Distributed Systems
Master of Science in Engineering in Computer Science

AA 2020/2021

INTRODUCTION

General Information

- 6 CFU all in the first semester
  - October 6th– December 18st

- Instructor: Silvia Bonomi

General Information

- Due to the current COVID-19 situation, lectures will be given in *blended mode*
  - You can find always updated information here https://www.uniroma1.it/en/notizia/covid-19-phase-3-person-and-online-classes-exams-and-graduation-sessions

- Schedule
  - Tuesday 13:00 – 15:00
  - Thursday 10:00 – 13:00

- In presence
  - Tuesday Room T01, Marco Polo Building
  - Thursday room 205 Marco Polo Building

- Remotely
  - On zoom
    - https://uniroma1.zoom.us/j/86875216769?pwd=d2tzRkUrSE1ob1hlUjMwa0NsT1BlQT09
    - Meeting ID: 868 7521 6769
    - Passcode: 728538

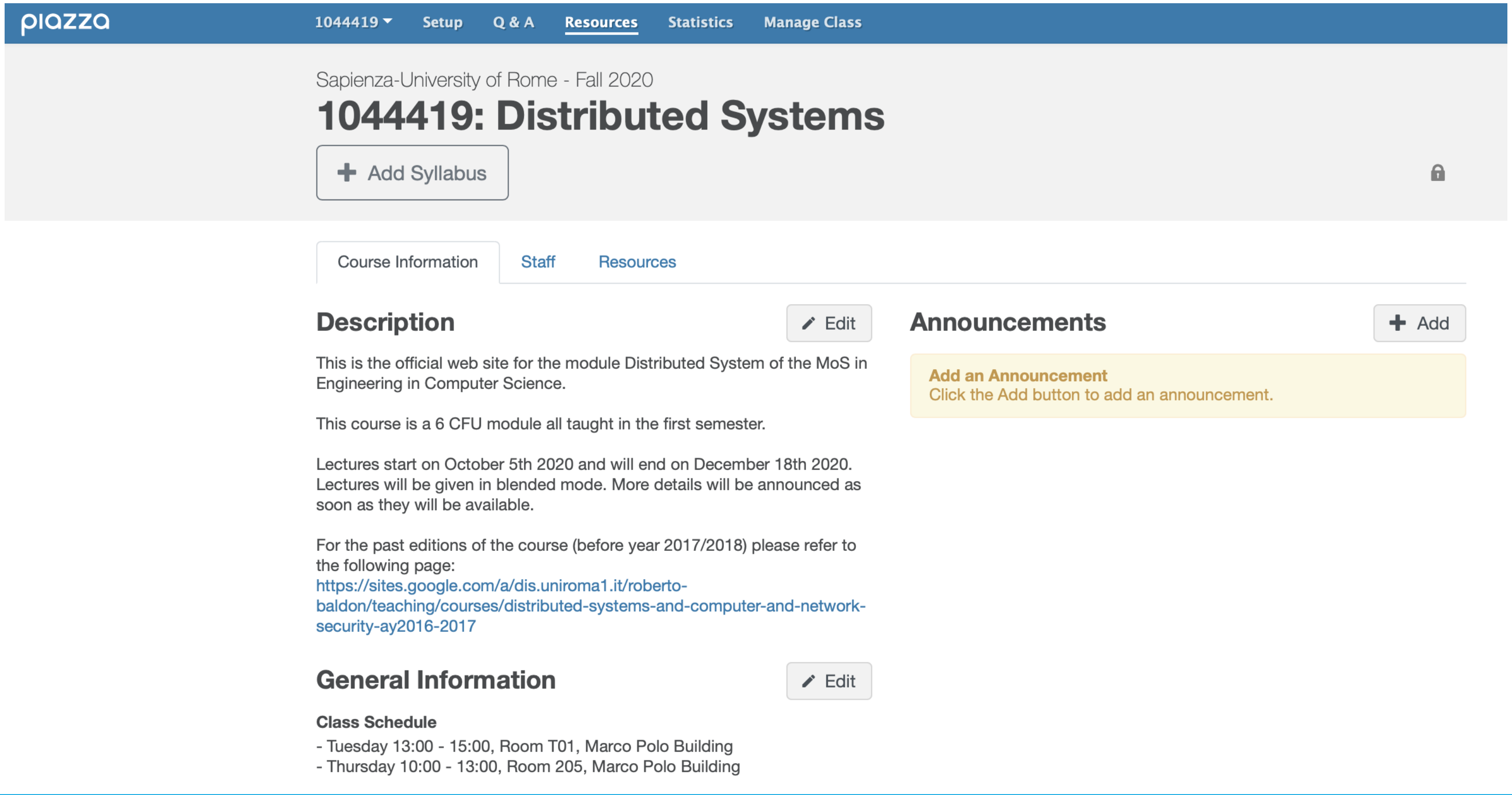
General Information

➢ For everybody’s safety take particular care to the following 4 rules:

1. **Wash your hands,** and do it often.
   - Use water and soap or hydroalcoholic solutions.
2. **With fever or other respiratory symptoms, stay at home**
   - e.g., cough, cold, sore throat, gastrointestinal symptoms, alterations in smell or taste.
3. **Physical distancing** is very important.
   - Two meters (around six feet) or more but never less than 1 meter.
4. **Wear your mask** at all times in public spaces and when you cannot keep the six-foot rule.

General Information

Web site: https://piazza.com/uniroma1.it/fall2020/1044419/info



Sapienza-University of Rome - Fall 2020

1044419: Distributed Systems

Description

This is the official web site for the module Distributed System of the MoS in Engineering in Computer Science.

This course is a 6 CFU module all taught in the first semester.

Lectures start on October 5th 2020 and will end on December 18th 2020. Lectures will be given in blended mode. More details will be announced as soon as they will be available.

For the past editions of the course (before year 2017/2018) please refer to the following page: https://sites.google.com/a/dis.uniroma1.it/roberto-baldon/teaching/courses/distributed-systems-and-computer-and-network-security-ay2016-2017

General Information

Class Schedule
- Tuesday 13:00 - 15:00, Room T01, Marco Polo Building
- Thursday 10:00 - 13:00, Room 205, Marco Polo Building

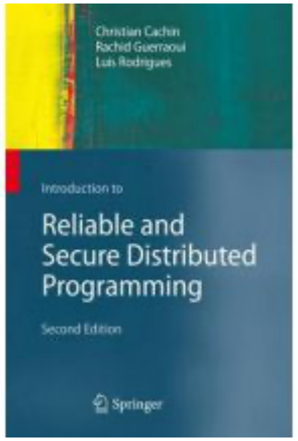
Announcements

Add an Announcement
Click the Add button to add an announcement.

General Information

Material

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

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

Where I am

- My office is at DIAG in Via Ariosto 25
  Room B116, 1st floor, B wing

General Information

- Exam (12 CFU)
  - **IMPORTANT** for students enrolled in Engineering in Computer Science:
    - The overall exam is made of 12 CFU split in two modules:
      - Distributed Systems (DS): 6 CFU
      - Computer Network Security (CNS): 6 CFU
    - To get your exam registered on Infostud, you need to pass both modules and it will not be possible to register them separately.
    - Each module is organised independently and has its own exam dates and exam rules
    - you can take DS and CNS in different sessions but please do it before the end of the academic year to avoid to loose the mark for an already passed module.
    - The final mark for the 12 credits will be the average of the two marks obtained in DS and CNS.

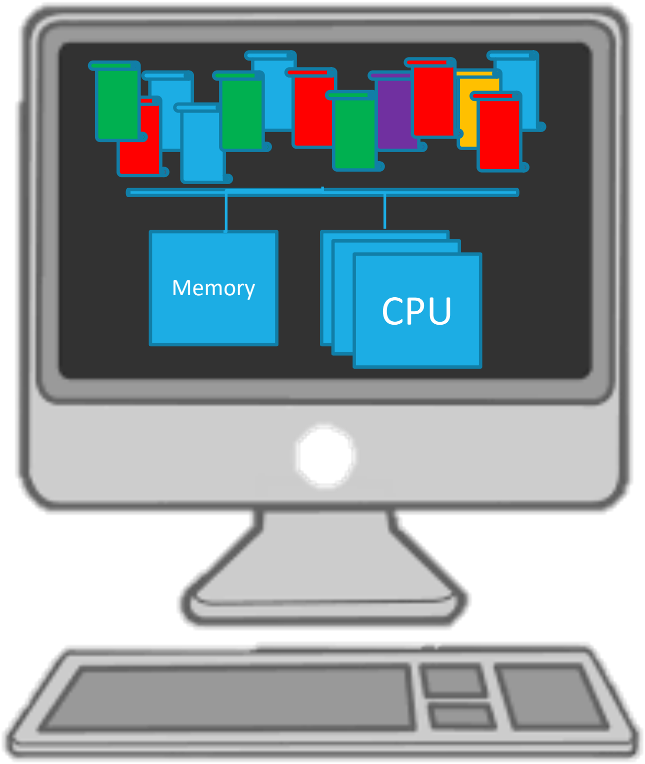
General Information

- DS Exam
  - The exam is made of a written test
  - Questions may cover any topic contained in the final syllabus
  - When you pass the written test, if you are not completely satisfied with your mark, you may ask for an oral exam
    - The oral exam is not mandatory
    - Be aware that if you ask for the oral exam, your final DS mark will results form the average of the written + the oral exam

Introduction to Distributed Systems

From concurrent to distributed systems

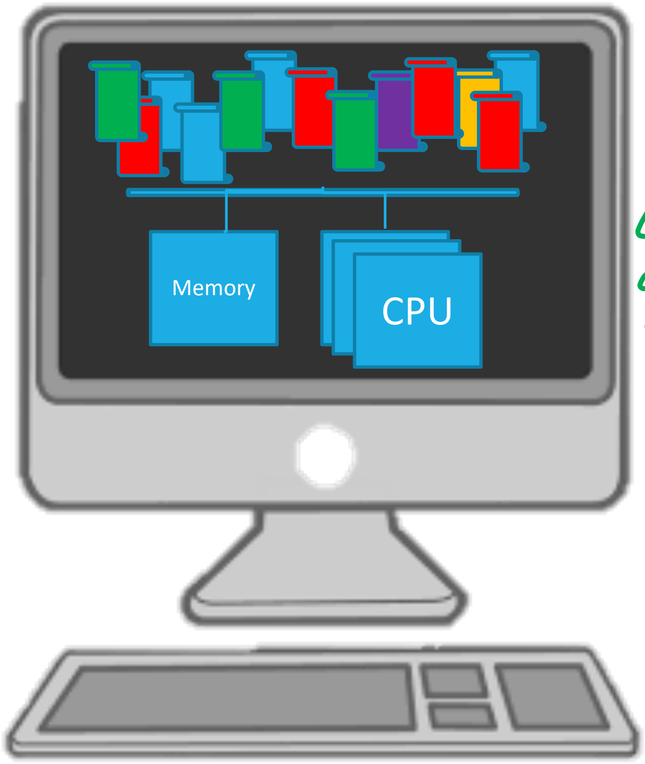
CONCURRENT SYSTEMS



DISTRIBUTED SYSTEMS

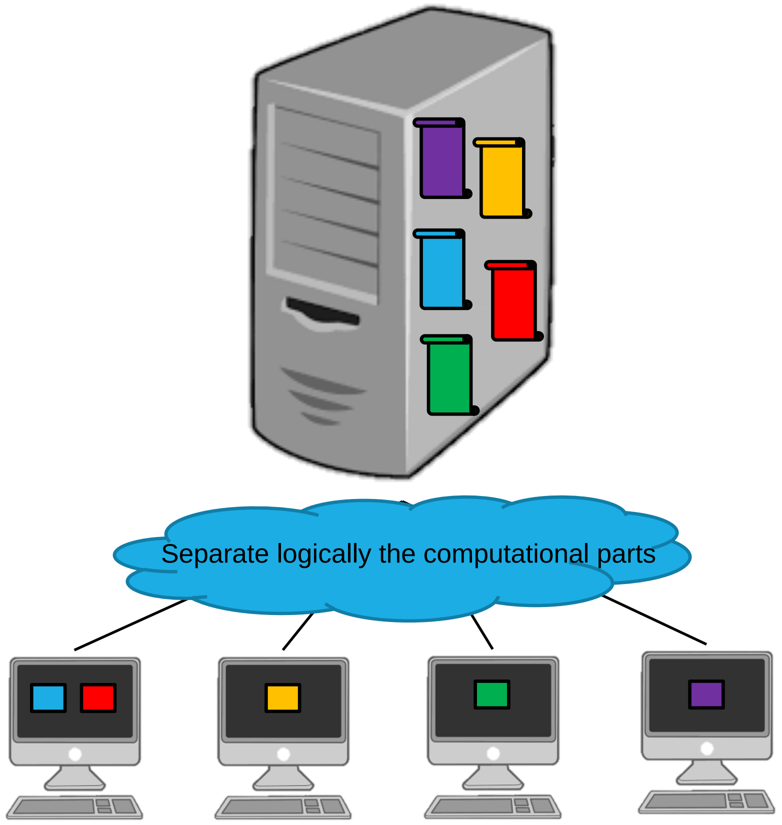
From concurrent to distributed systems

CONCURRENT SYSTEMS



LIMITATION OF POWER SO →

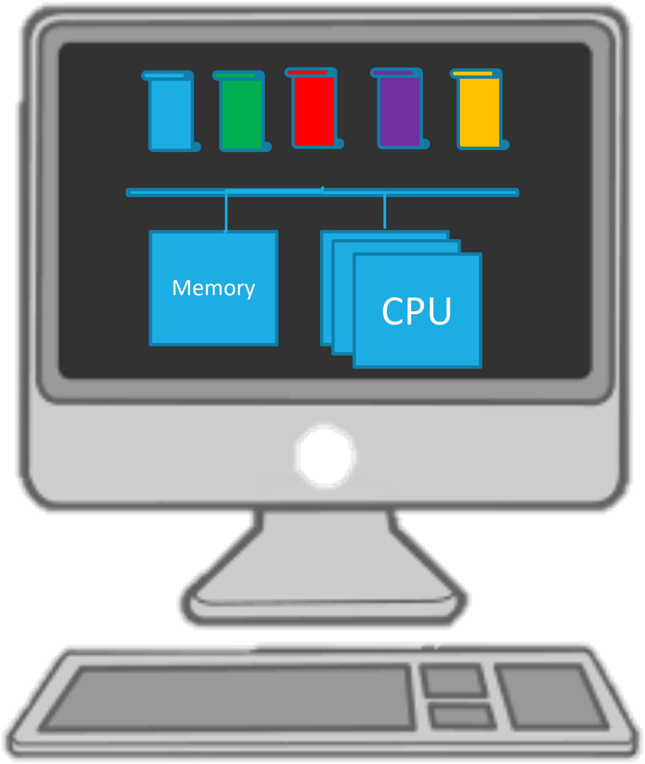
DISTRIBUTED SYSTEMS



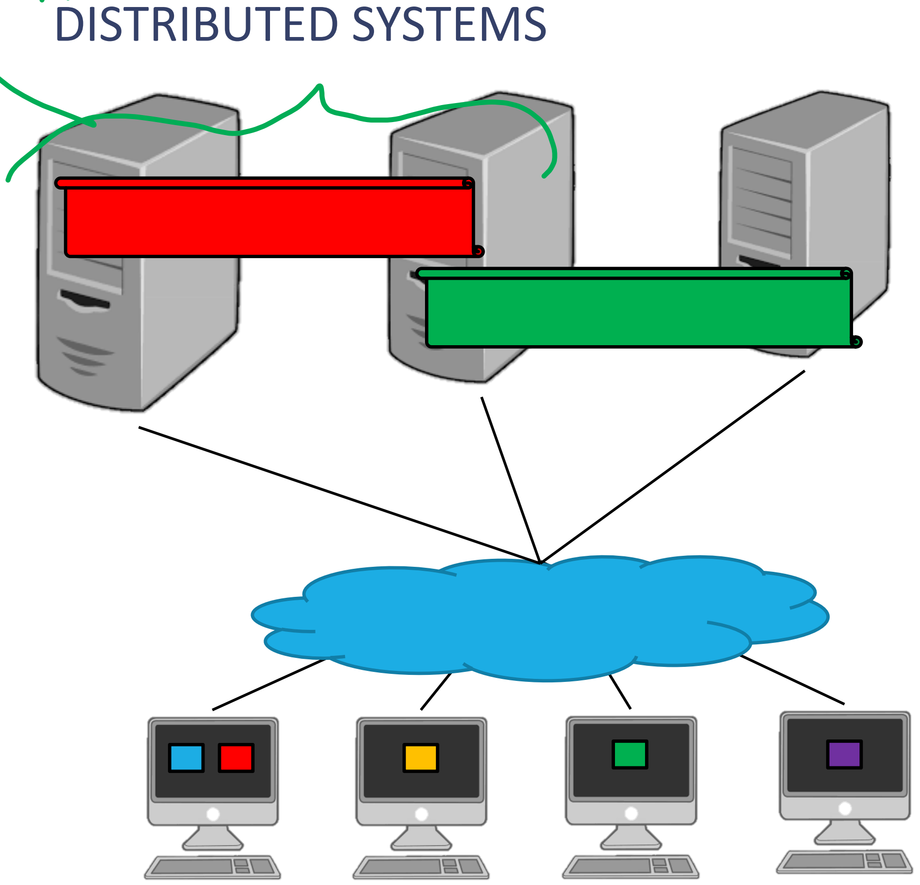
From concurrent to distributed systems

cooperation to run
TAKNOO app.

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 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)*

TRANSPARENCY

Definitions

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 one in which the failure of a computer you didn't even know existed can render your own computer unusable (Leslie Lamport)

DRAWBACK.

Common points across definitions

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

Facebook is a distributed system, you don’t see the systems in the back, a fail can cause the crash of the system; looks like a simple application

Why Distributed Systems?

1. To Increase Performance
   - To cope with the extremely higher 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)

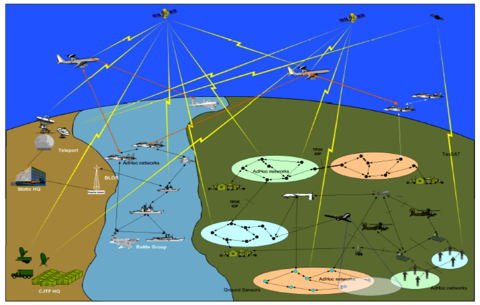
Use of distributed servers

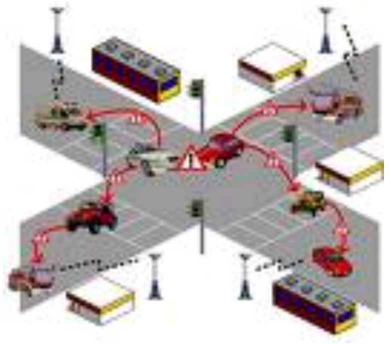
2. To Build Dependable Services
   - To cope with failures

She is fault tolerance specialized

Distributed Systems: examples

Internet

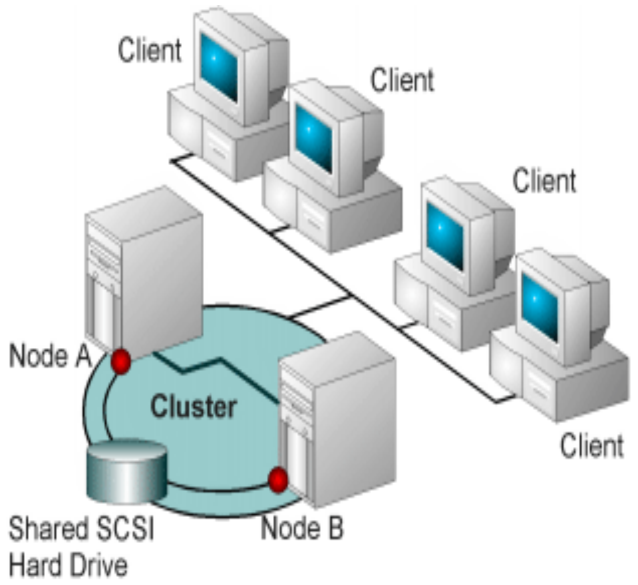




Primary Goal: Overcome the limitation of centralized environment

Problems

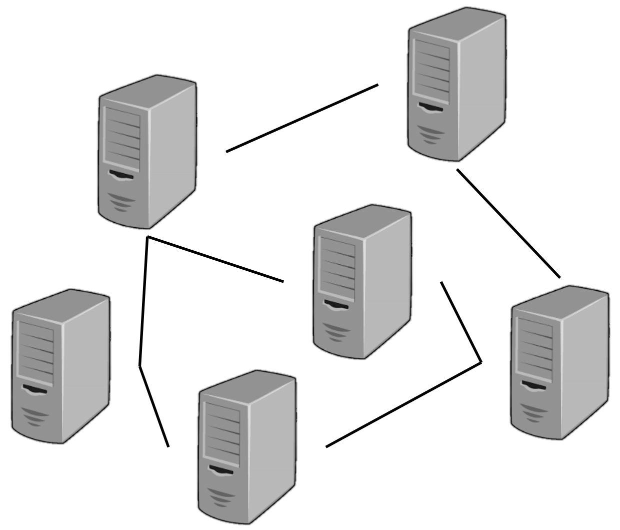
- Connectivity and Communication
- Synchronization
- Coordination



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

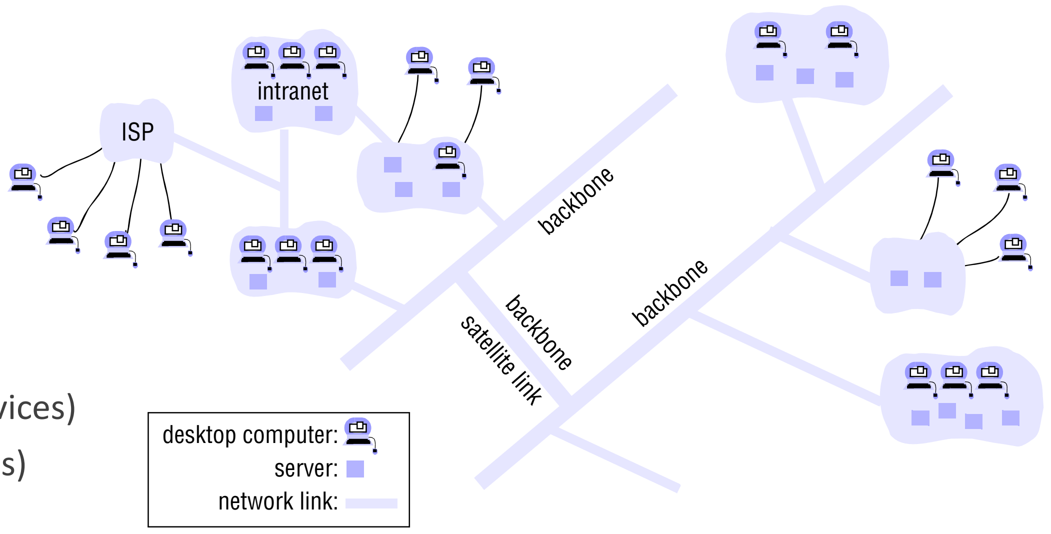
Distributed systems are undergoing a period of significant change and this can be traced back to a number of influential trends:

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

Problem: heterogeneous products based on brands and protocols

Pervasive networking and the modern Internet

Figure 1.3 A typical portion of the 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

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. 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

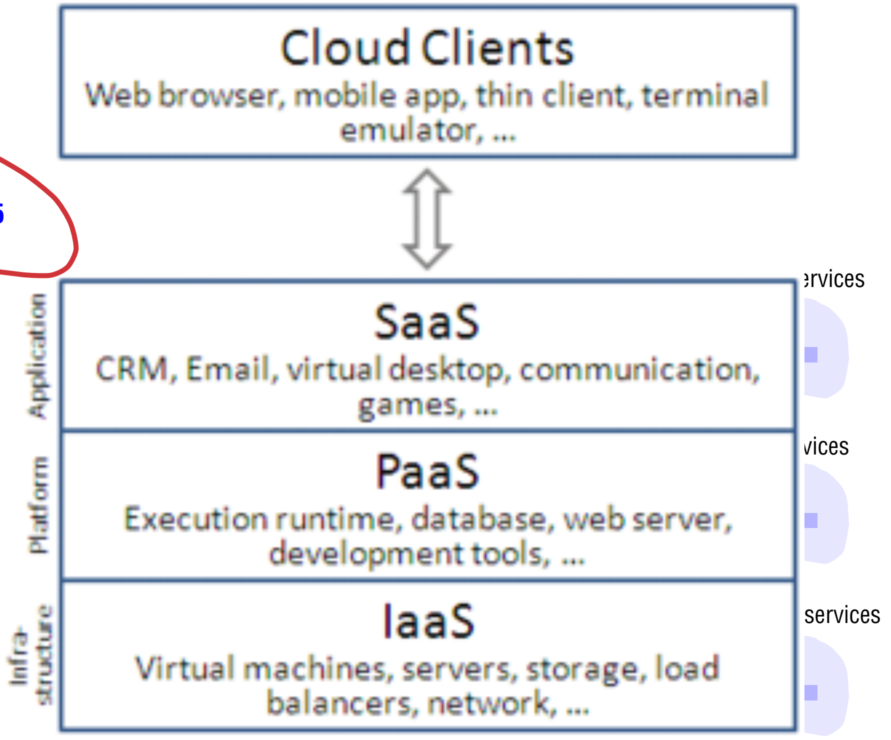
Distributed computing as a utility

CLOUD COMPUTING

- IaaS
- PaaS
- SaaS

Figure 1.5

find it



Cloud Clients
Web browser, mobile app, thin client, terminal emulator, …
SaaS
CRM, Email, virtual desktop, communication, games, …
PaaS
Execution runtime, database, web server, development tools, …
IaaS
Virtual machines, servers, storage, load balancers, network, …
Application
Platform
Infra-structure
ervices
vices
services

Characteristics and Challenges

- Heterogeneity
- Openness
- Security
- Scalability
- Fault Tolerance
- Concurrency
- Transparency

Learn them

Heterogeneity

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

Learn them

Solutions

Middleware (from RPC to Service oriented Architectures)

Mobile code and Virtual Machine

Openess

Capability of a system to be extended and re-implemented

Necessary condition, set of documents with software interfaces

Interface Definition Language (it describes the syntax and the semantic of a service/component, available functions/services, input parameters, exceptions, etc)

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

- Complete. A specification is complete if every thing related to the implementation has been specified. If something has not been specified, the designer needs to add implementation dependent details.
- Neutral. A specification is neutral if it does not offer any detail on a possible implementation

Openess (ii)

Most important

Interoperability. The capability of two systems to cooperate by using services/components specified by a common standard

Portability. The capability of a service/component implemented on a distributed system A to work on a system B without doing any modification

Flexibility. The capacity of a system to configure/ orchestrate components developed by various programmers

Add-on Features. The capacity of a distributed system of adding components/services and be integrating in a running system

Openess (iii)

Other recent capabilities:

- Evolvability. The capacity of a system to evolve in time for example leaving active two different version of the same service.
- Self-* (self organization, self management, self healing etc.). The capacity of a system to reconfigure, to manage itself without human intervention

The number of independent software developers make very complex the development of a distributed platform

Security

Avoid data breaches

Confidenziality (protection against the interception of data from unauthorized users)

Integrity (protection against data alteration)

Availability (protection against the interference in the access to a resource)

Scalability

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

Centralization is against scalability :

- Service (single service for all users)
- Data (a single table for all users)
- Algorithms (routing using complete information)

Scalability (ii)

It becomes necessary using:

- Service Replication
  - Coordination Problems
- Data Replication
  - Consistency Problems
- Distributed Algorithms
  - 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 compromize the goal of the algorithm

Geographic Scalability

Scalability (iii)

The project of a scalable system shows four main problems:

System Dynamicity

- Adding/removing servers/processes on-the-fly

Check performance metrics

- E.g. 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

- Centralized vs distributed DNS

Note that deployments can impose for security or enterprise requirements centralized solutions under several conditions.

Failure Management

Failure detection
- Example: Checksum detects a corrupted packet

First: how to detect in distributed
So try to interact to check if app is still working
Sometimes is not possible

Failure masking
- Example: message retransmission

Tolerating Failures
- Example: intrusion tolerant systems

Failure Recovery
- Example: completing long running computation

Redundancy
- Example: DNS

Failure forecasting

Concurrency

Multiple access to shared resources

- If clients invoke concurrently read and write methods on a shared variable, which value returns each read?

Coordination

Synchronization

Transparency

- Access: allow to access remote and local resources with the same operations
- Location: allows to access resources without knowing their physical location
- Concurrency: allows a set of processes to run concurrently on shared resources without interfering among themselves
- Failures: allow to mask failures in order that users can complete remaining requested operations
- Mobility: allows to move resources and users without influencing operation issued by users
- Performance: allow system reconfiguration changing the load

Performance of a solution based on a distributed system not always improve with respect to a solution based on a centralized system.

Quality of Service

The main non-functional properties of systems that affect the quality of the service experienced by clients and users are

- reliability,
- Security,
- Performance,
- Adaptability

What is dependability?

"Dependability of a computing system is the ability to deliver service that can justifiably be trusted"

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

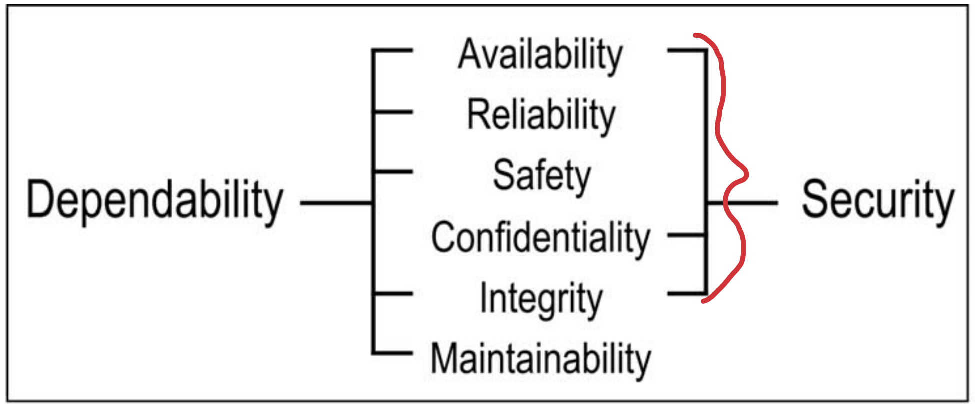
Dependability Attributes

- **availability**: readiness for correct service.
- **reliability**: continuity of correct service.
- **safety**: absence of catastrophic consequences on the user(s) and the environment.
- **integrity**: absence of improper system alterations.
- **maintainability**: ability to undergo modifications and repairs.

Dependability and Security

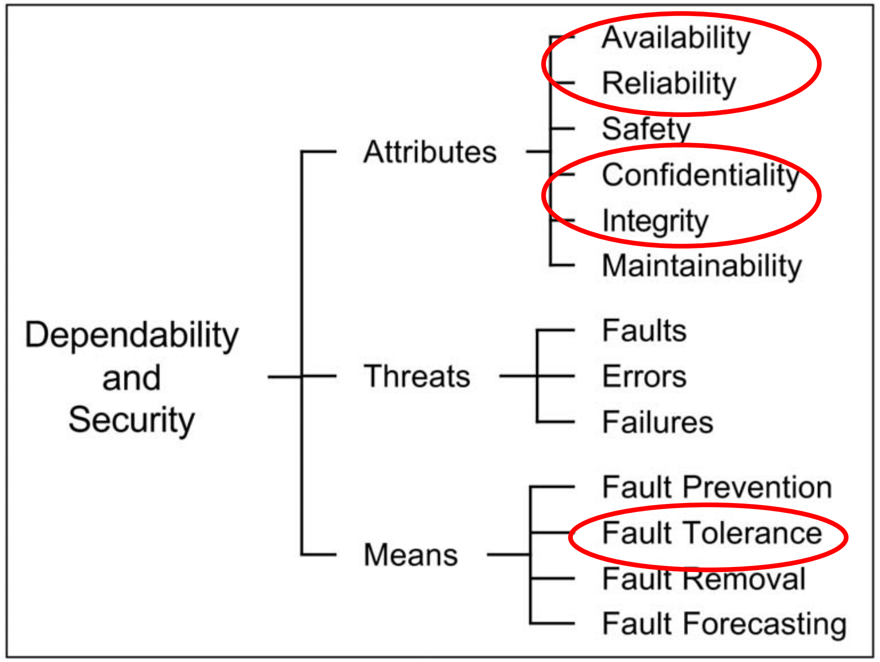
Security is a composite of the attributes of

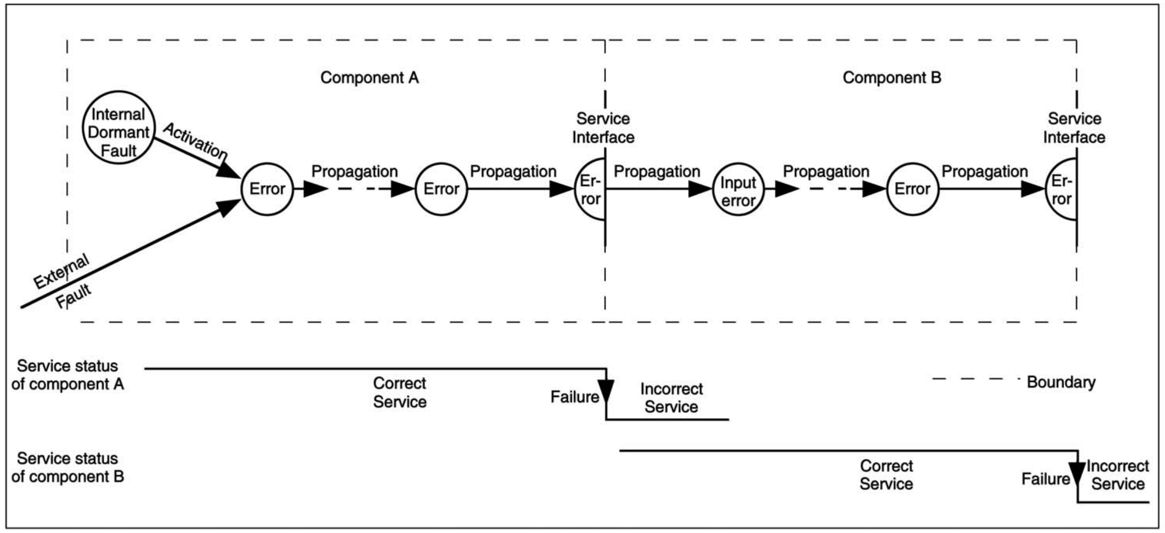
- Confidentiality
- Integrity
- Availability

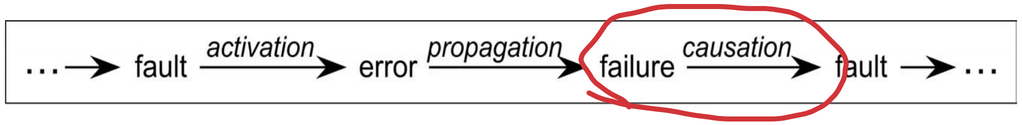


-> IS A SUBSET OF DEPENDABILITY

Dependability and Security





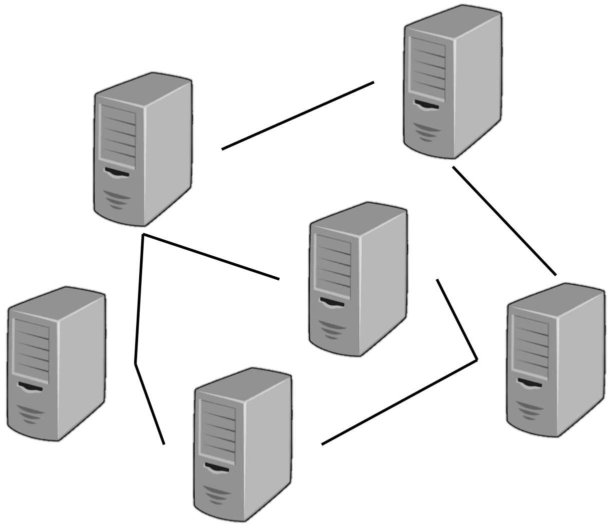


In this course only red things

What will we learn during this course?

Fundamentals of Distributed systems design

- Realizing Reliable communication
- Process synchronization
- Reaching agreement
- Building consistent data store



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/

NOTE: Use the Sapienza proxy to access this paper. Instruction on how to do it can be found here https://web.uniroma1.it/sbs/easybixy/easybixy