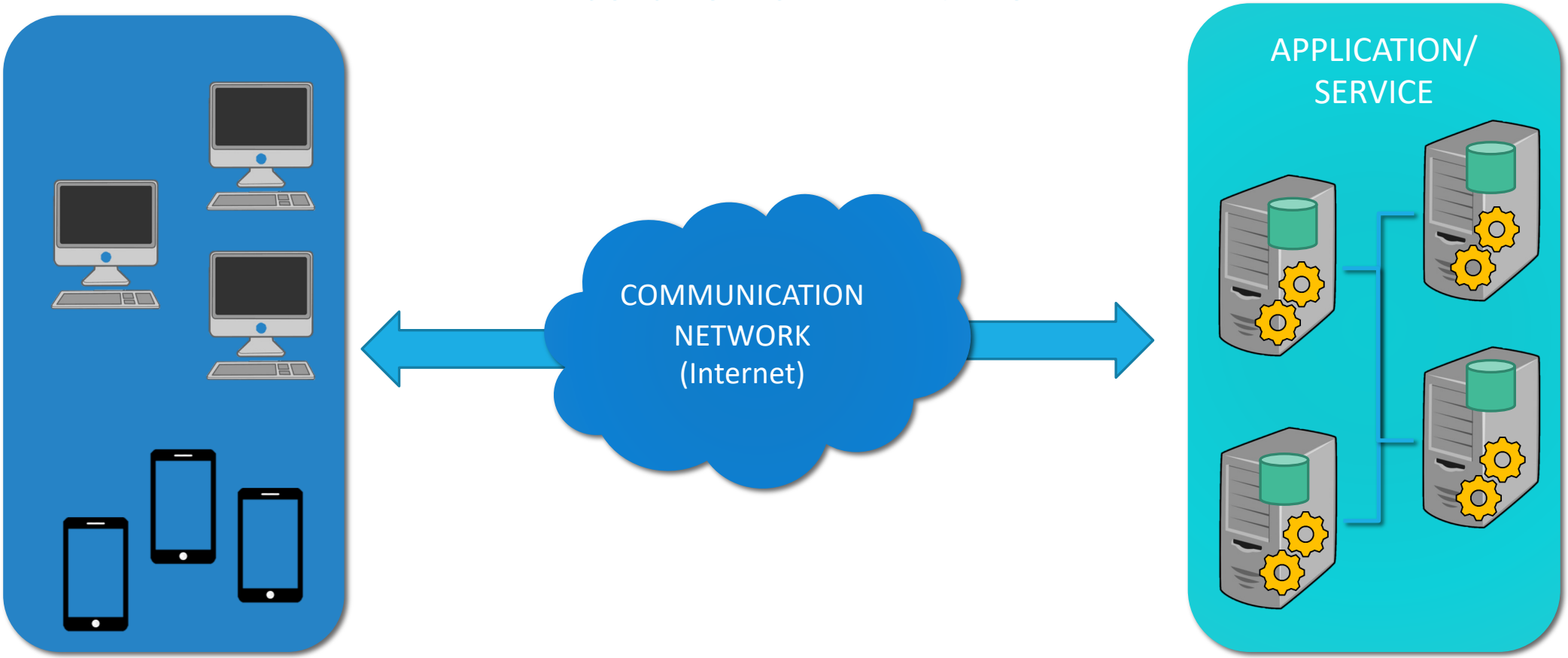
A system is scalable if it remains running with adequate performance even if the number of resources and/or the number of users grow up of orders of magnitude

Centralization is against scalability



Scalability

THE SOLUTION IS IN REPLICATION

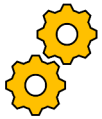


Scalability

REPLICATION ISSUES

- Service
  - Coordination Problems

- Data
  - Consistency Problems



- Computation
  - No node has the current state of the whole system
  - Nodes base their decisions on data they own
  - A failure of a node should not compromise the goal of the algorithm

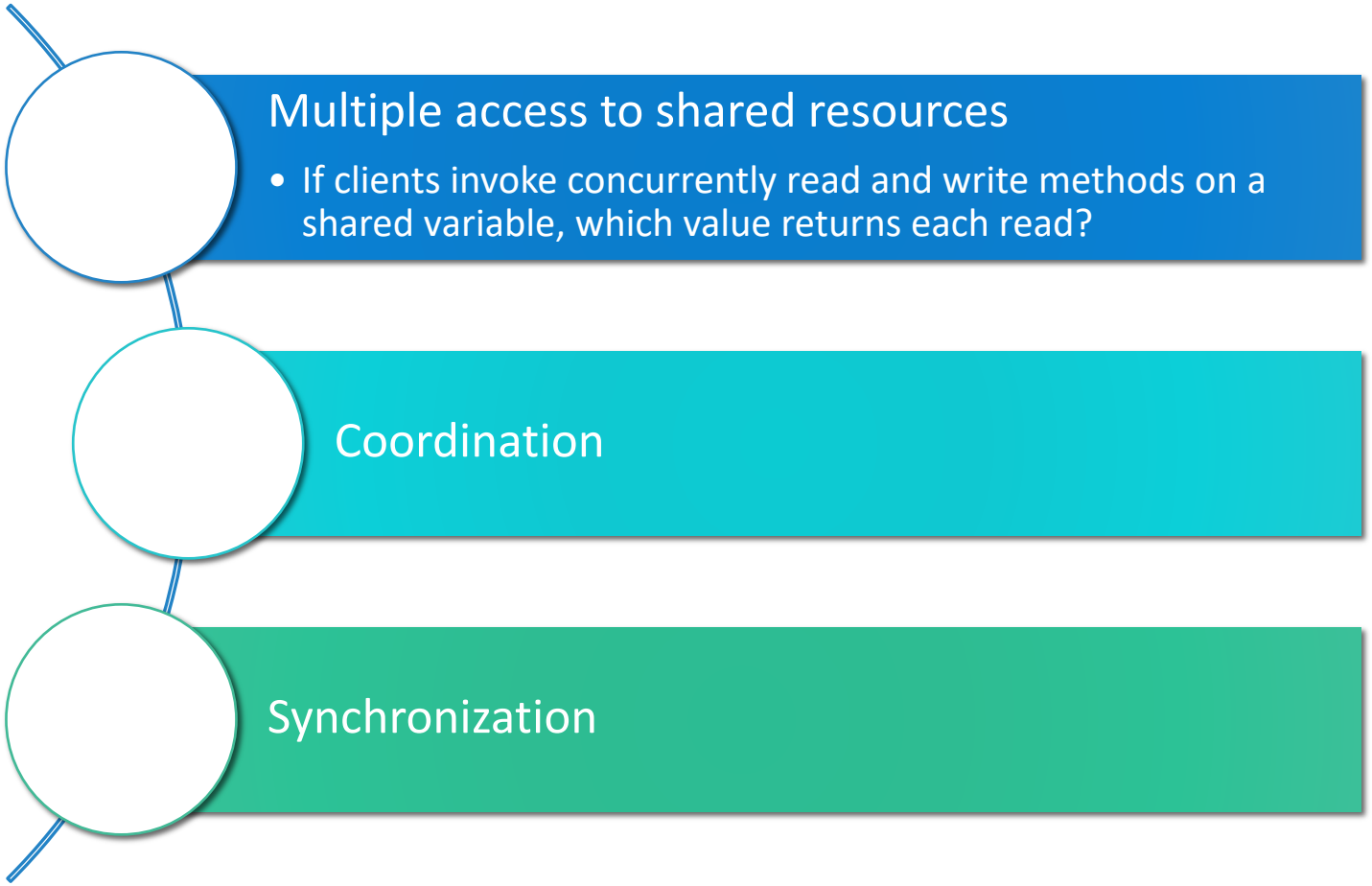
Scalability

The project of a scalable system shows four main problems

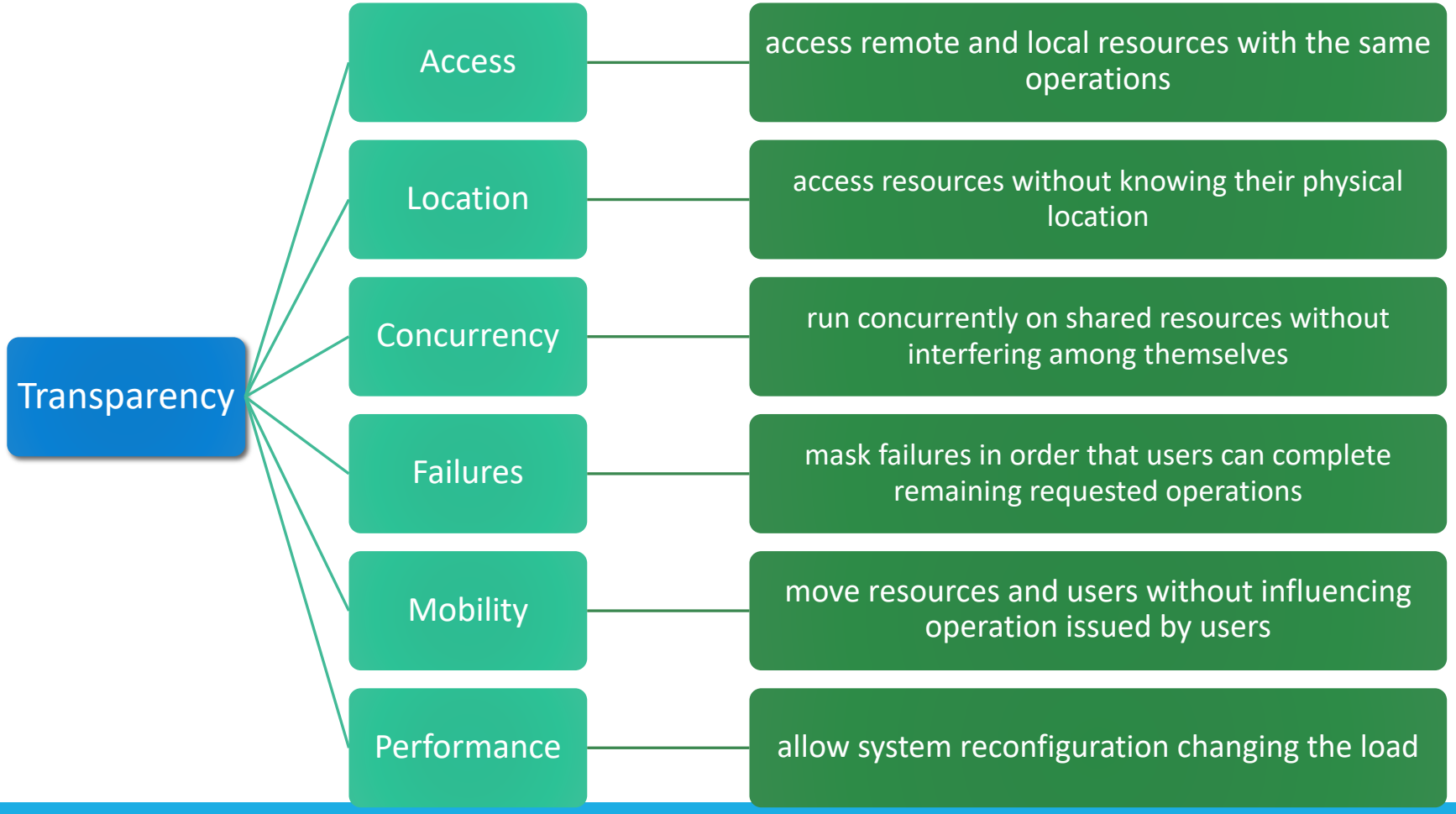
System Dynamicity	• Adding/removing servers/processes on-the-fly
Check performance metrics	• Servers/processes have not to interact with all application' users • Employ algorithms that do not require to use the entire set of data
Using carefully scarce resource	• e.g., battery drain in embedded systems
Avoiding bottlenecks	• e.g., Centralized *vs* distributed DNS

Note that deployments can impose (for security or business requirements) centralized solutions under several conditions

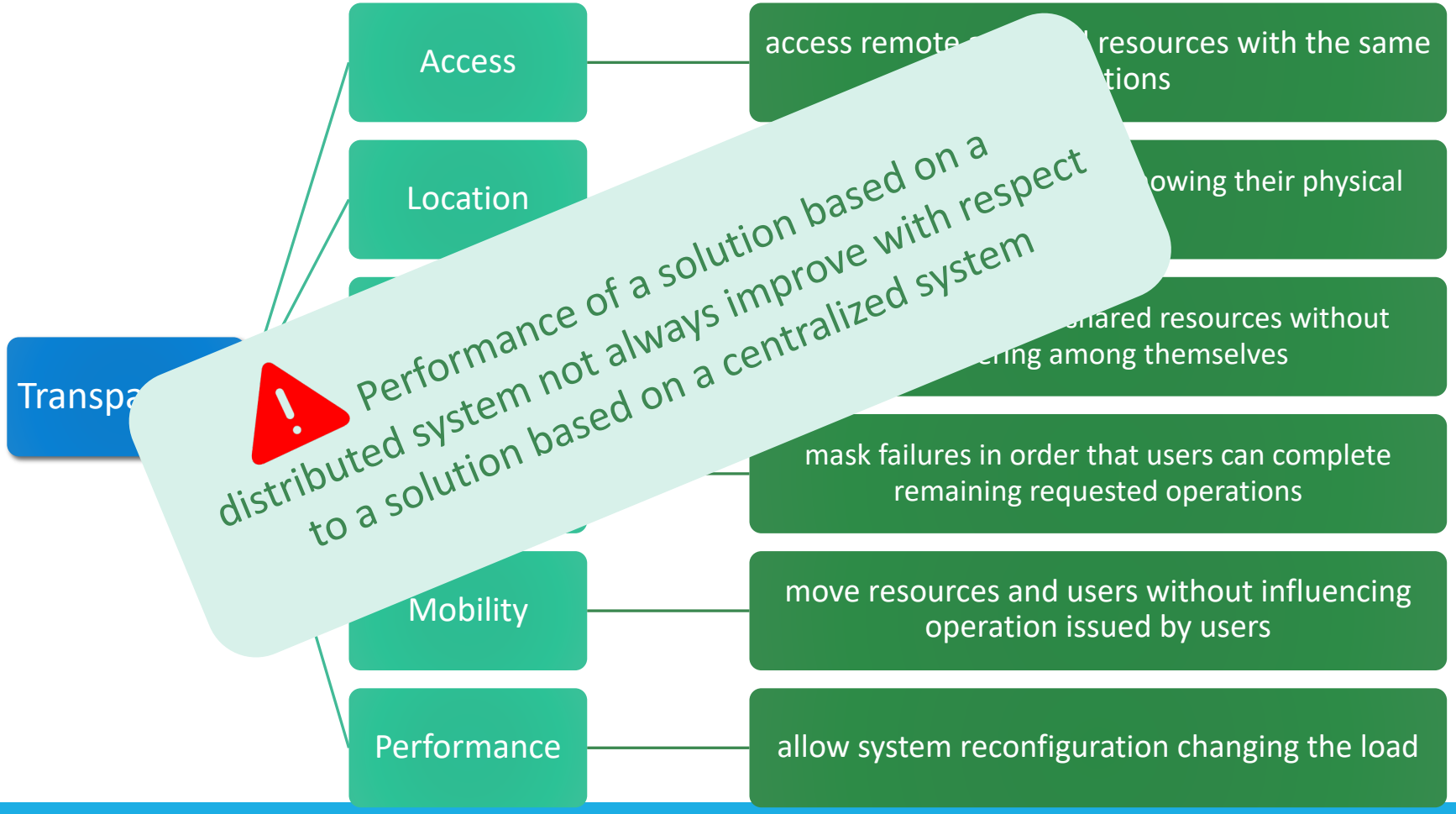
Concurrency



Transparency



Transparency



What will you learn?

References

- Algirdas Avizienis, Jean-Claude Laprie, Brian Randell, Carl E. Landwehr: Basic Concepts and Taxonomy of Dependable and Secure Computing. IEEE Transactions on Dependable and Secure Computing 1(1): 11-33 (2004) https://ieeexplore.ieee.org/document/1335465/