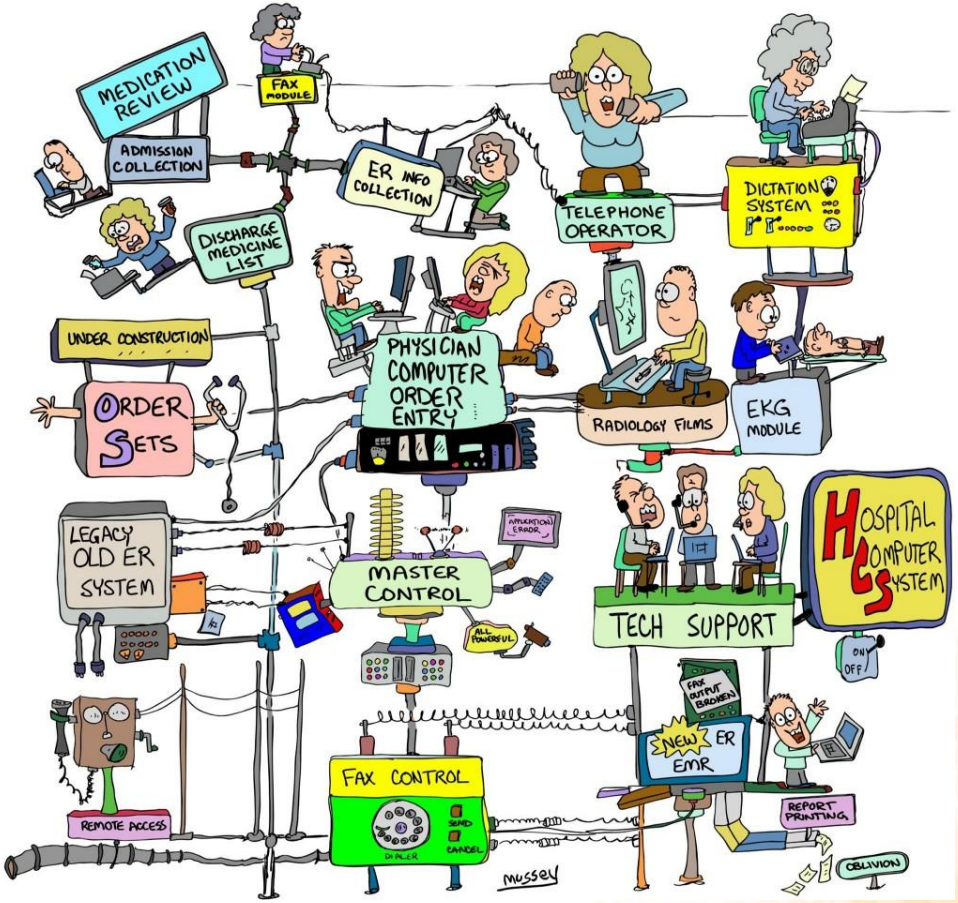
Distributed Programming

Lecture 01 - Introduction to Distributed Systems and Distributed Programming

Edirlei Soares de Lima
<edirlei.lima@universidadeeuropeia.pt>

What is Distributed Programming?

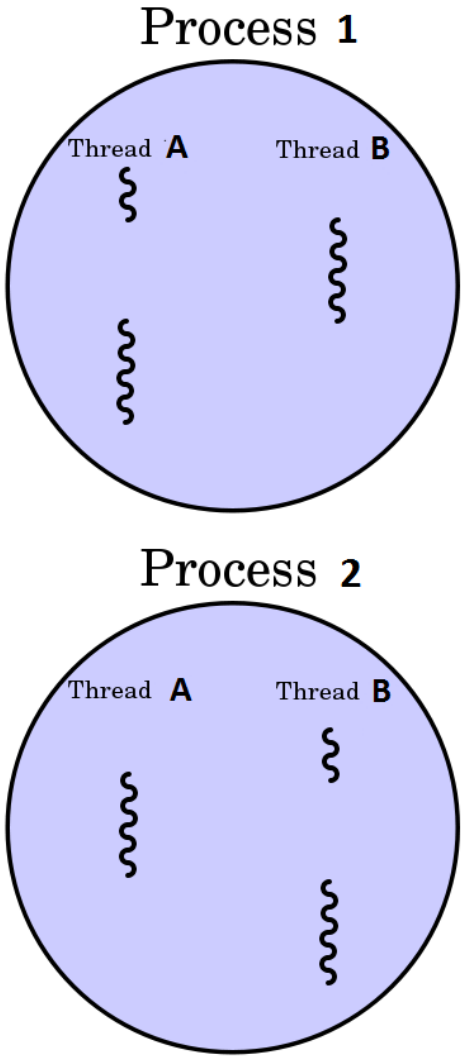
- Distributed computing is a field of computer science that studies distributed systems.
- A distributed system is a system whose components are located on different networked computers, which then communicate and coordinate their actions by passing messages to each other.
- Distributed programming involves the implementation of distributed systems.



Distributed Programming – Concurrency

- **Process vs. Thread:**
  - A process runs in its own address space (managed by the OS), while a thread runs within the address space of a single process and its managed by the process.

- **Parallel vs. Concurrent:**
  - Parallel refers usually to cases where two computation are taking place physically independently of each other (e.g.: in two different machines, or in two different processors) while concurrent is an abstraction that allows us to have apparent parallelism even in the cases of one processor.



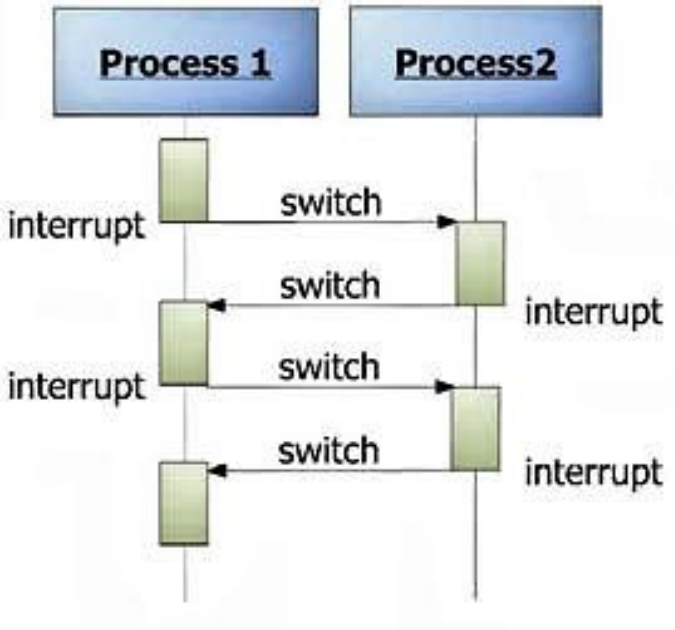
Concurrent Programming

- **Main challenge:** synchronize the execution of different processes and enable them to communicate with each other.

- Type of systems:
  - Multitasking System: concurrent execution of multiple processes in a single CPU.
  - Multiprocessor Systems: parallel execution of multiple processes in multiple CPUs.
  - Distributed Systems: parallel execution of multiple processes in multiple computers connected through a network.

Multitasking Systems

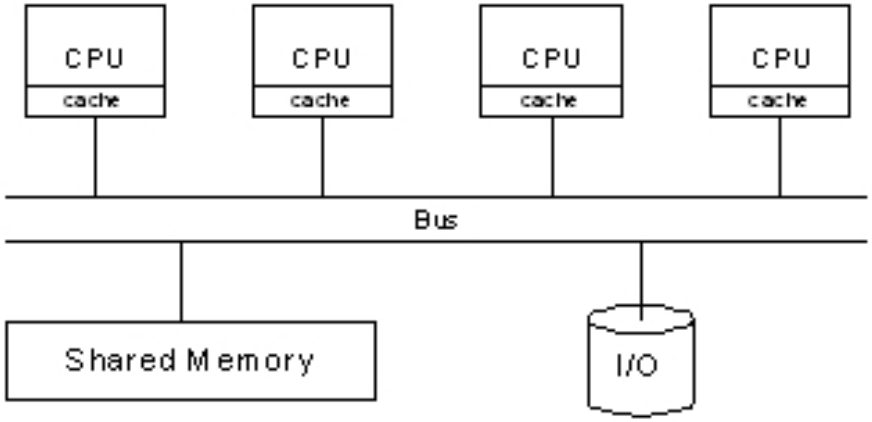
- Concurrency is based on interleaving instructions from different processes on a single CPU.
- Multitasking does not require parallel execution of multiple tasks at exactly the same time; instead, it allows more than one task to advance over a given period of time.



Multiprocessor Systems

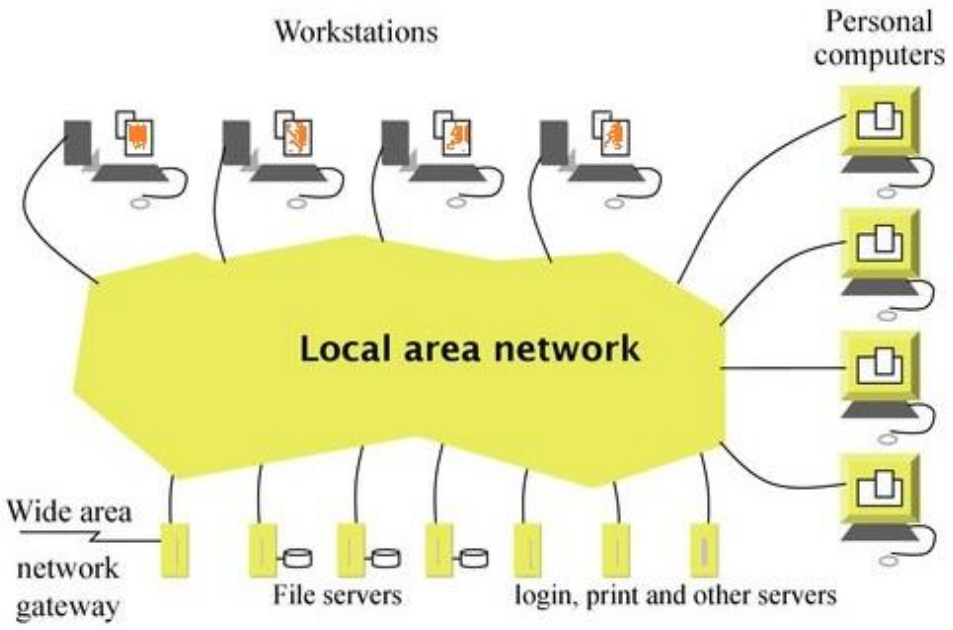
- Parallelism is achieved by executing processes in multiple CPUs on the same computer.

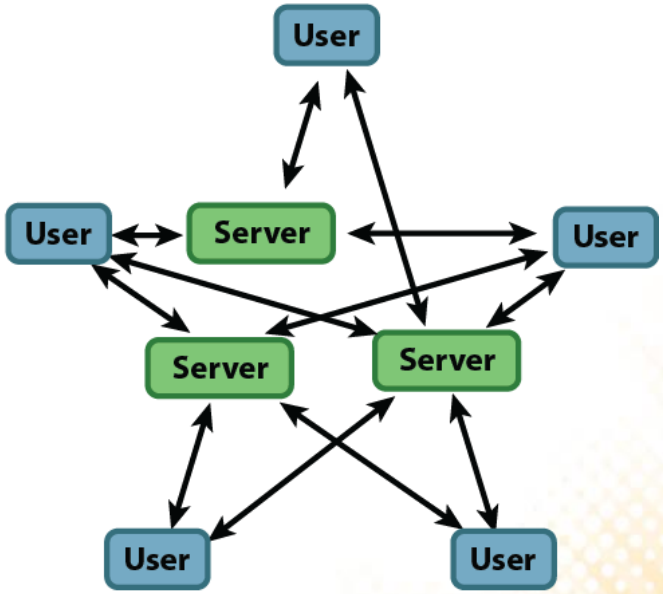
- Each processor has access to both local memory and global memory.



Distributed Systems

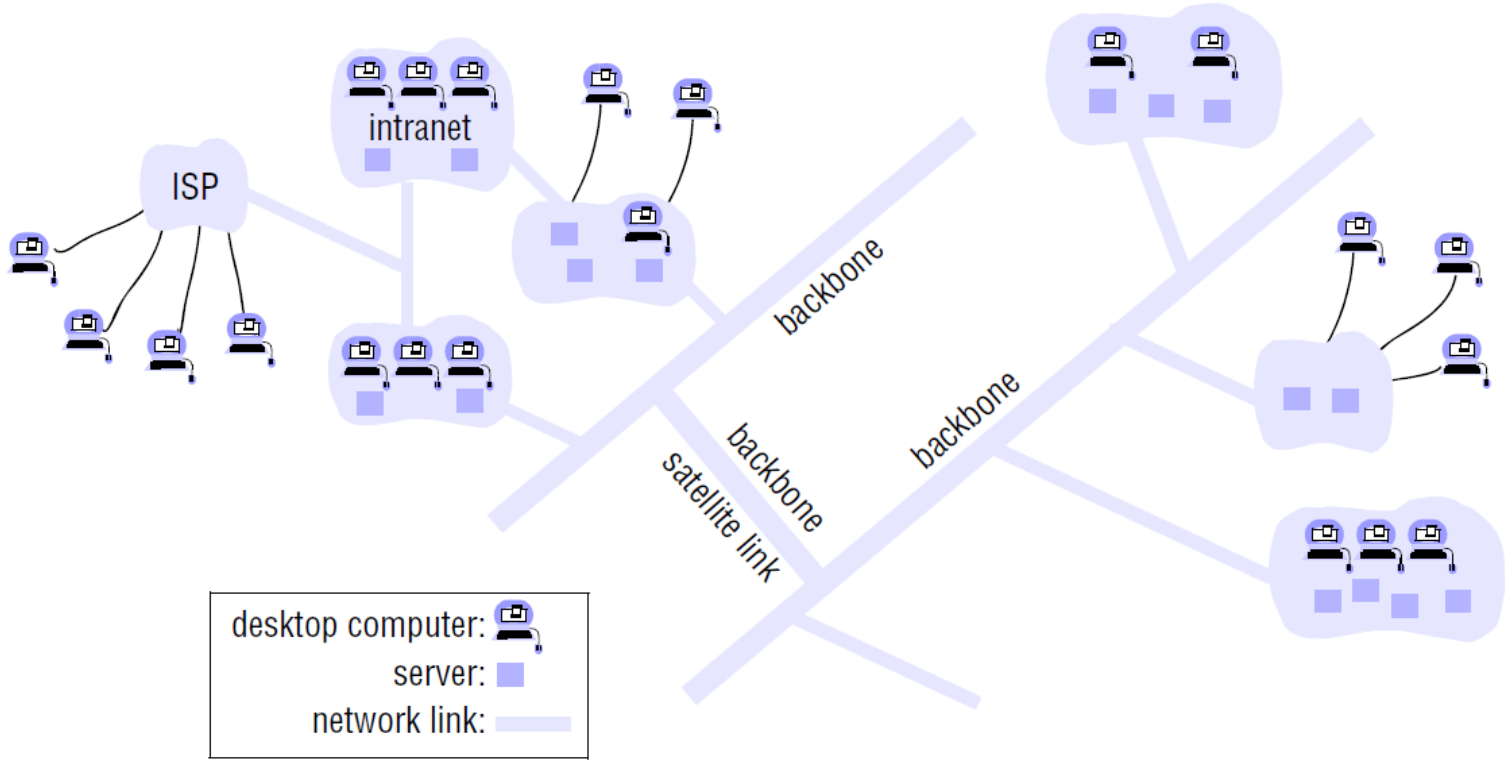
- Parallelism is achieved by executing processes in multiple computers that are connected through a network.





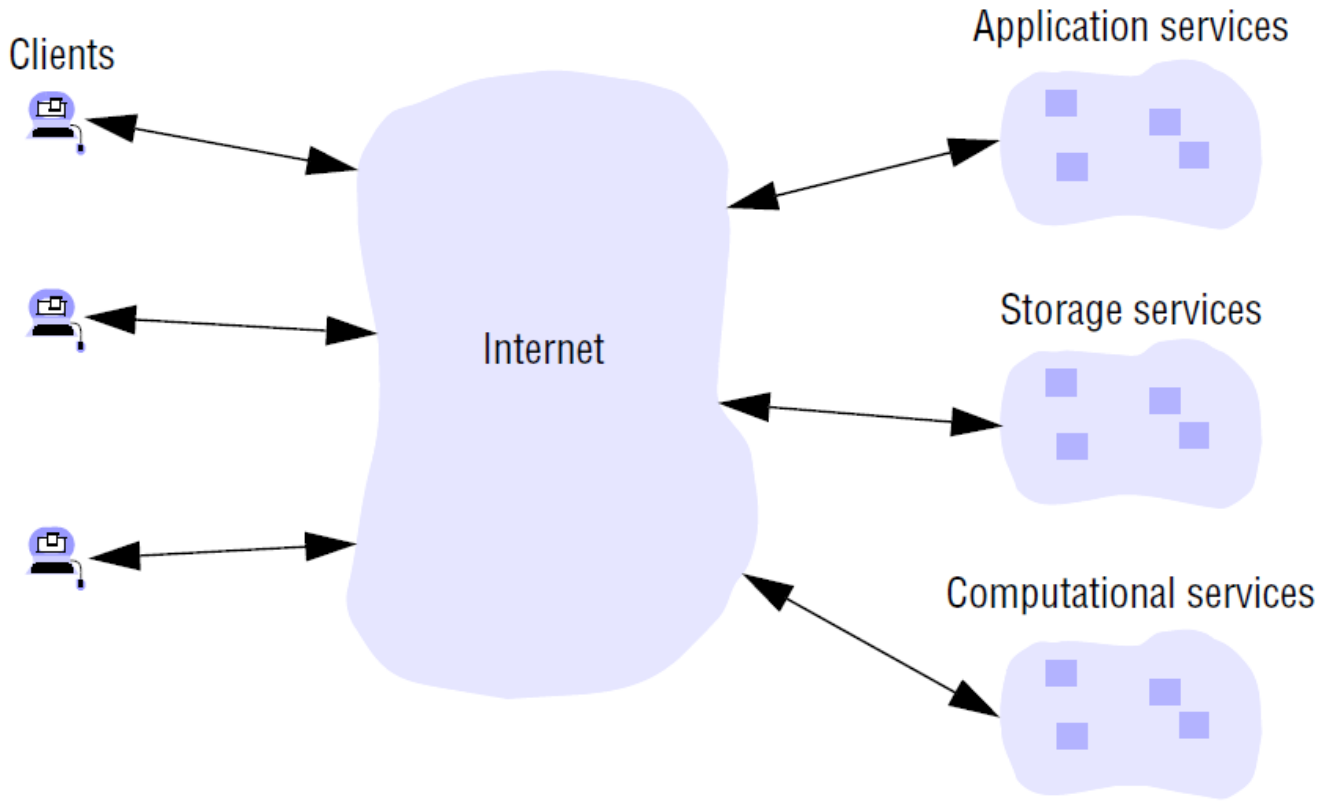
Distributed Systems – Examples

- Web Services: WEB (HTTP), E-Mail (SMTP), VoIP, Instant Messaging, …



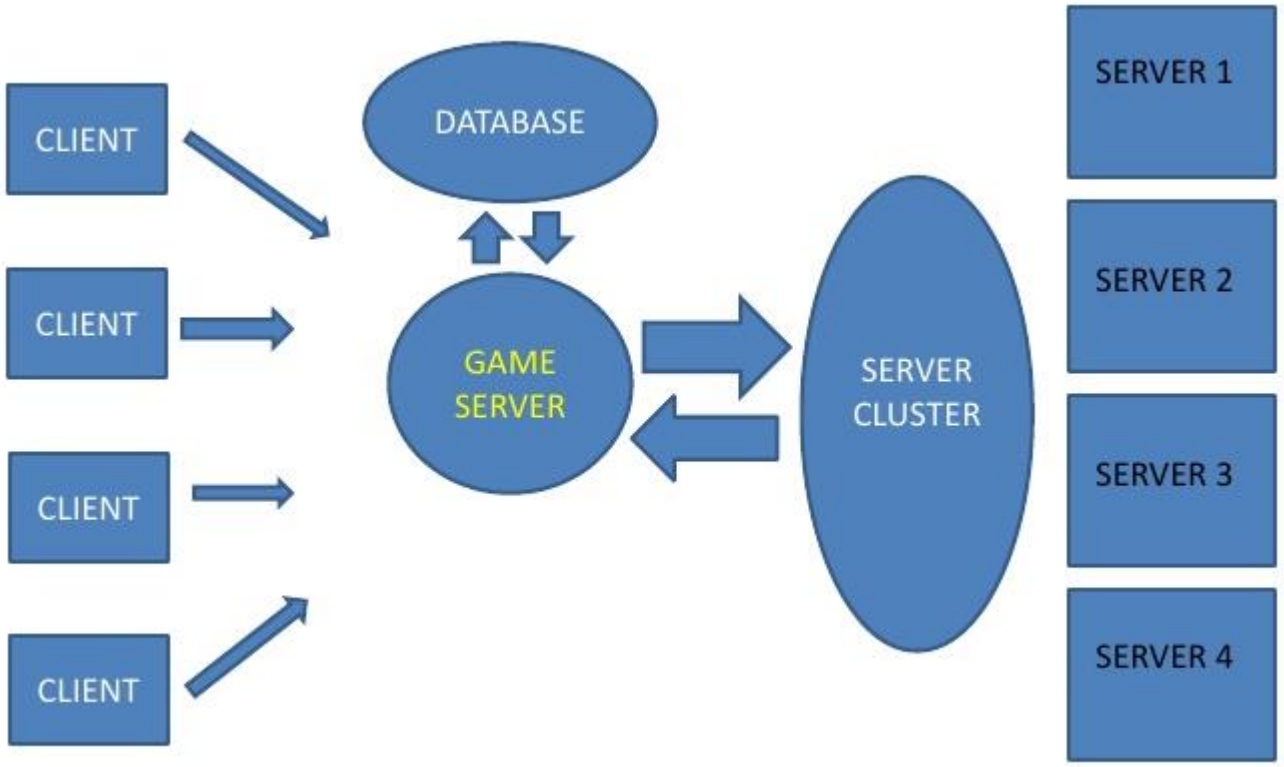
Distributed Systems – Examples

- Cloud Computing: Google Docs, Google Drive, Dropbox, Amazon Web Services, …



Distributed Systems – Examples

- MMORPGs: require fast response times and real-time propagation of events.



Multiplayer Games – History

- **Empire (1973):** turn-based strategy game with support for networked multiplayer.
- **Maze War (1973):** networked multiplayer first-person shooter.
- Both games were designed run on **small networks** composed of mainframe computers (PLATO system).

Multiplayer Games – History

- **Doom (1993)** was the progenitor of the modern networked games.
  - The first-person shooter supported up to four players in a single game session (in a **local area network – LAN**), with the option to play cooperatively or competitively.

Multiplayer Games – History

- **Quake (1996)** allowed players to connect to a server (which may be a dedicated machine or on one of the player's computers), where they could either play cooperatively or competitively.

- **Unreal (1998)** followed the same model of Quake with a multiplayer mode that allowed up to 16 players to play over the Internet.

Multiplayer Games – History

- **Ultima Online (1997)** was one of the first persistent MMORPGs.
- **EverQuest (1999)** was the first commercially successful MMORPG to employ a three-dimensional game engine.
- **World of Warcraft (2004)** is one of the most successful MMORPGs, with a peak of 12 million subscriptions in 2010.

Why are Distributed Systems Needed?

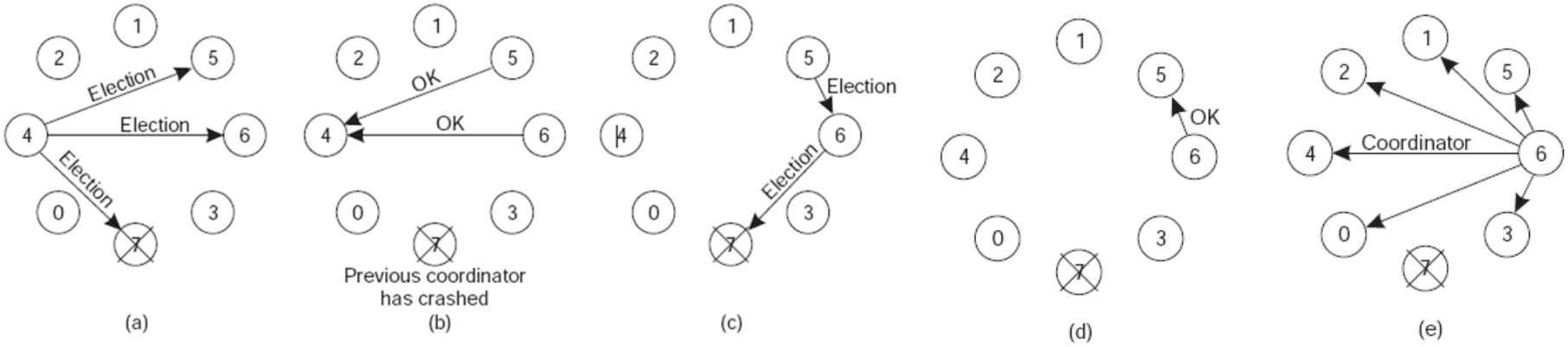
- General access without location restrictions (e.g.: banking network, games);
- Sharing resources across many users (hardware and software);
- Load balancing (e.g.: distributing game players across many servers instead of overloading a single server);
- Fault tolerance (when a fraction of the processors fail, the remaining processes can take over the tasks and keep the application running);
- Flexibility and adaptability by decomposing a global system into smaller (and simpler) systems;

Distributed Systems – Challenges

- Heterogeneity: hardware, operating systems, programming languages, …
- The knowledge of a process is local: no process is expected to have global knowledge about either the network topology or the global state.
- Communication, cooperation and synchronization between processes is done through message exchange.
- The handling of failures is an important and complex part of a distributed system.
- Scalability: a distributed system is considered scalable when its performance is not influenced by the final scale of the system or the number of users.

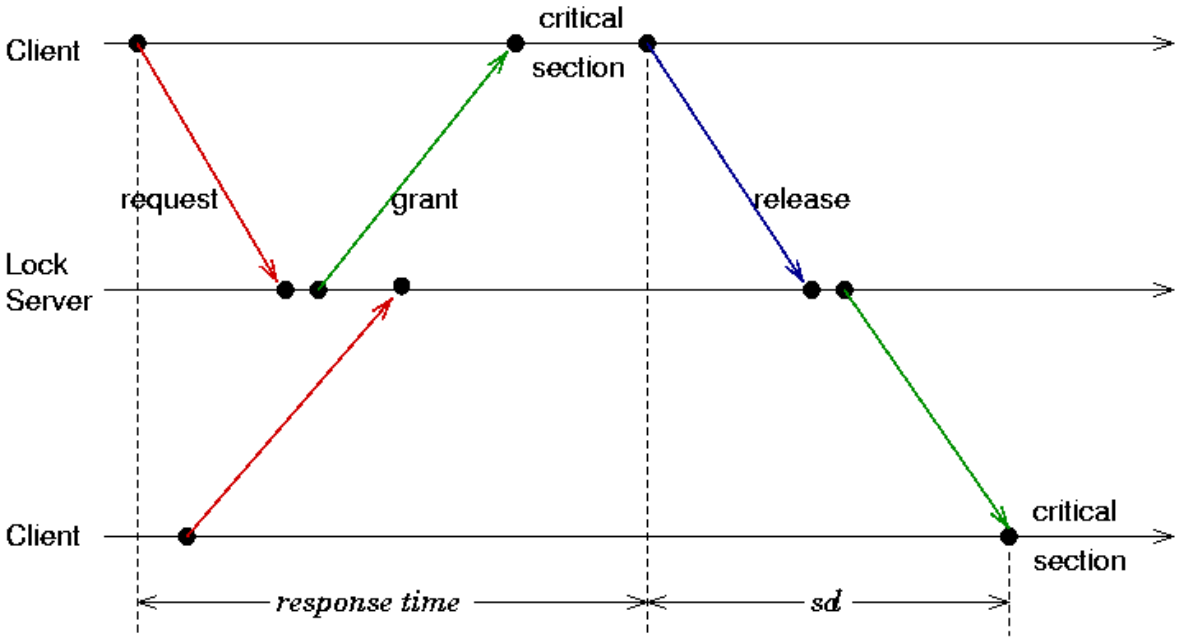
Distributed Computing – Common Problems

- Leader election: when a number of processes cooperate with one another for solving a problem, many implementations prefer to elect one of them as the leader and the remaining processes as followers. If the leader crashes, then one of the followers is elected the leader.



Distributed Computing – Common Problems

- Mutual exclusion: when the access to a resource or shared data is critical, it is necessary to guarantee that only one process will acquire the resource or perform critical operations on a shared data at any time.



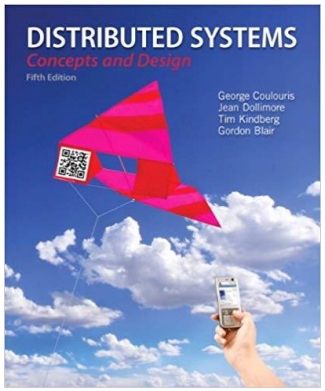
Distributed Computing – Common Problems

- Multicasting: sending of a given data to multiple processes in a distributed system is a common subtask in many applications. As an example, in group communication, one may want to send some breaking news to millions of members as quickly as possible.

- Replica management: to support fault tolerance and improve system availability, the use of process replicas is quite common. When the main server is down, one of the replica servers replaces the main server.

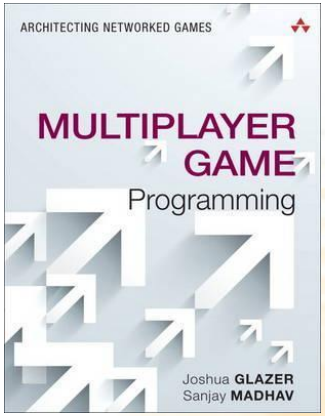
Further Reading

- Coulouris, G., Dollimore, J., Kindberg, T., Blair, G. (2004). **Distributed Systems: Concepts and Design** (5th edition), Pearson. ISBN: 978-0132143011.



  - **Chapter 1: Characterization of Distributed Systems**

- Glazer, J., Madhav, S. (2015). **Multiplayer Game Programming: Architecting Networked Games**. Addison-Wesley Professional. ISBN: 978-0134034300.



  - **Chapter 1: Overview of Networked Games**