Universität Stuttgart
Master of Science

Distributed Systems Models

Marco Aiello

Distributed System Definition

from Coulouris et al.

A system model is necessary to precisely and uniquely specify the relationship between different components of a distributed system and its behaviour as a whole. Different aspects of a distributed system need modeling:

- *Computation state* (e.g, consistency)
- *Architecture* (e.g., client-server)
- *Interaction* (e.g., asynchronous messages)
- *Failure* (e.g., types of channel exceptions)
- *Security* (e.g., types of attacks to a host)

Modelling the state of the computation

- State machine to model a system who's output depends on the current state and input (e.g., incoming messages)

- Finite State Machine (FSM):
  - S is a finite non-empty set of states;
  - $s_0 \in S$ is the initial state;
  - I is a set of input messages
  - O is a set of output messages
  - $\delta : S \times I \rightarrow S \times O$ is the state transition function

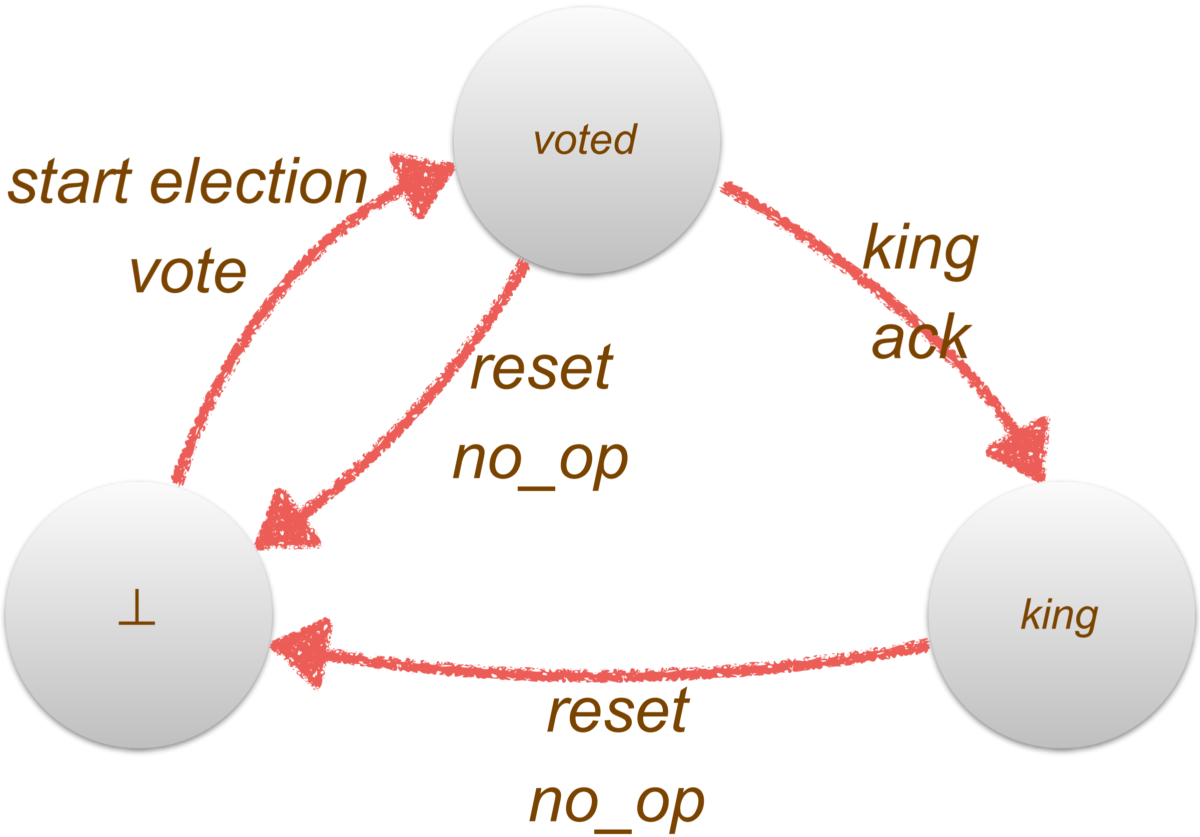
Finite State Machine

Example

- S={⊥,voted, king}
- s0 = ⊥
- I = {start election, king, reset}
- O = {vote, ack, no_op}
- Next state function:
    - (⊥,start election) -> (voted,vote)
    - (voted, king) -> (voted, ack)
    - (voted,reset) -> (⊥,no_op)
    - (king, reset) -> (⊥,no_op)

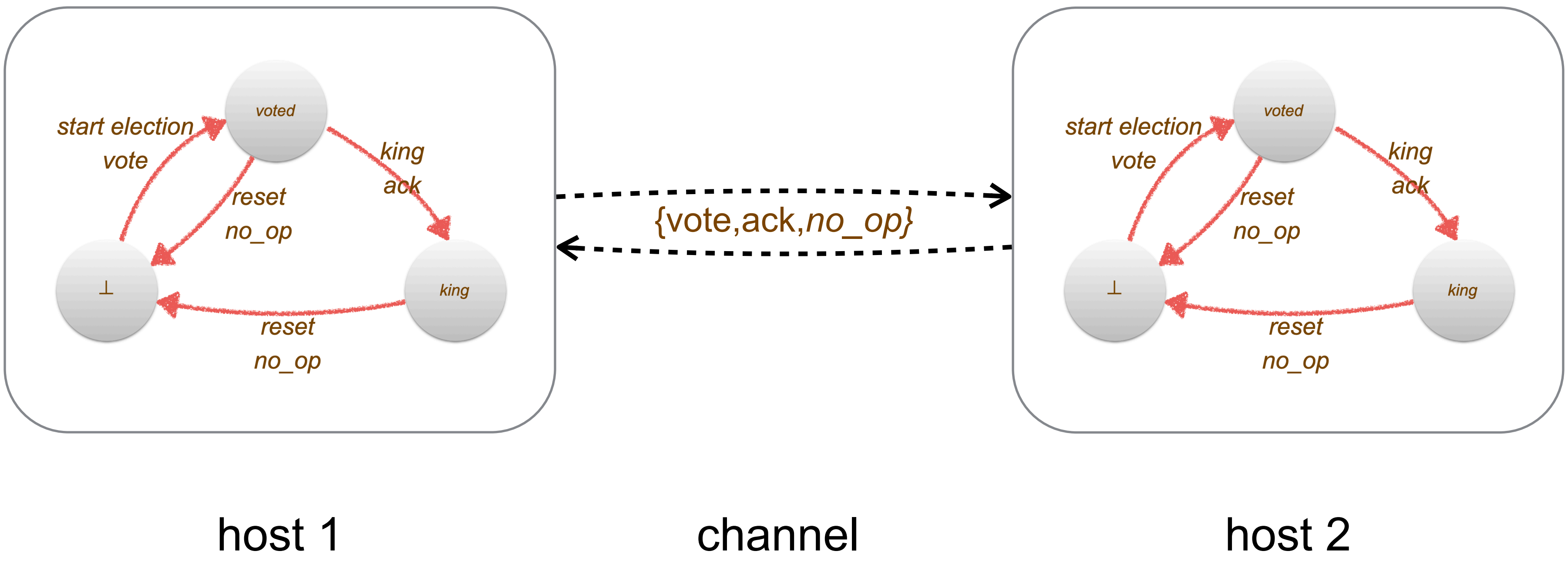
Finite State Machine

Example schema



State of a Distributed System

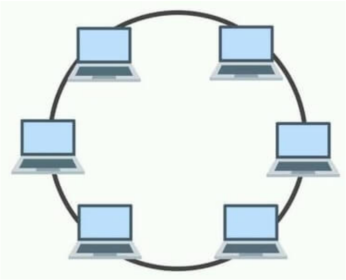
- A distributed system is modelled by the set of states of its components and of its channels



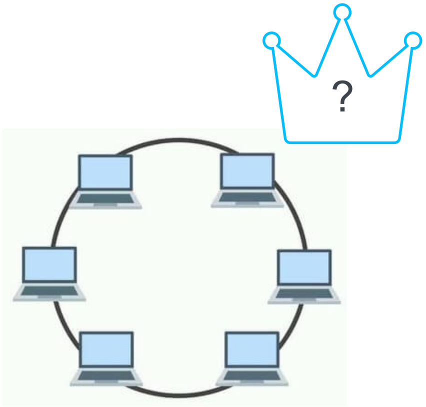
Some important (negative) results

- No leader election in anonymous rings
- No consensus in asynchronous systems
- Byzantine fault tolerance with at most $\lceil \frac{1}{3}n - 1 \rceil$
- CAP theorem (Consistency, Availability, Partitioning Tolerance)

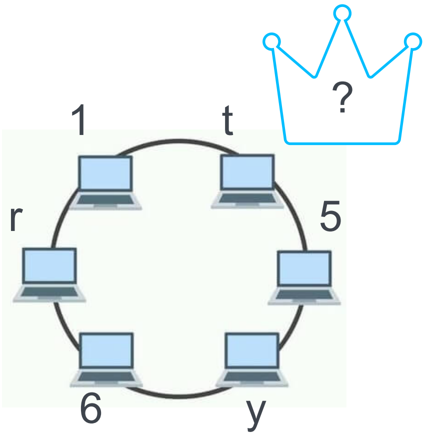
In pictures



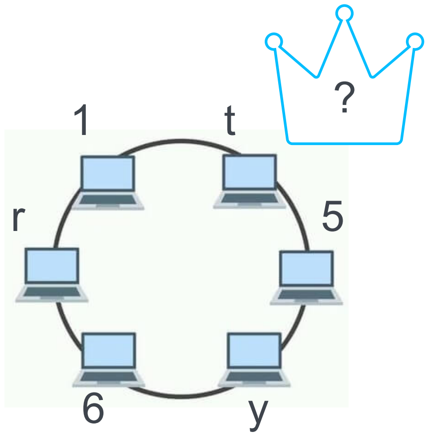
In pictures

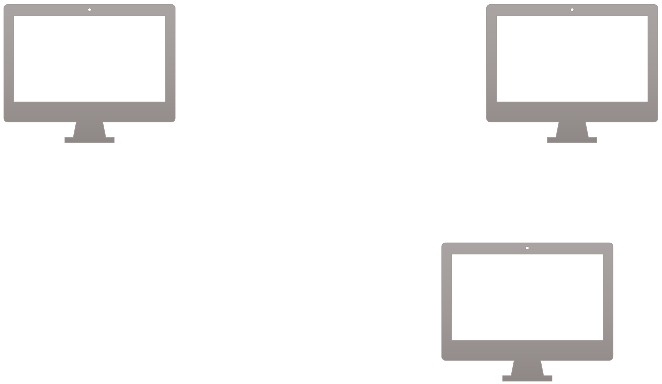


In pictures

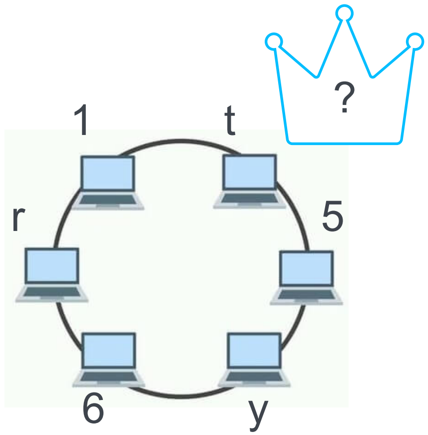


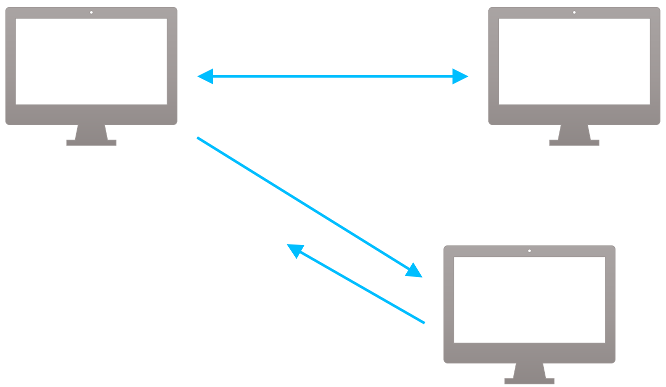
In pictures



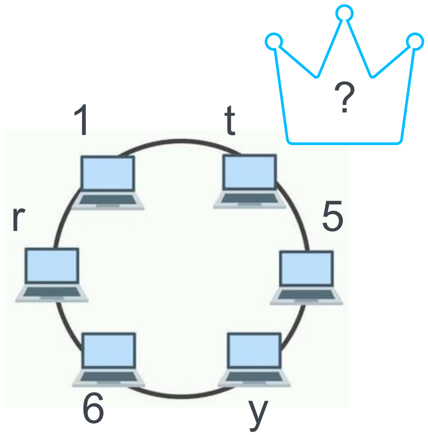


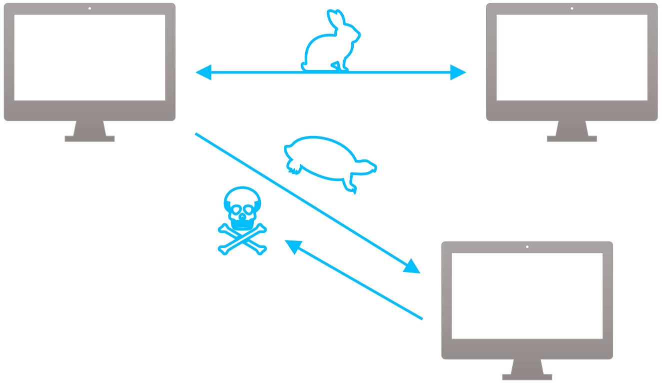
In pictures



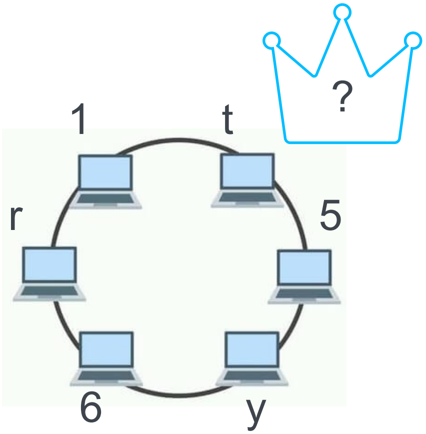


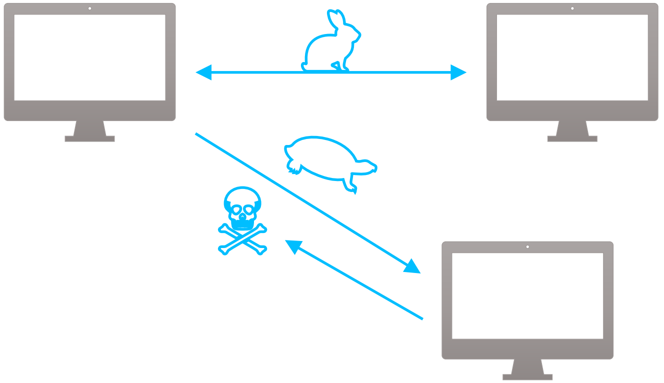
In pictures

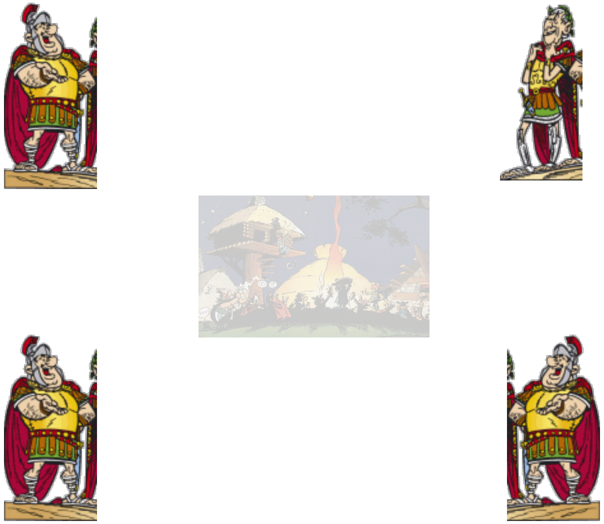




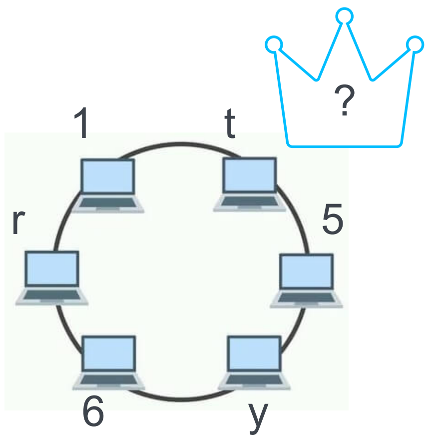
In pictures

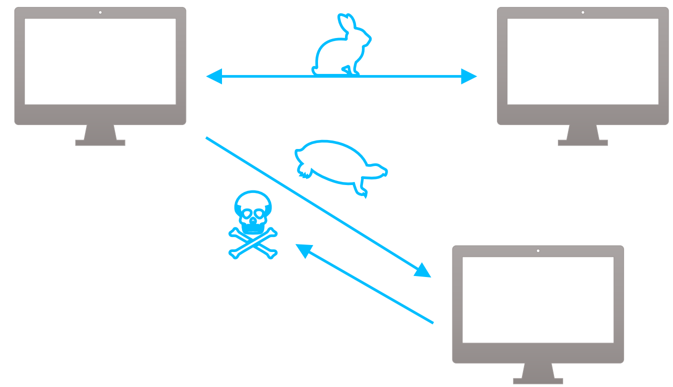


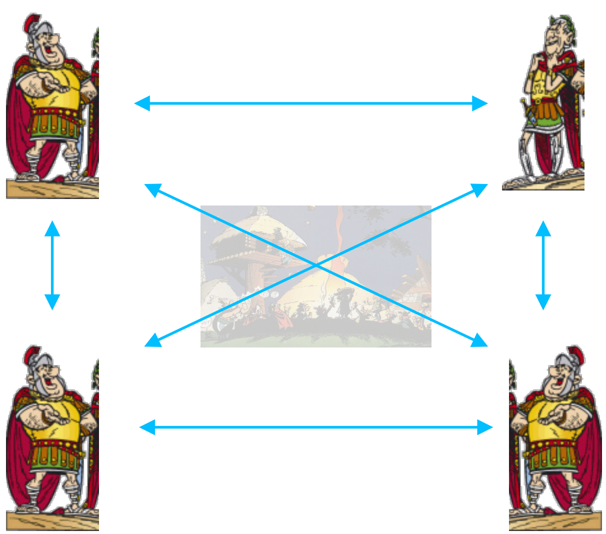




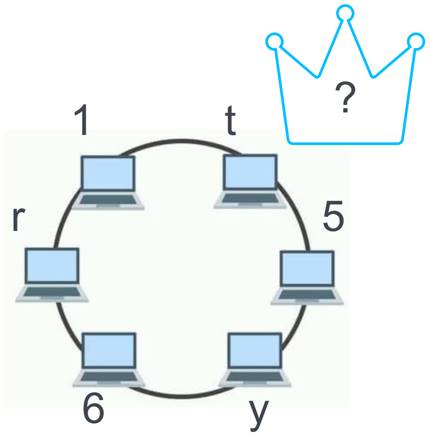
In pictures

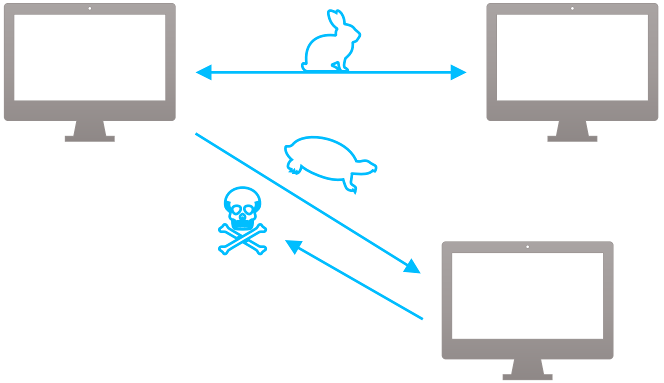


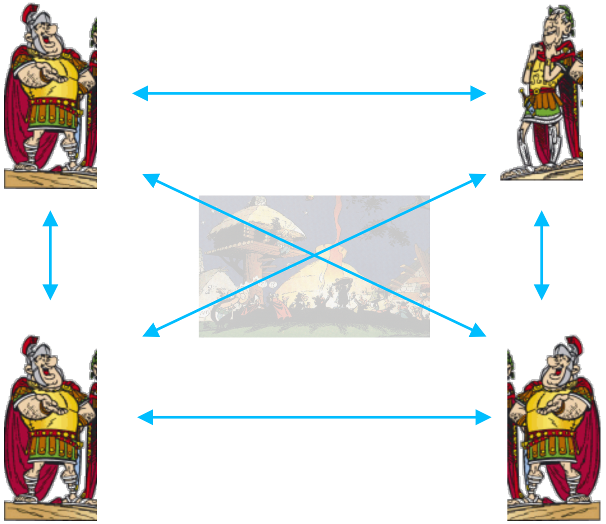


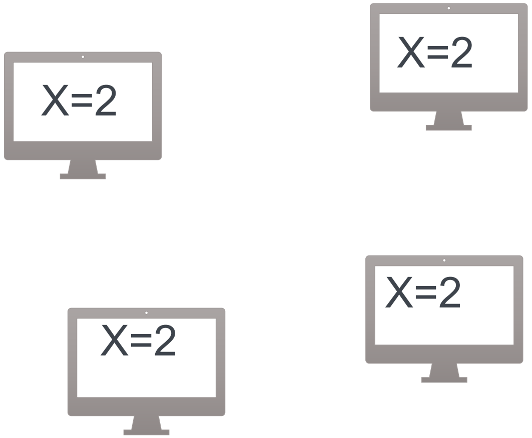


In pictures

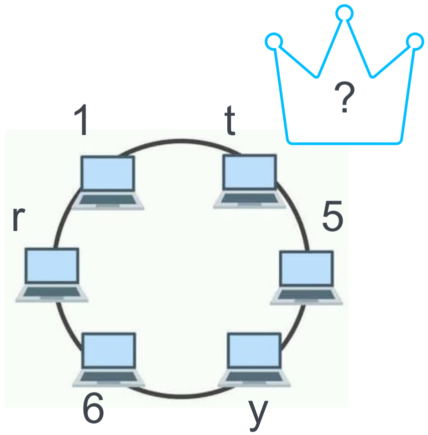


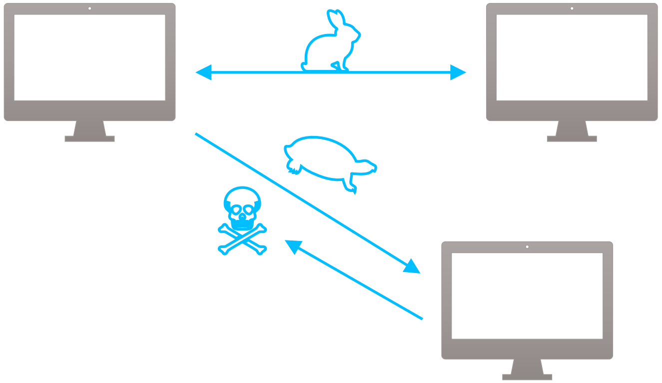


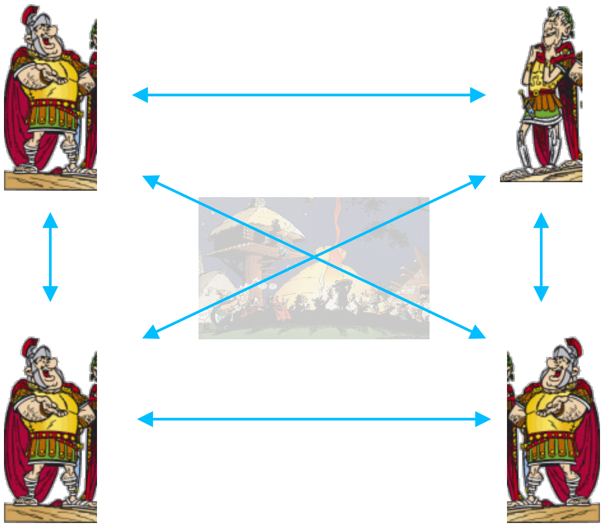




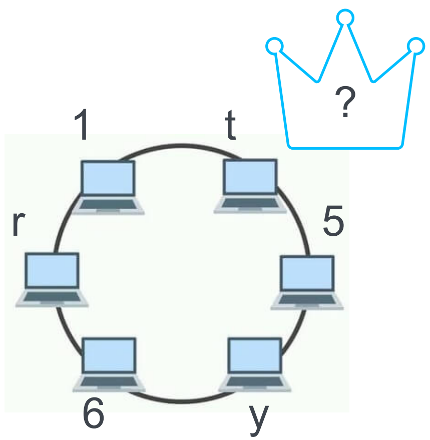
In pictures

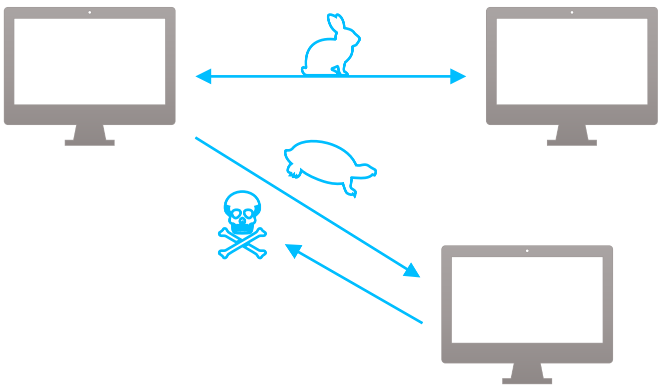


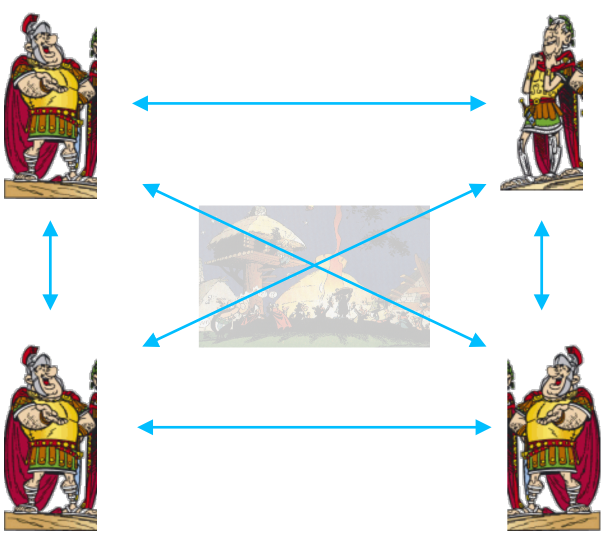




In pictures



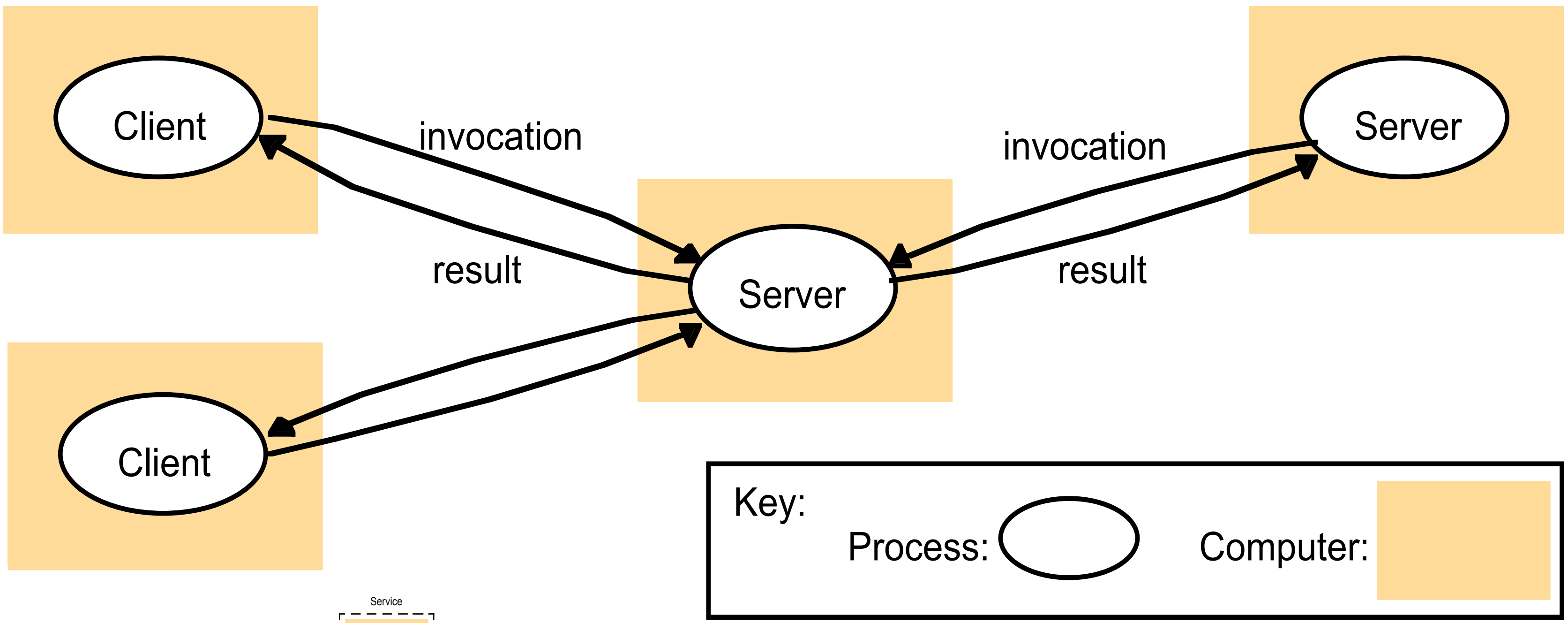


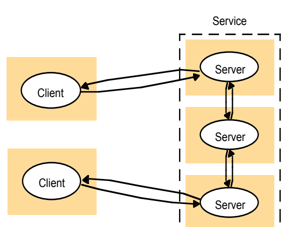


Architectural Models

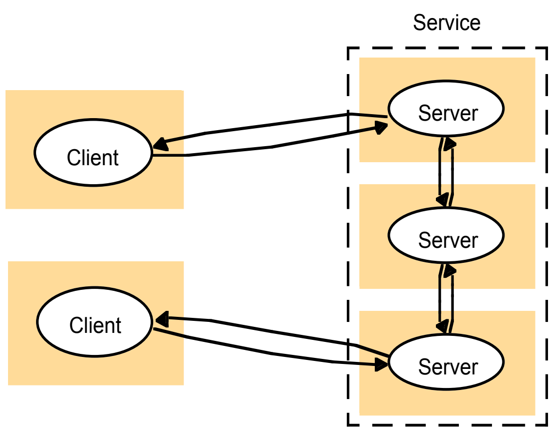
- **Client-server**
  - *Architectural variations:*
    - One server-many clients
    - Many servers-many clients
    - Intermediaries: proxies, load balancing mediators
  - *Computational load variations*
    - Mobile code (applets, scripts)
    - Mobile agents
    - Network computers
    - Thin clients
- **Peer-to-peer**

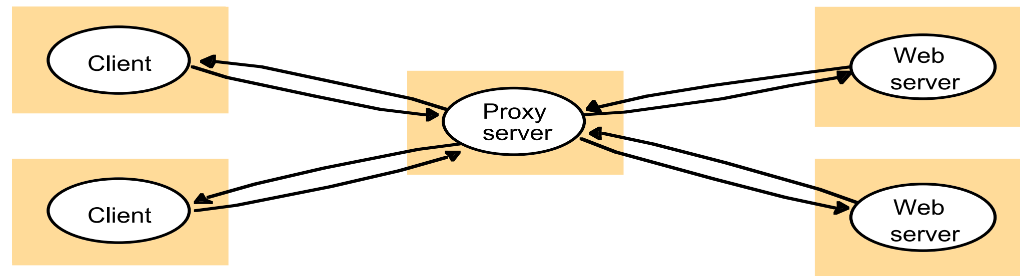
Client Server



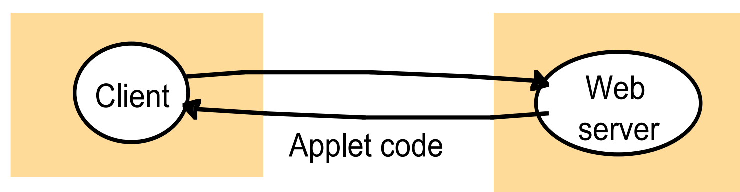


Client Server Variations



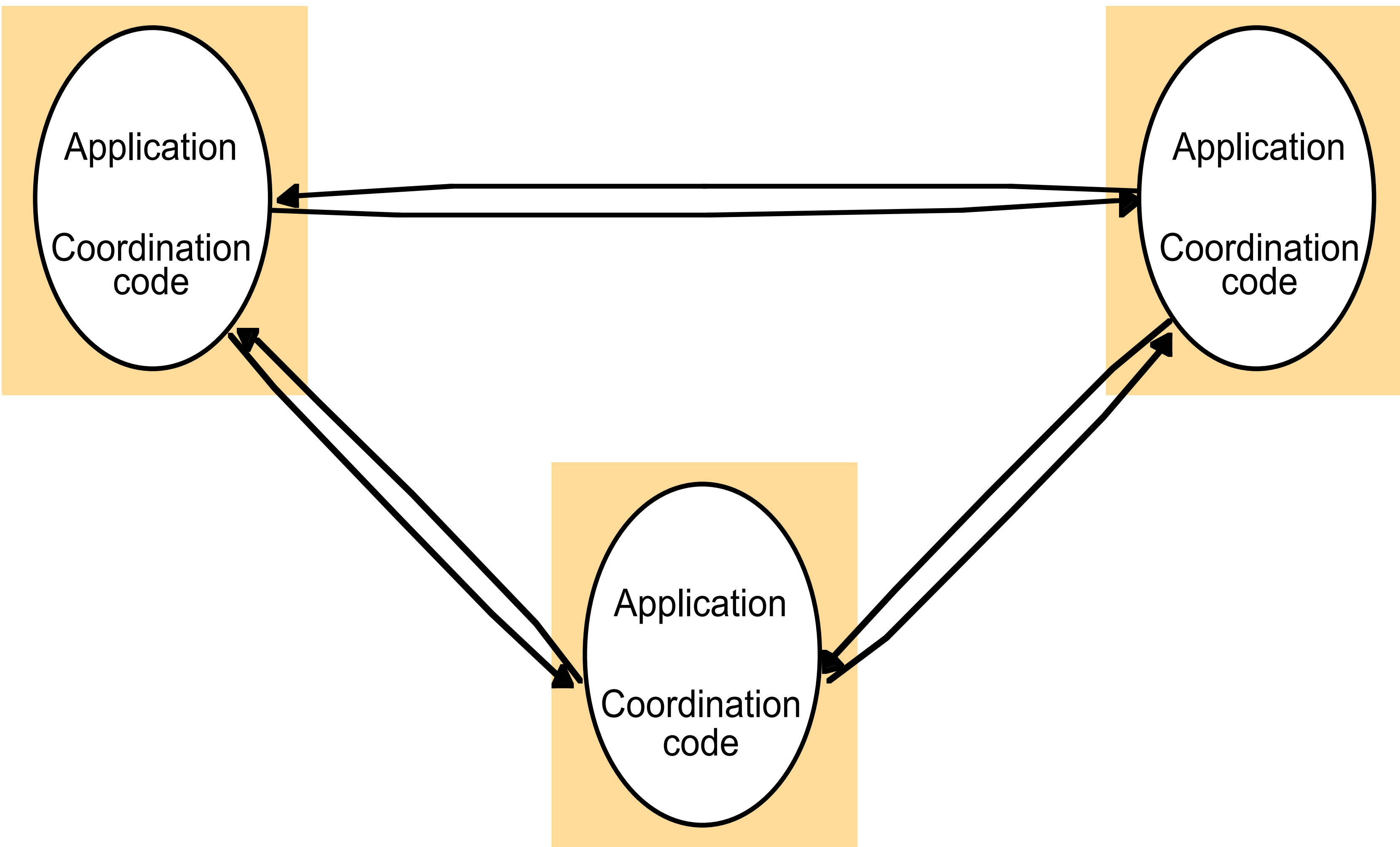


a) client request results in the downloading of applet code



b) client interacts with the applet

Peer-to-peer


Application
Coordination code
Application
Coordination code
Application
Coordination code

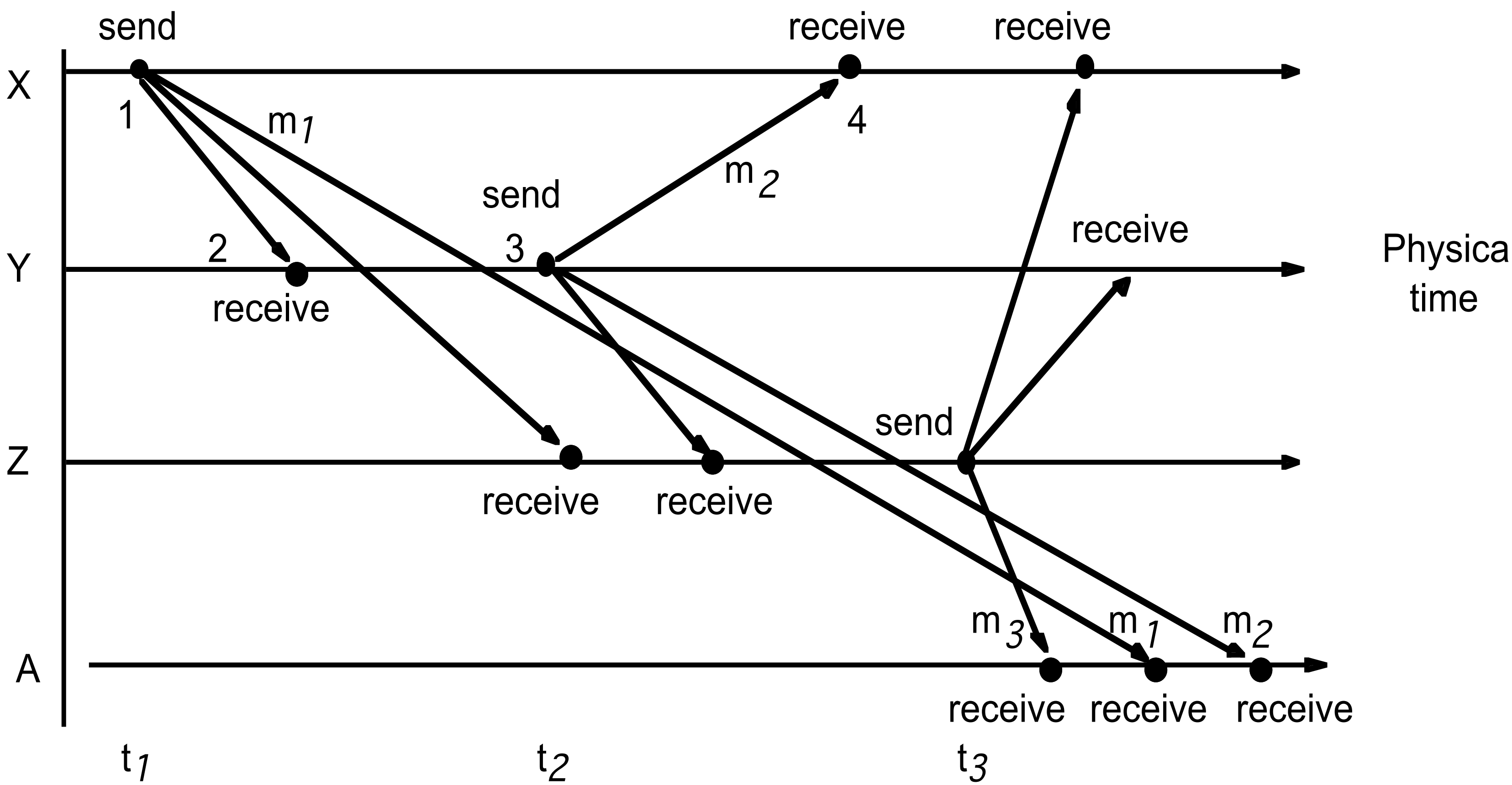
Interaction model

- **Synchronous** distributed system
  - Time to execute a step has lower and upper bounds
  - Each message is received within a given time
  - Each process has a local clock with a given max drift

- **Asynchronous** distributed system
  - No bounds on process execution time
  - No bounds on message receival time
  - Arbitrary clock drifts

Synchronous = Blocking is often used, but not the definition, rather a consequent effect

Example: eMail

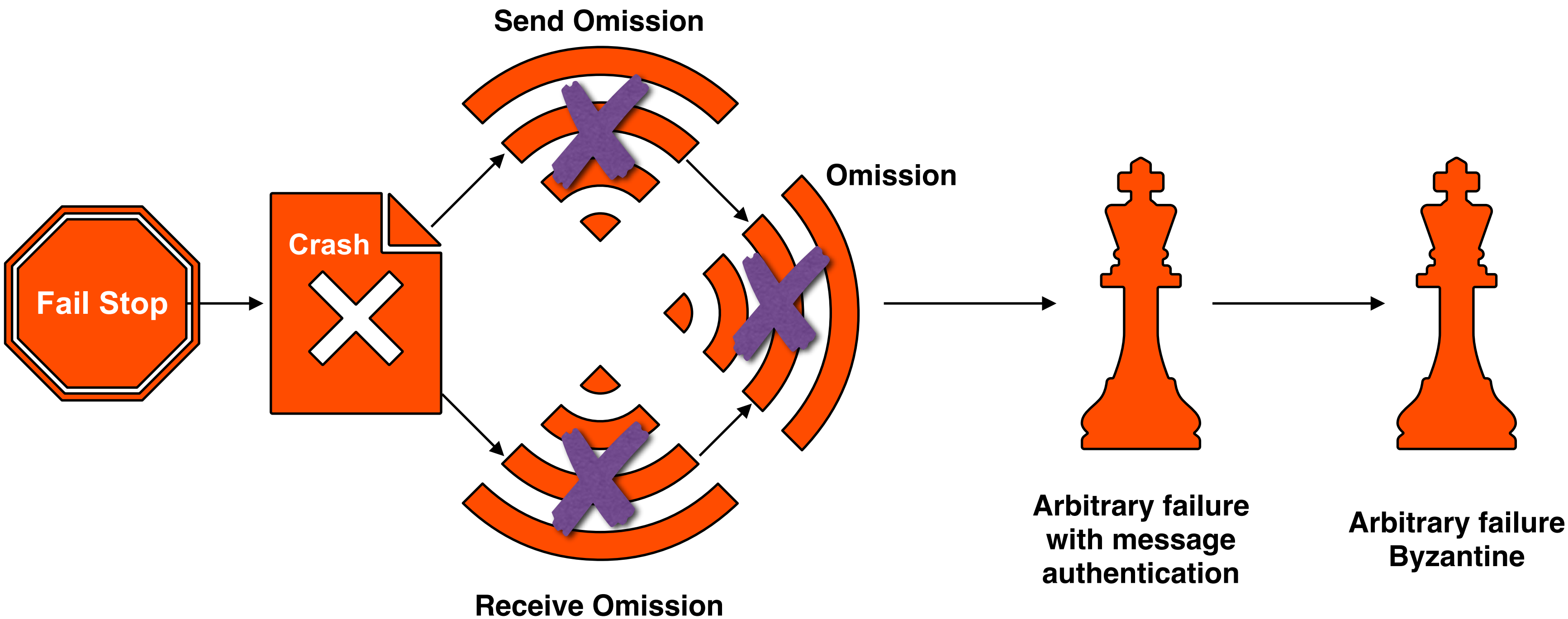


Interaction model

- Performance
  - *Latency*: Δ time transmission begins and beginning of receipt
  - *Bandwidth*: amount of information that can be transmitted over a channel in the unit of time
  - *Jitter*: difference in time needed to transmit a series of messages
- Clock drift rate: timing events (GPS)
- Event ordering

Failure model

- Defines how failures can occur in order to detect and mask them, thus establishing the level of fault tolerance of the system.



Security Models

- Model the enemy’s attacks
    - list type of attacks, to which components, for which users and processes
- Goal:
    - To protect objects, processes, communication, data integrity and privacy
- Solutions:
    - Cryptography
    - Authentication
    - Secure channels
    - Data mixing and obfuscation