Course rules

ARCOS Group

Distributed Systems

Bachelor In Informatics Engineering

Universidad Carlos III de Madrid

Open questions

- ¿Why are you HERE?
- ¿Do you think this course is usefull?
- ¿What do you expect?
- ¿What would you liketo learn?

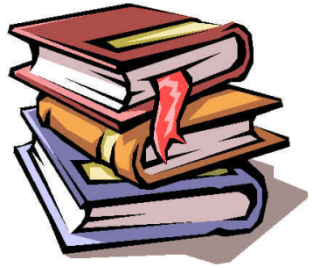
Course profile

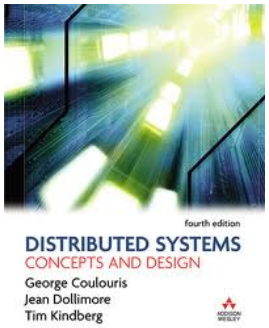
- BACHELOR IN INFORMATICS ENGINEERING
- DISTRIBUTED SYSTEMS
- REQUIRED
- ECTS Credits: 6
- Hours/week: 3
- The purpose is to understand the basic concepts for designing and developing distributed systems and applications

Program

- Unit 1. Basic concepts
- Unit 2. Concurence and communication between processes
- Unit 3. Message passing
- Unit 4. Client-server apps
- Unit 5. Sockets communication
- Unit 6. Remote procedure calls (RPC)
- Unit 7. Web services
- Unit 8. Distributed synchronization
- Unit 9. Distributed file systems
- Unit 10. Distributed and fault tolerance applications
- Unit 11. State-of-the-art of distributed systems

Bibliography





- Distributed Systems, Concepts and Design.

  G. Coulouris, J. Dollimore, T. Kindberg.

  4th edition, 2005.
  Editorial Addison-Wesley

- Computación distribuida. Fundamentos y Aplicaciones

  M. L. Liu.
  2004
  Editorial Pearson.
  Addison-Wesley

Methodology

- Theory classes: present and explain basic concepts. Students must also consult the textbooks (both for theory and problems); it is possible that the professor may not have time to explain all details during class! Ask anything that is unclear, ideally before exam week!!

- Problem solving in class: the professor will solve exercises to illustrate how to apply the concepts learned in the theory class.

- Problem solving in class: the students will solve exercises to make sure they get practical experience and they understand what concepts are still unclear.

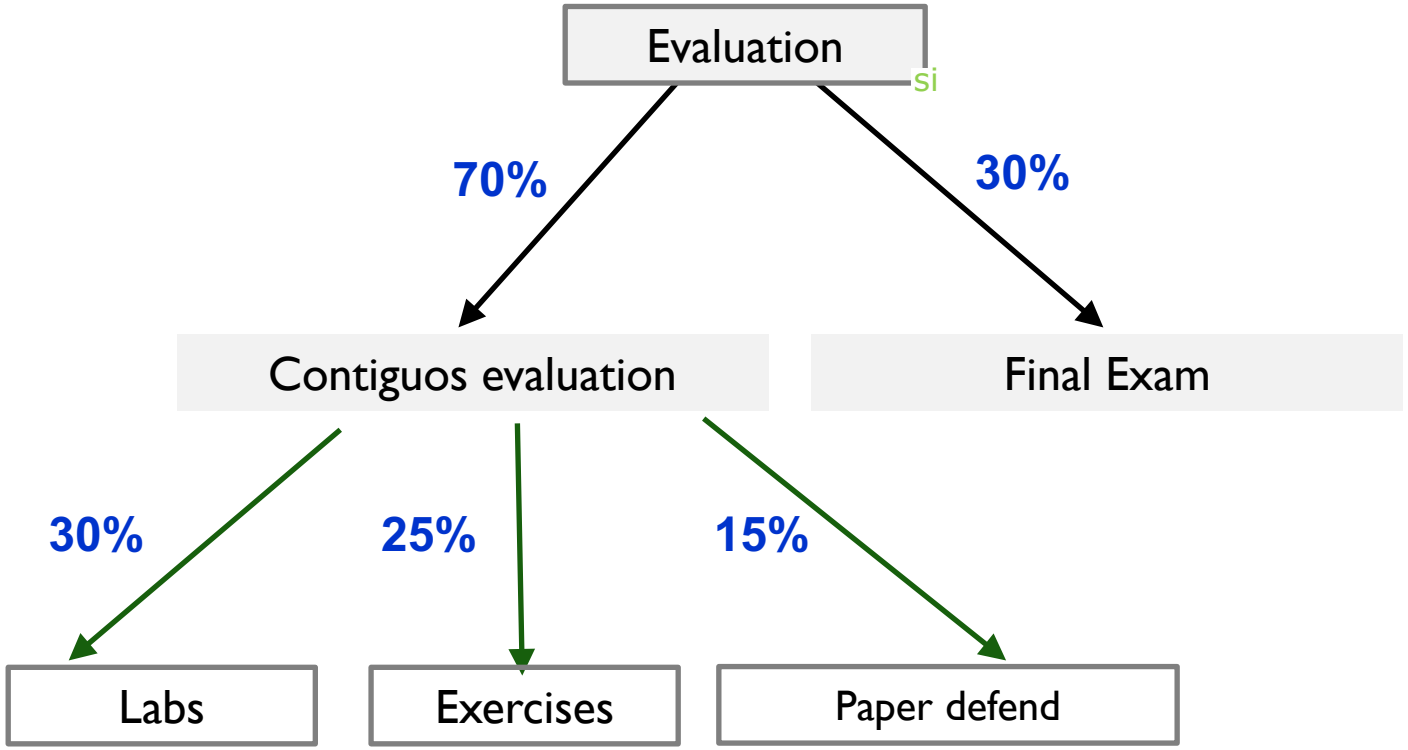
- Lab work: several sets of problems, solved in groups to encourage teamwork.

Schedule

- 14 weeks, 26 classes in total:
- 4 additional lab classes
  - Week 6: Sockets
  - Week 8: RPC
  - Week 10: Cloud computing
  - Week 11: Web services
- Lab room (2.2.C.05)

Evaluation

- Ordinary call (1/2):



Evaluation

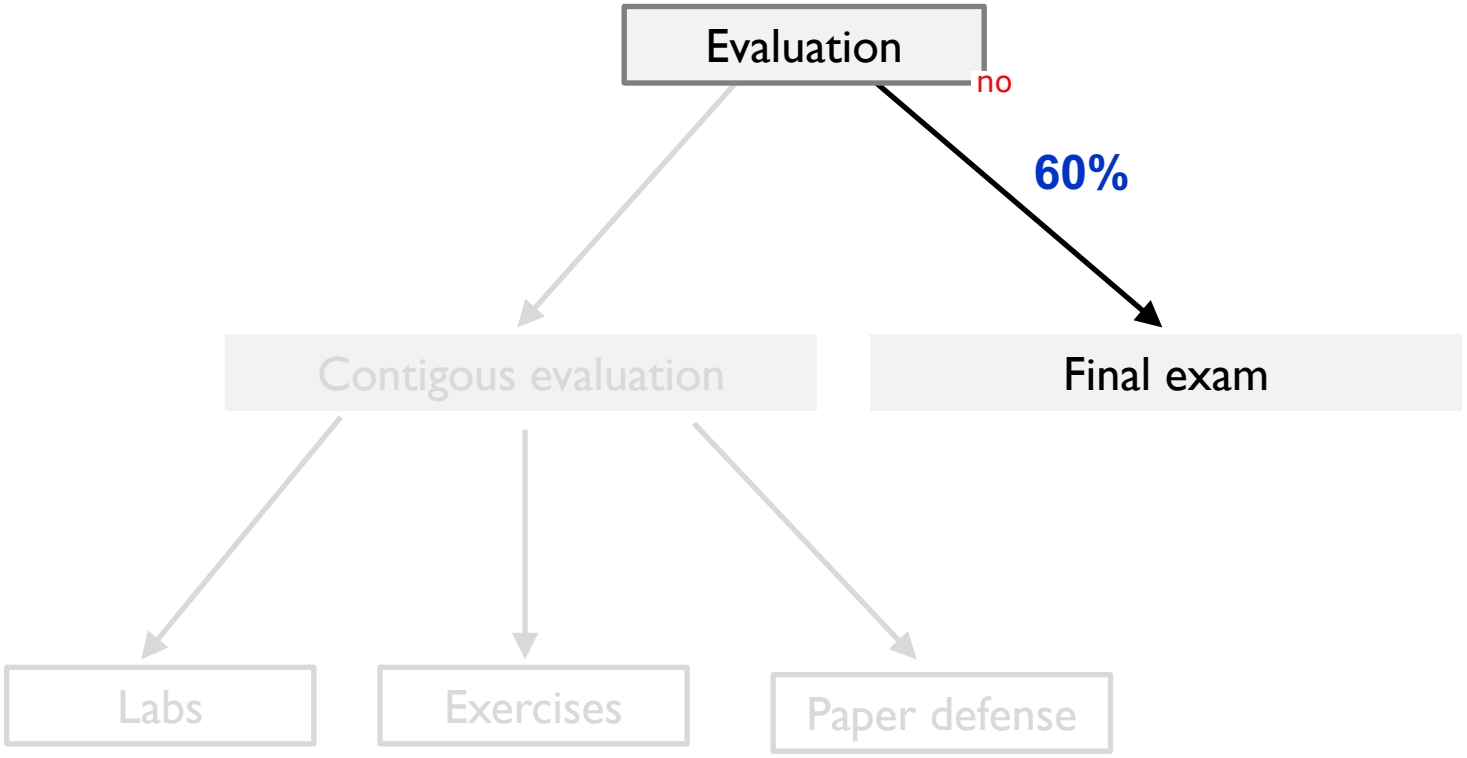
- Ordinary call (1/2):

- Final exam
  - All the unit of the course
  - Minimun mark to pass is **4**

- Paper defense:
  - Presentation and defense of a research paper.
  - Groups of 2 students (max).

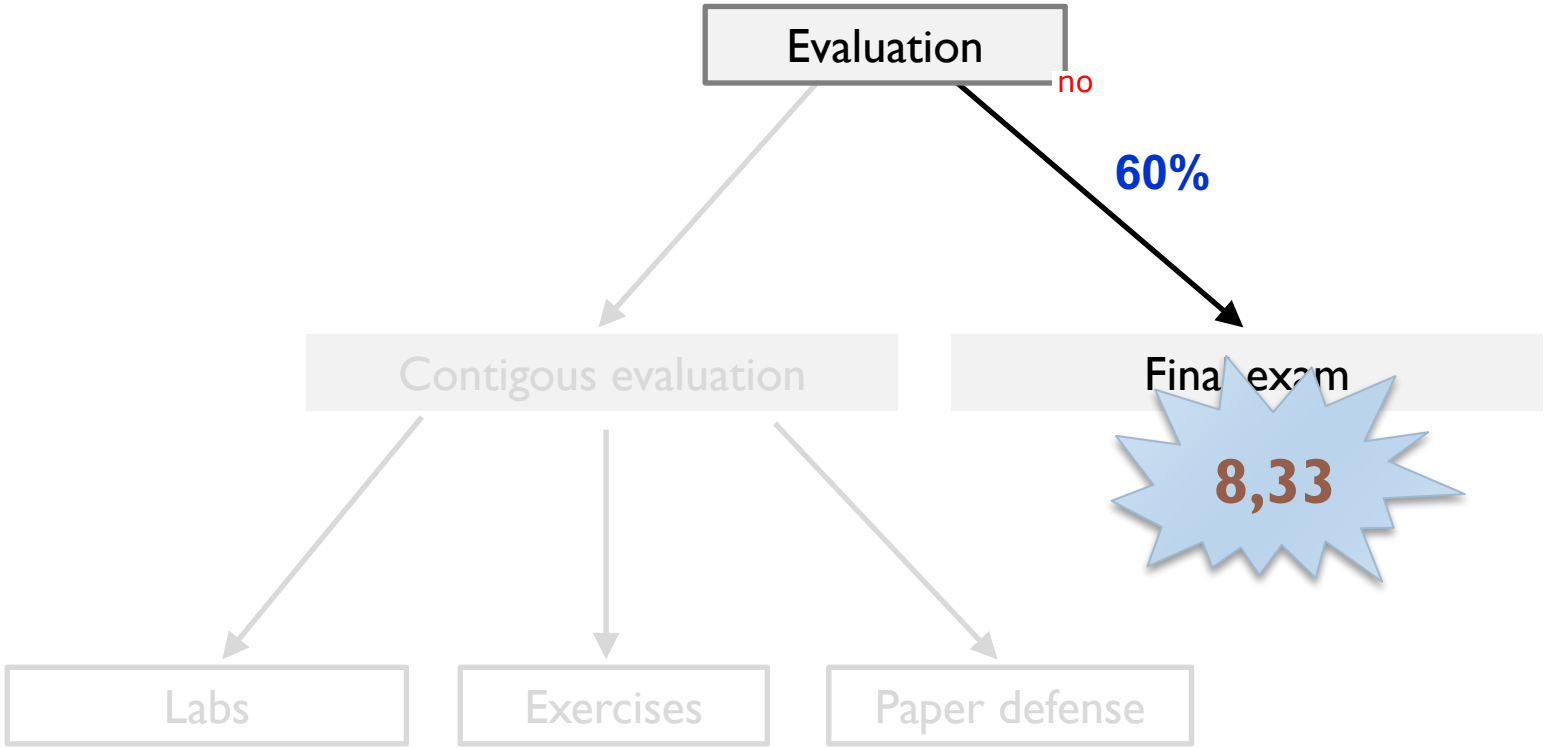
Student evaluation

- Ordinary call (2/2):



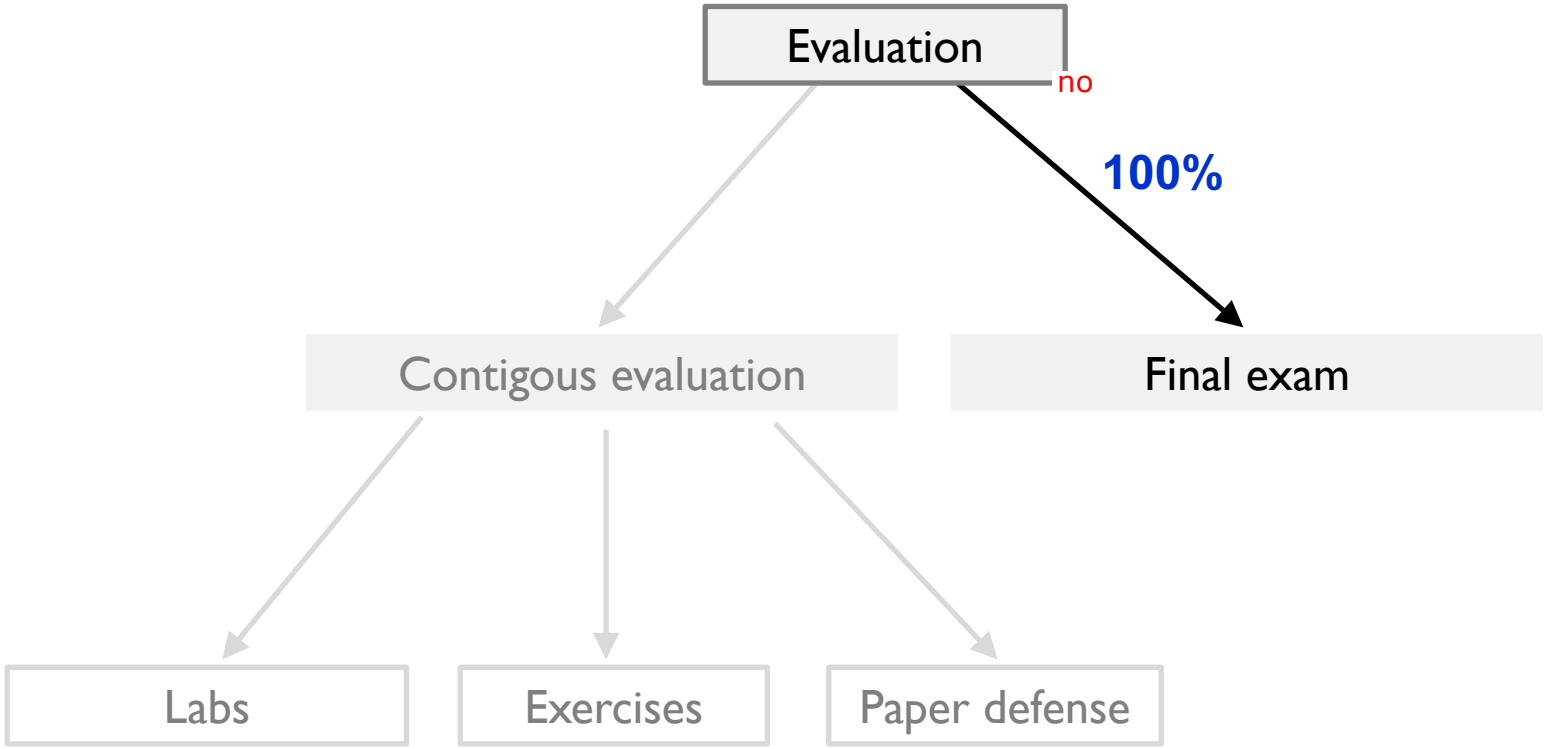
Evaluation

- Ordinary call (2/2):



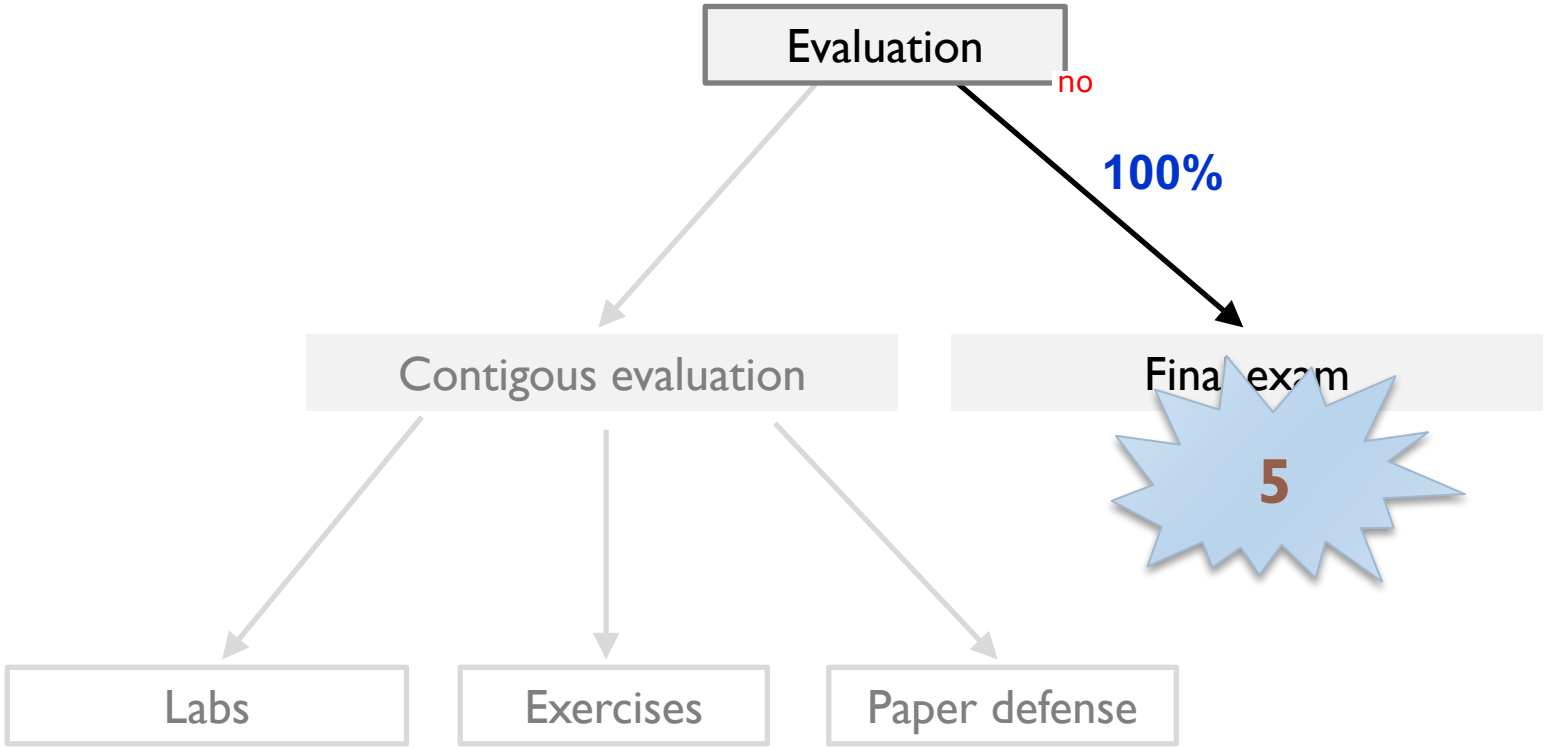
Evaluation

- Extraordinary call (1/2):



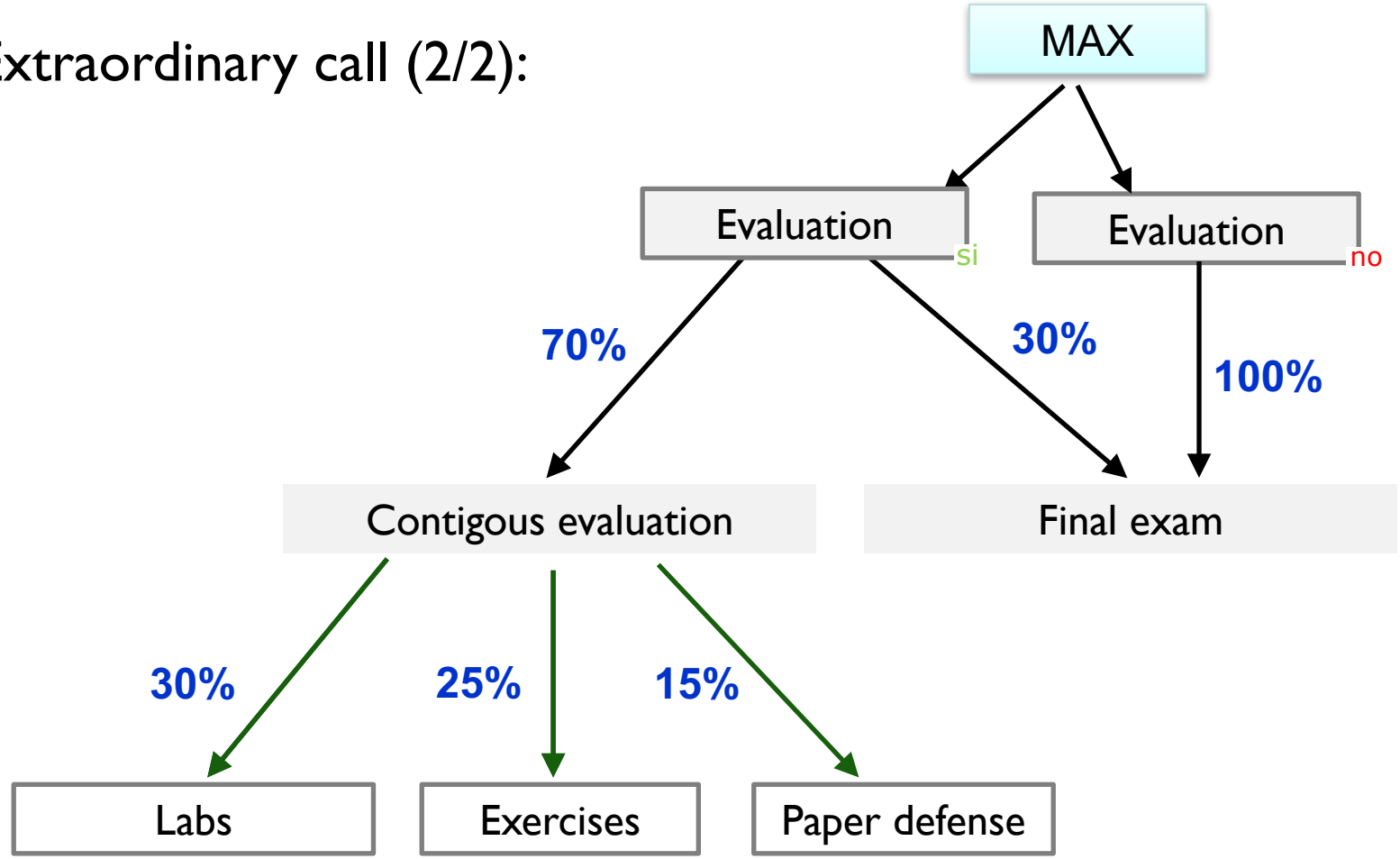
Evaluation

- Extraordinary call (1/2):



Evaluation

- Extraordinary call (2/2):



Evaluation: labs

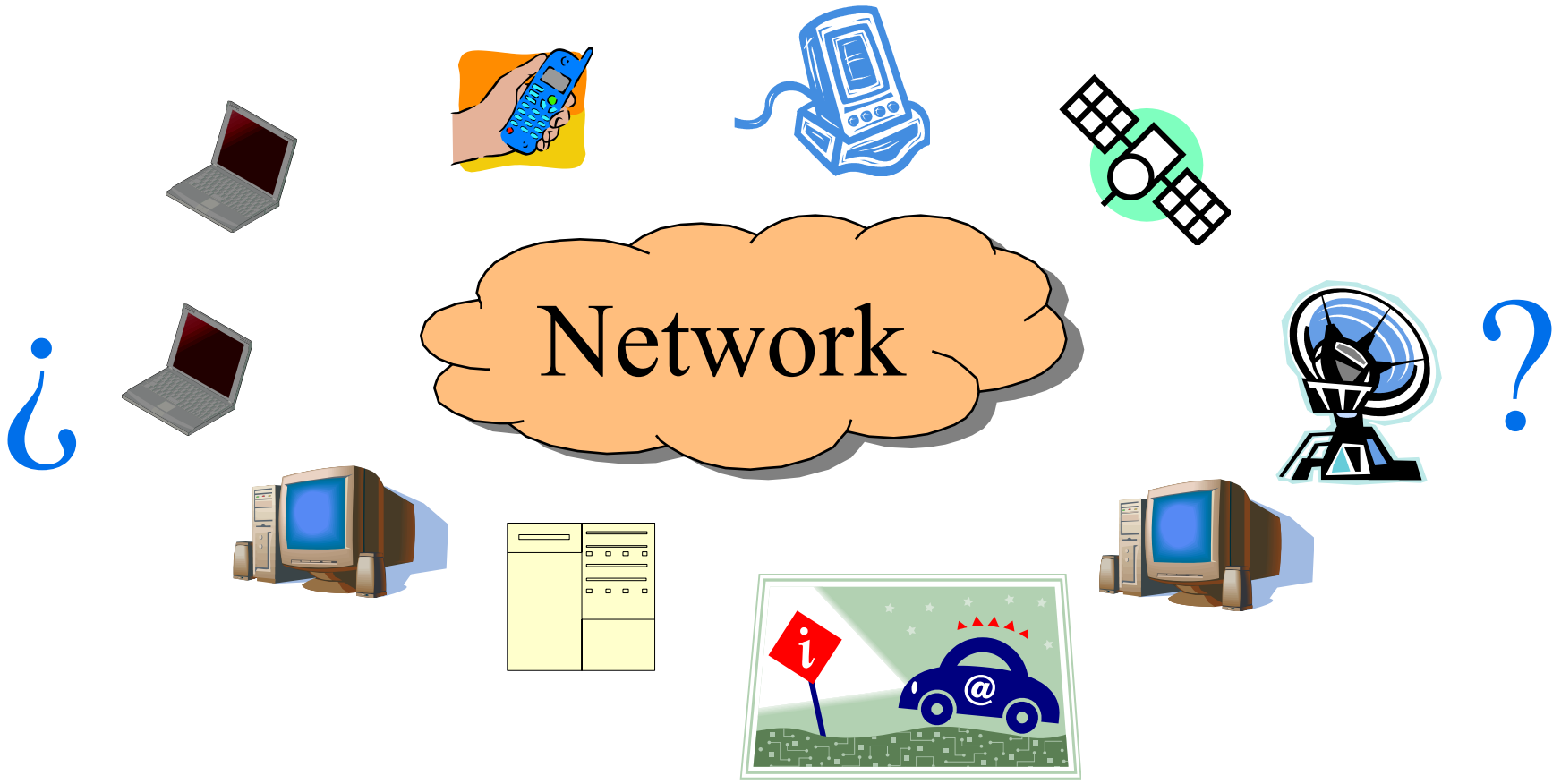
- Two programming assignments (**30%**)
  - Each assigment has a score of **15%**
  - Minimun score is 2 over 10

- Attending is not mandatory (but recommendable)
- Up to 4 students per group (max).

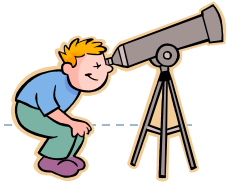
Evaluación: exercises

- Five exercises
  - **25%**
  - **Three exercises (homework)**
  - **Two quizs (paper defense)**
- One per student (**not collaborative**)

Objectives

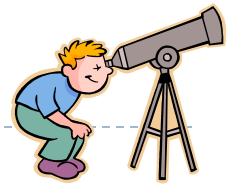


Objectives

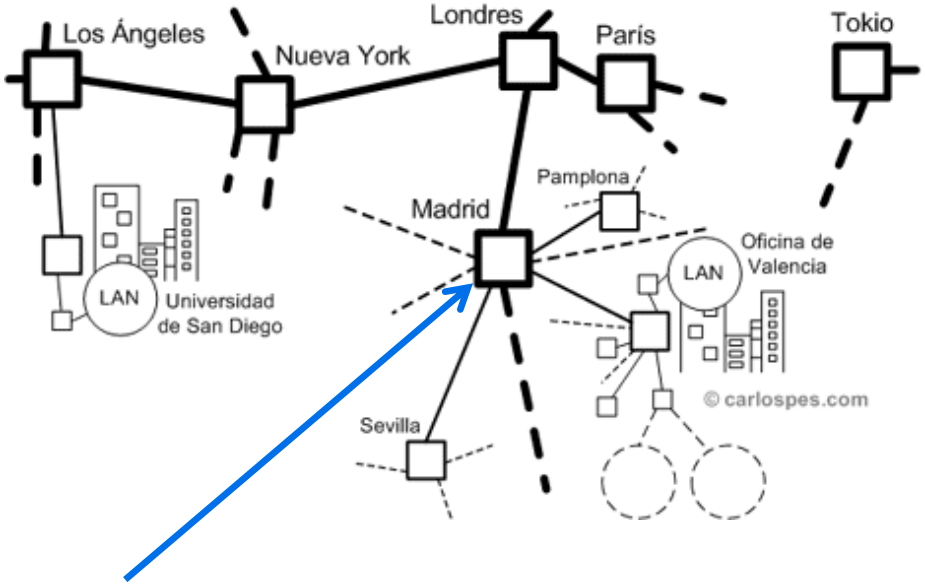


- **Study** the fundaments and basic concepts of distributed systems
- **Design** and **implement** distributed applications using basic concepts such as sockets, RPCs and web services
- **Evaluate** the **performance** of distributed applications
- **Learn how to solve problems** related to distributed systems

Example 1: Recap



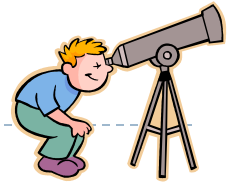
- What concepts do we need to learn or recap?
  - **Computer networks**
  - Basic OS concepts (**processes and concurrent programming**)



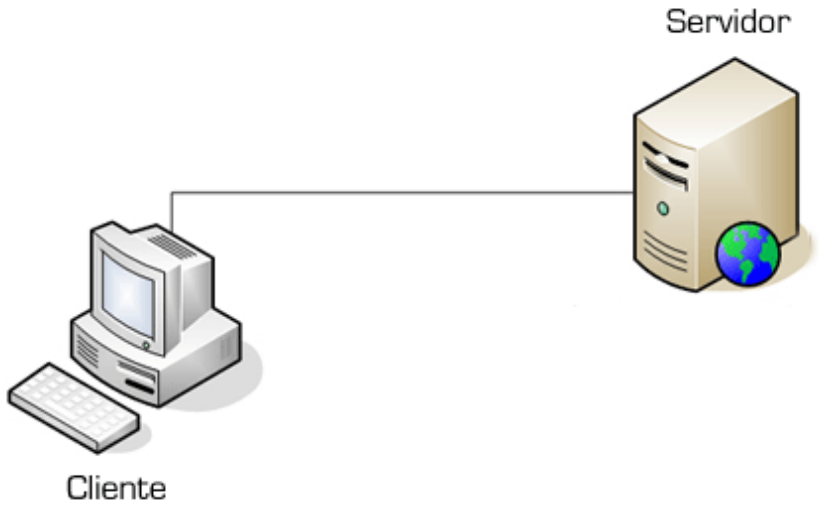
Universidad Carlos III
de Madrid

email
Paso de mensajes
Modelo OSI
Wireless
Modelo TCP/IP
Ethernet
switch
Puerto
HTTP
TCP/UDP
WWW
hub
firewall
router
backbone
ARP
Protocolos
Internet
ICMP
RPC
NFS
DNS
RMI
dirección IP
socket
middleware
socket

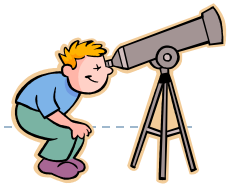
Step 2: Study

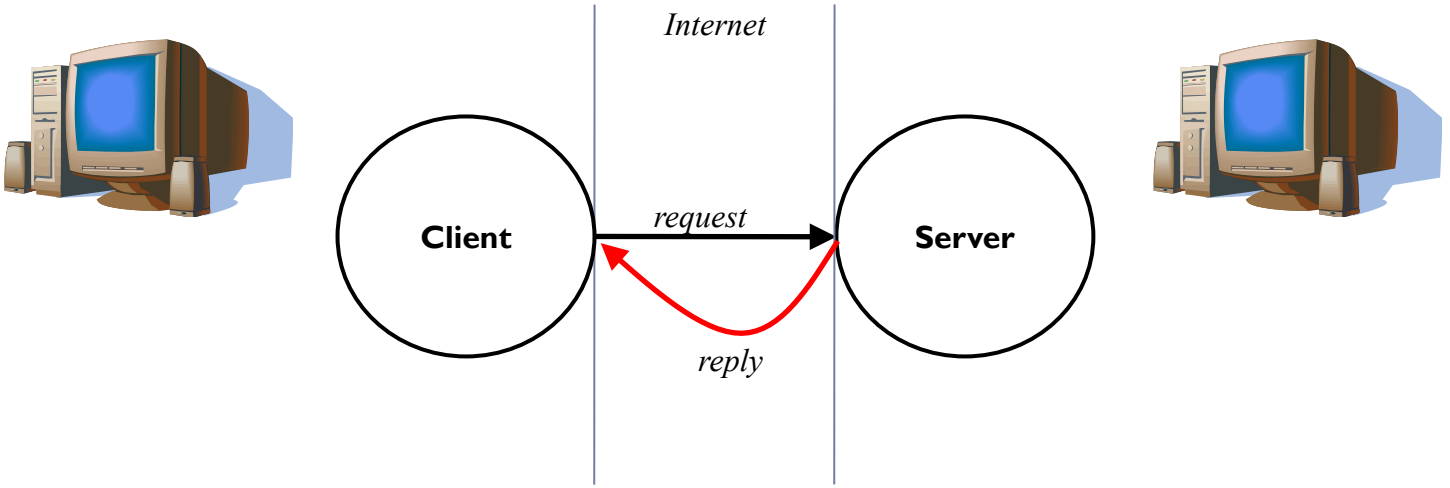


- **Communication paradigms** for distributed systems
- Which OS **services** we can use to implement distributed applications
- **Design** and **implement** distributed applications using these services



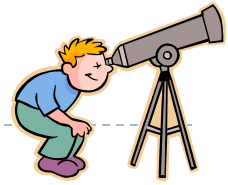
Step 3: Implementation

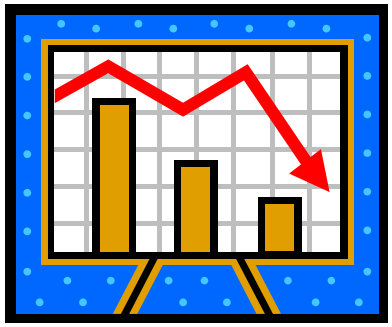




- What **requirements** does my application have?
- What is the most appropriate **communication paradigm**?
- How to design a communication **protocol**?
- What **data** does the application need to handle?
- What is the **naming convention** (machines, resources, services)?
- What aspects do we need to consider to realize the end-to-end communication (**data representation** , byte ordering, etc) ?

Step 4: … evaluate





- Which **parameters** are relevant for my specific application?
- What is the **performance** of my application and what is its latency?
- Does the application **scale** well?
- Does the application need to perform specific **error handling**?