System Models

Marco Aiello

(based on http://www.cdk5.net)

System models

- 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., asyncronus messages)
  - Failure (e.g., types of channel exceptions)
  - Security (e.g., types of attacks to a host)

2

Modeling state of 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):
  - Finite set of states S $\supseteq$ {initial state s_0}
  - Set of inputs messages I
  - Set of outputs messages O
  - Next state function: $f: S \times I \rightarrow S \times O$

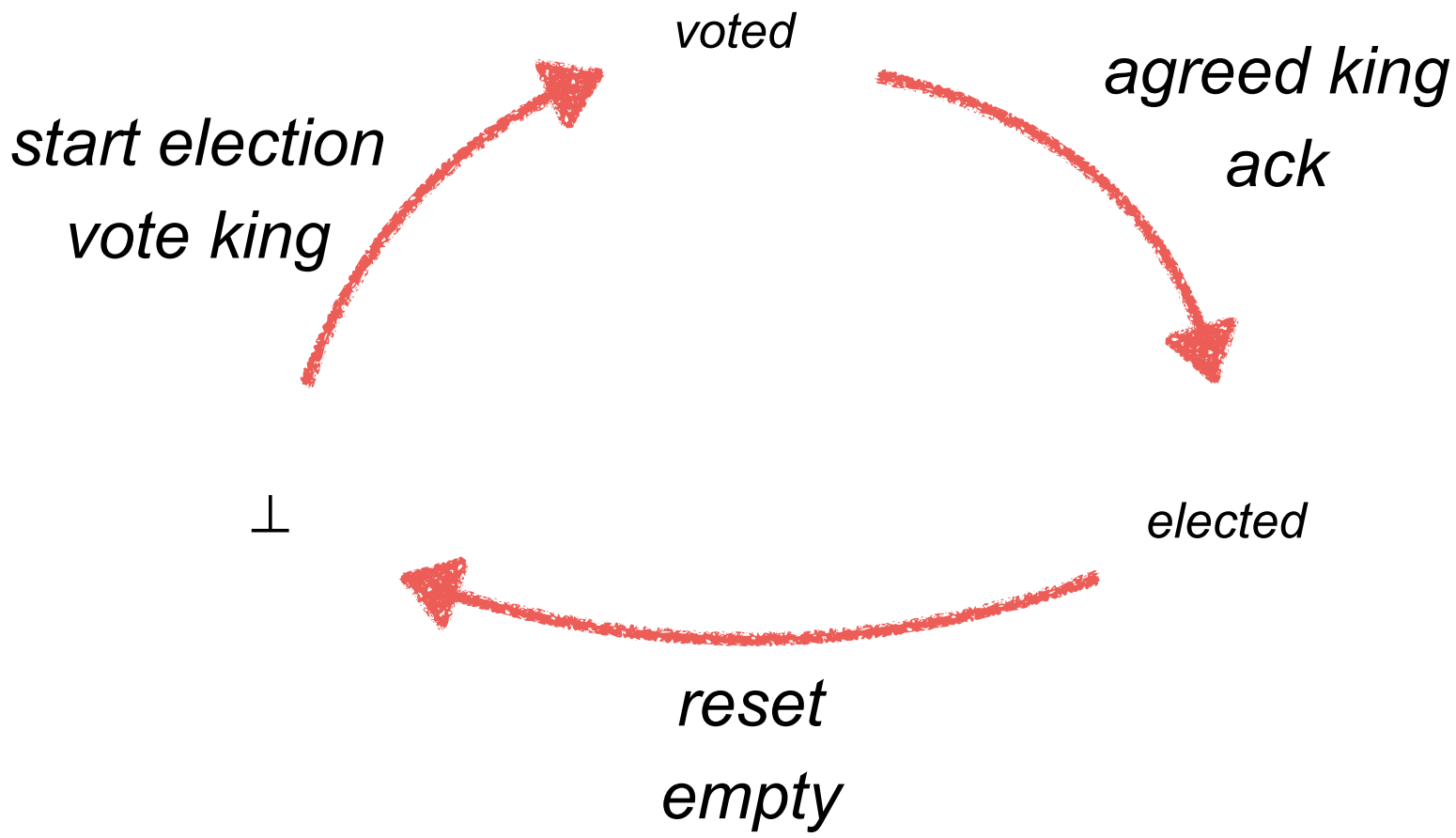
3

FSM Example

- *S={⊥,voted,elected}*
- s_0 = ⊥
- *I = {start election, agreed king, reset}*
- *O = {vote king, ack, empty}*
- Next state function:
  - (⊥,*start election) -> (voted,vote king)*
  - (voted,*agreed king) -> (elected,ack)*
  - *(elected,reset)* -> (⊥,*empty)*
  - any other combination - > (⊥,*empty)*

4

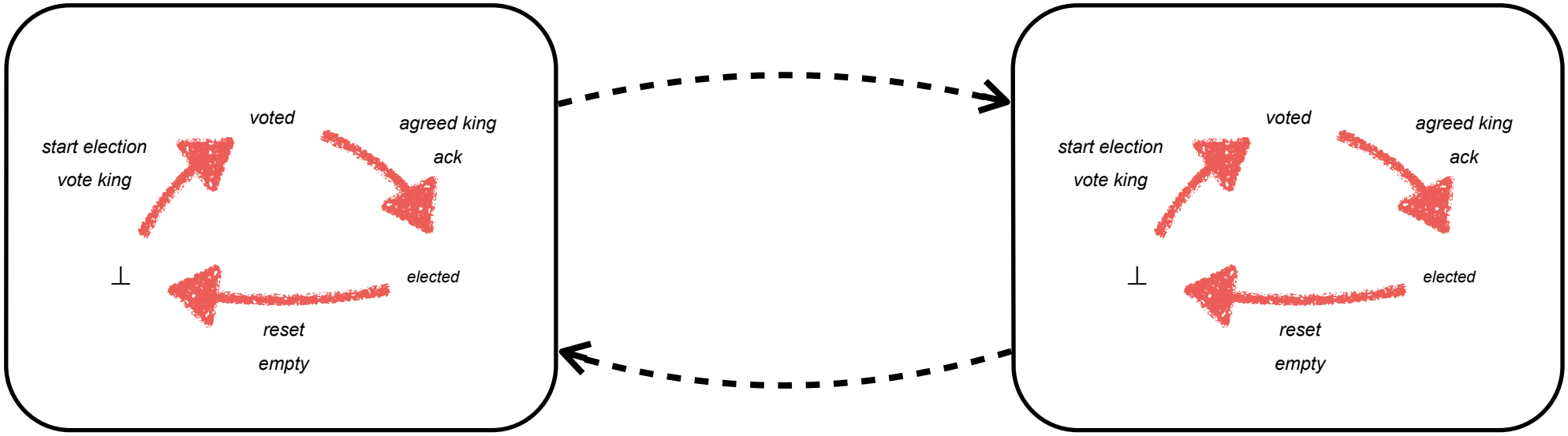
FSM Example



5

State of a distributed system

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



host 1 channel host 2

6

Software layers

- A distributed system can be modeled in terms of different layers:
  - Application layer
  - Middleware
  - OS layer

- Lower layers are more heterogeneous
- Higher layers give the feeling of the distributed system as a whole

7

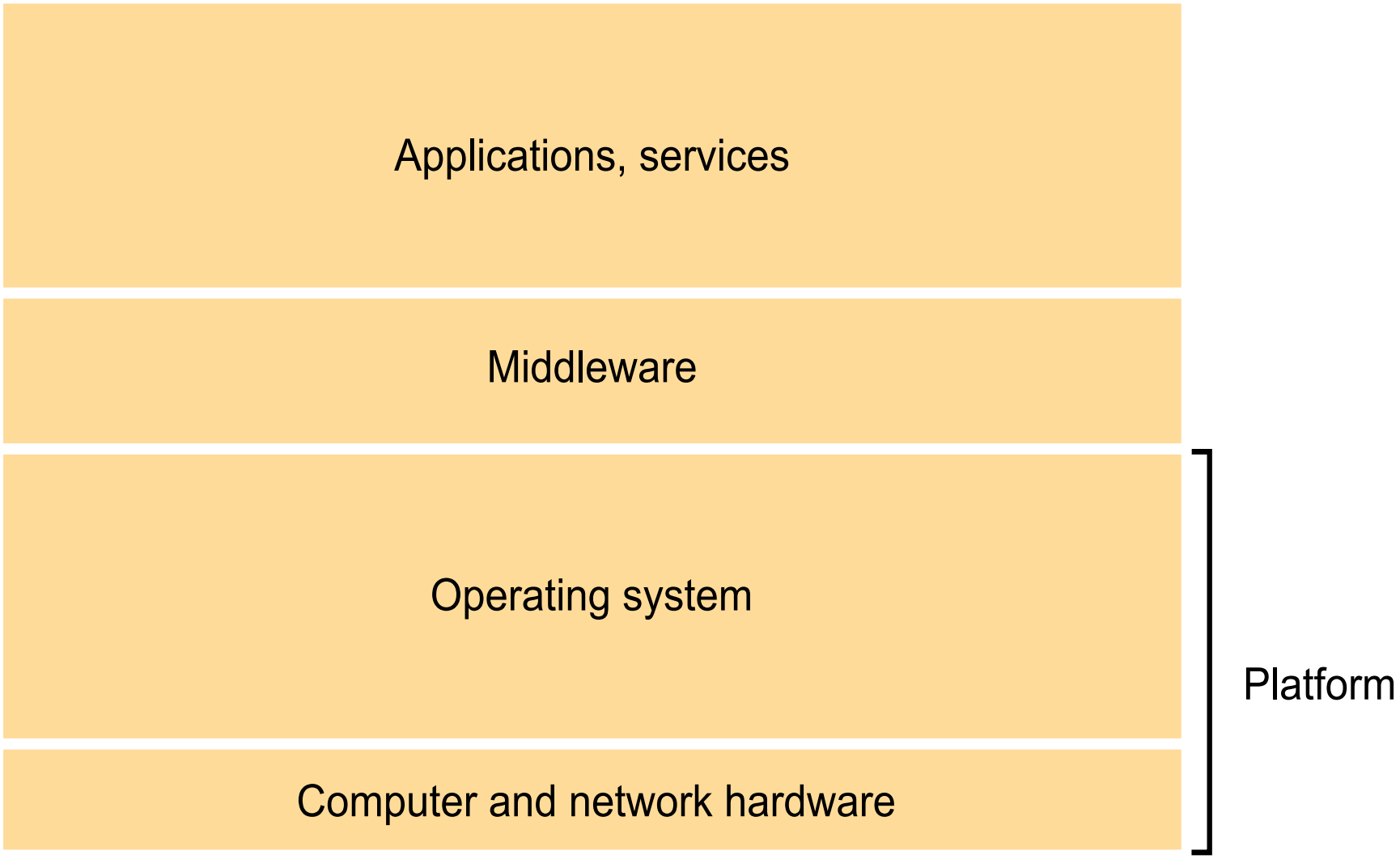
Middleware

- Middleware: software layer which abstracts from network hardware, computing hardware, operating systems, programming languages and different implementations providing a uniform computational model

- Examples, CORBA, Java RMI, DCOM, …

8

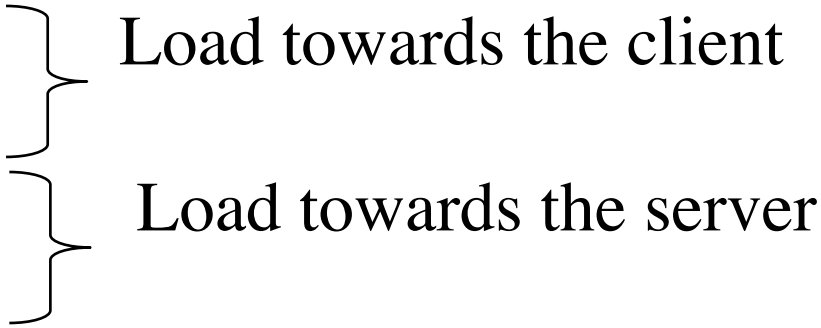
Software and hardware service layers



9

Architectural models

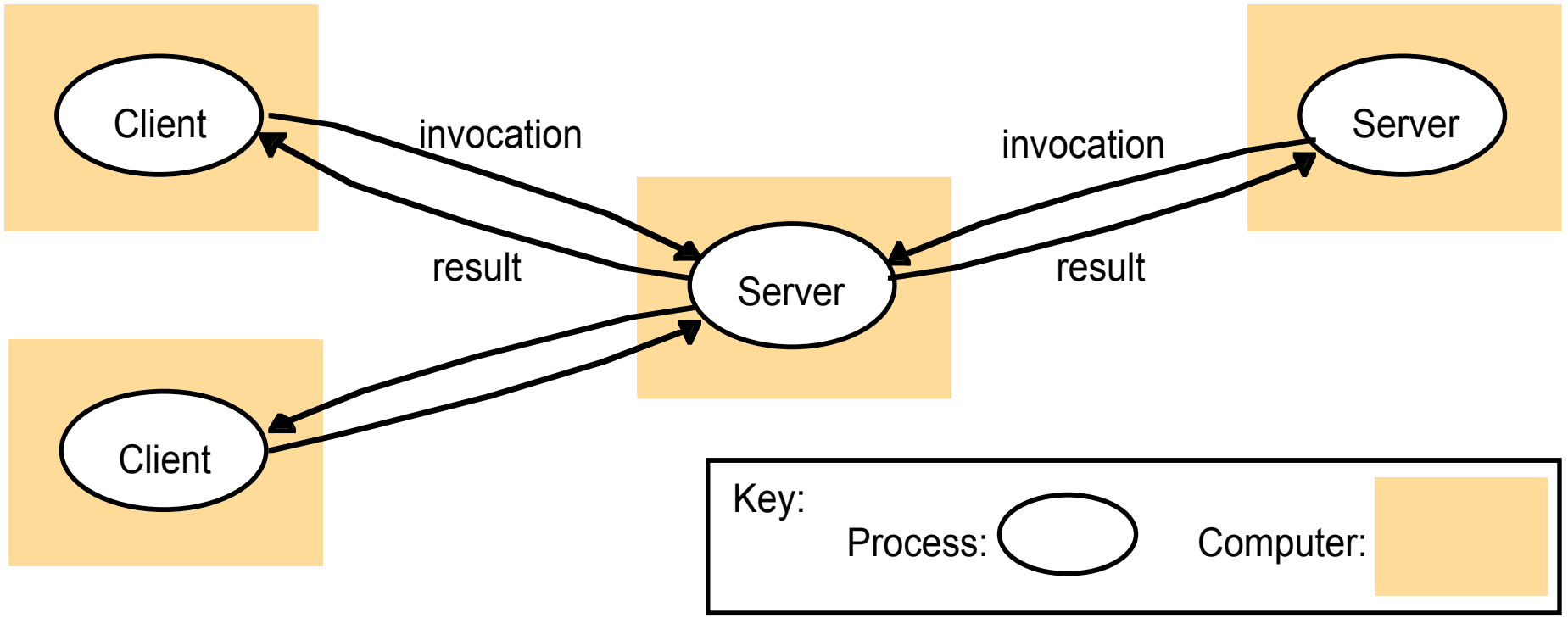
- Client-server
  - Architectural variations:
    - One server-many clients
    - Many servers-many clients
    - Intermediaries: proxies, load balancing mediators
  - Computational load variations
    - Mobile code (applets, scripts)
    - Mobile agents
    - Network computers
    - Thin clients



- Peer-to-peer

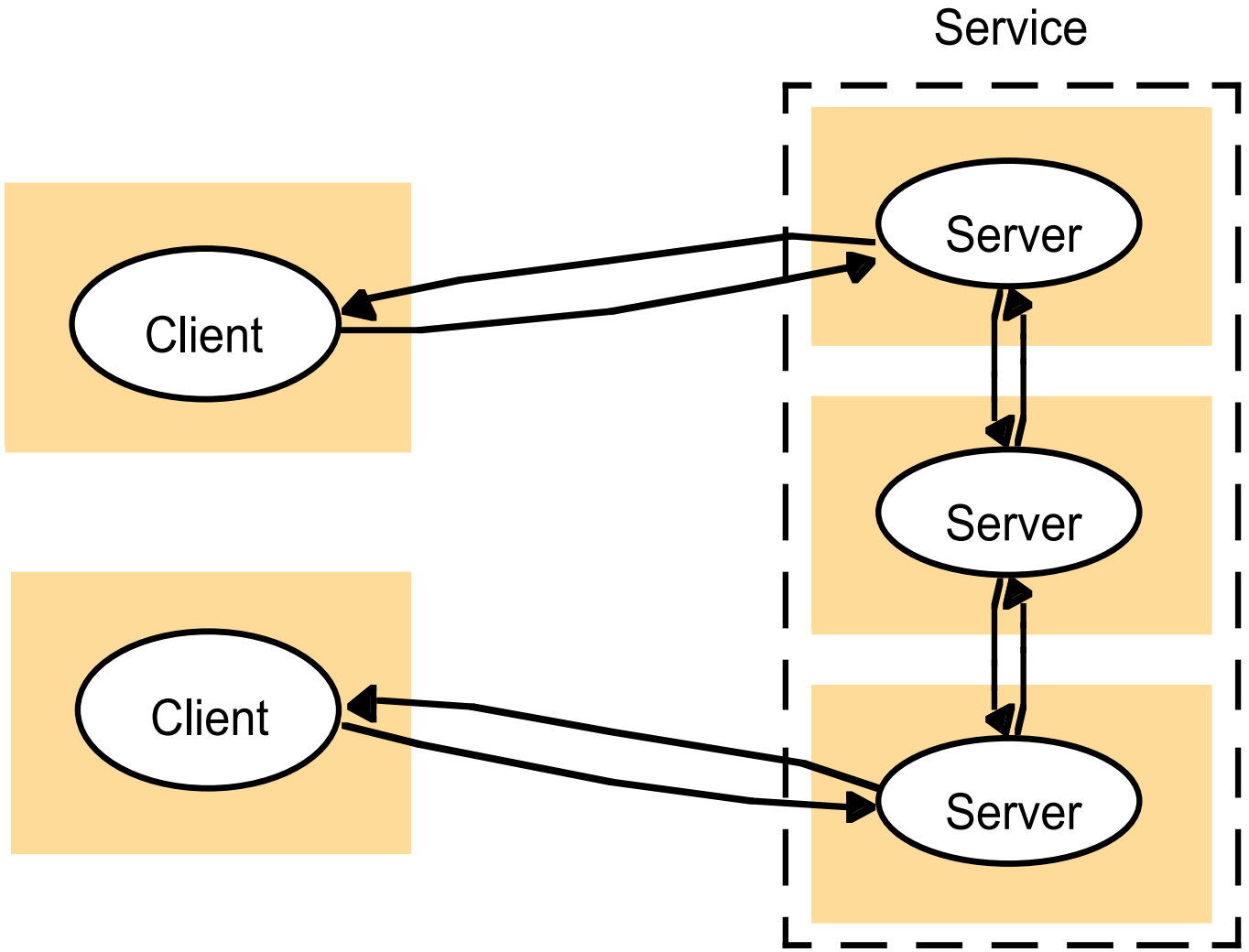
10

Architectural model: client-server



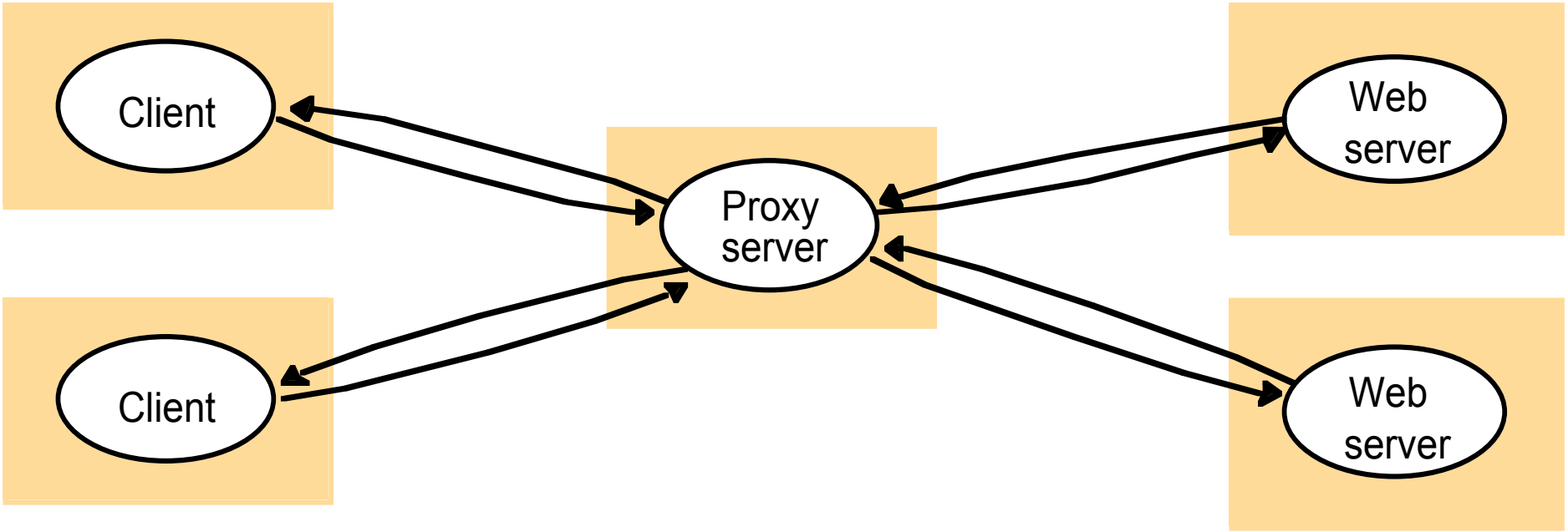
11

Architectural model: multiple-servers



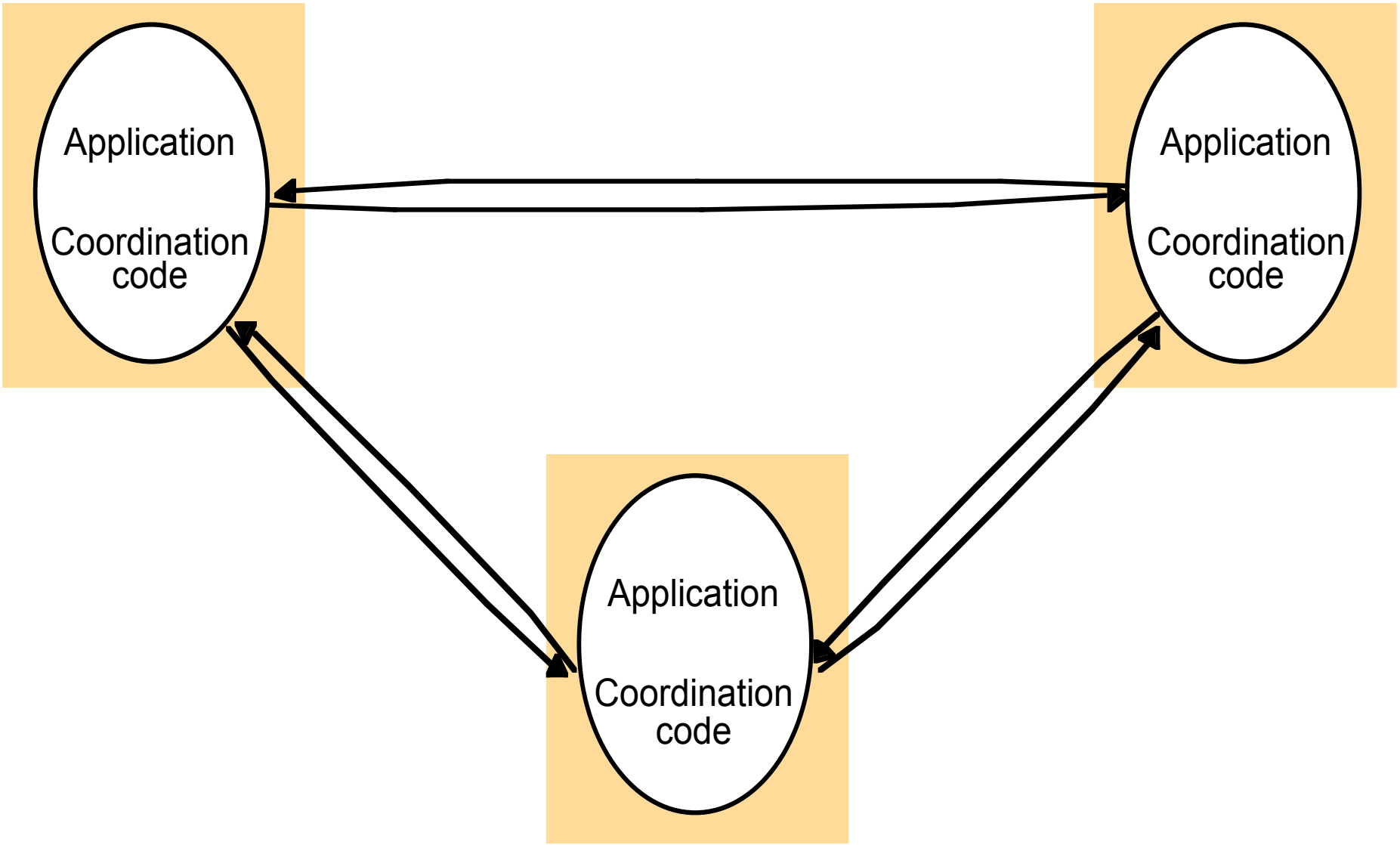
12

An example: Web proxy server



13

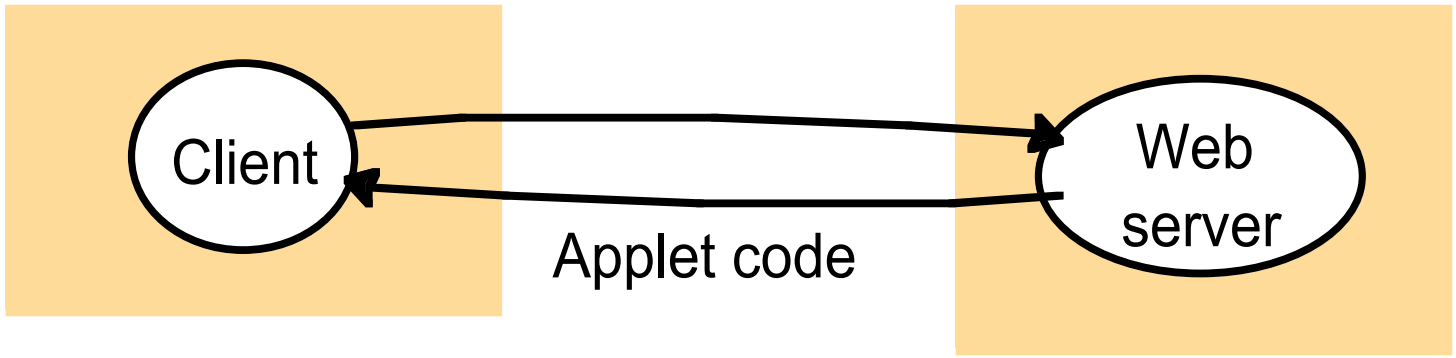
Architectural model: peer-to-peer processes



14

Mobile code: Web applets

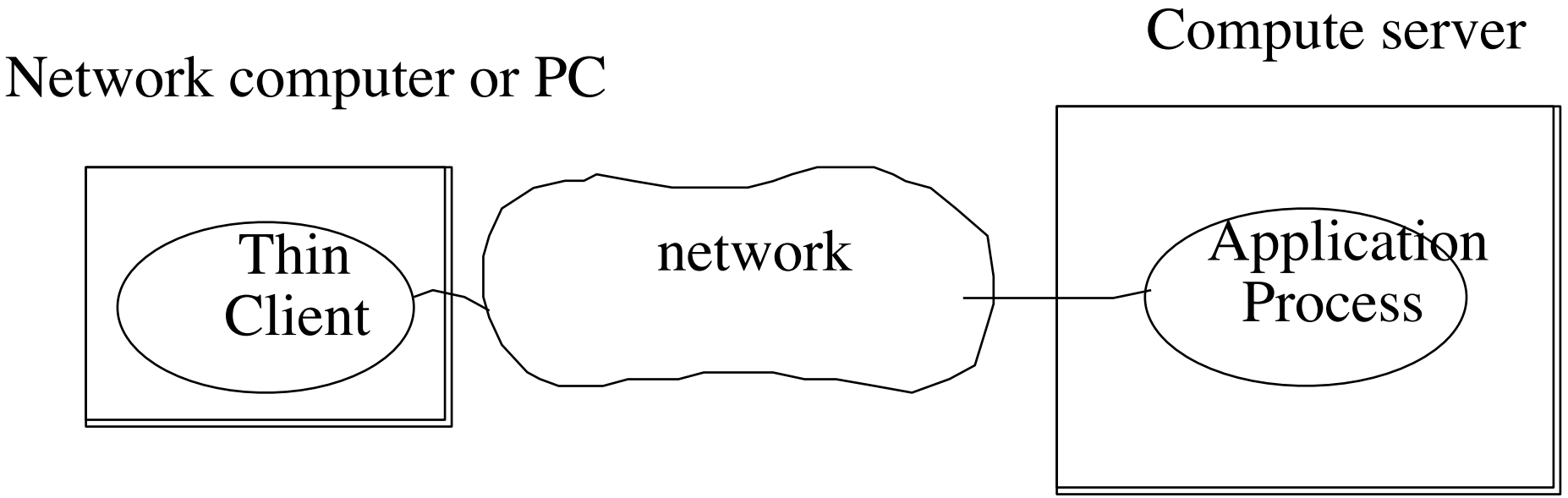
a) client request results in the downloading of applet code



b) client interacts with the applet

15

Thin clients and compute servers



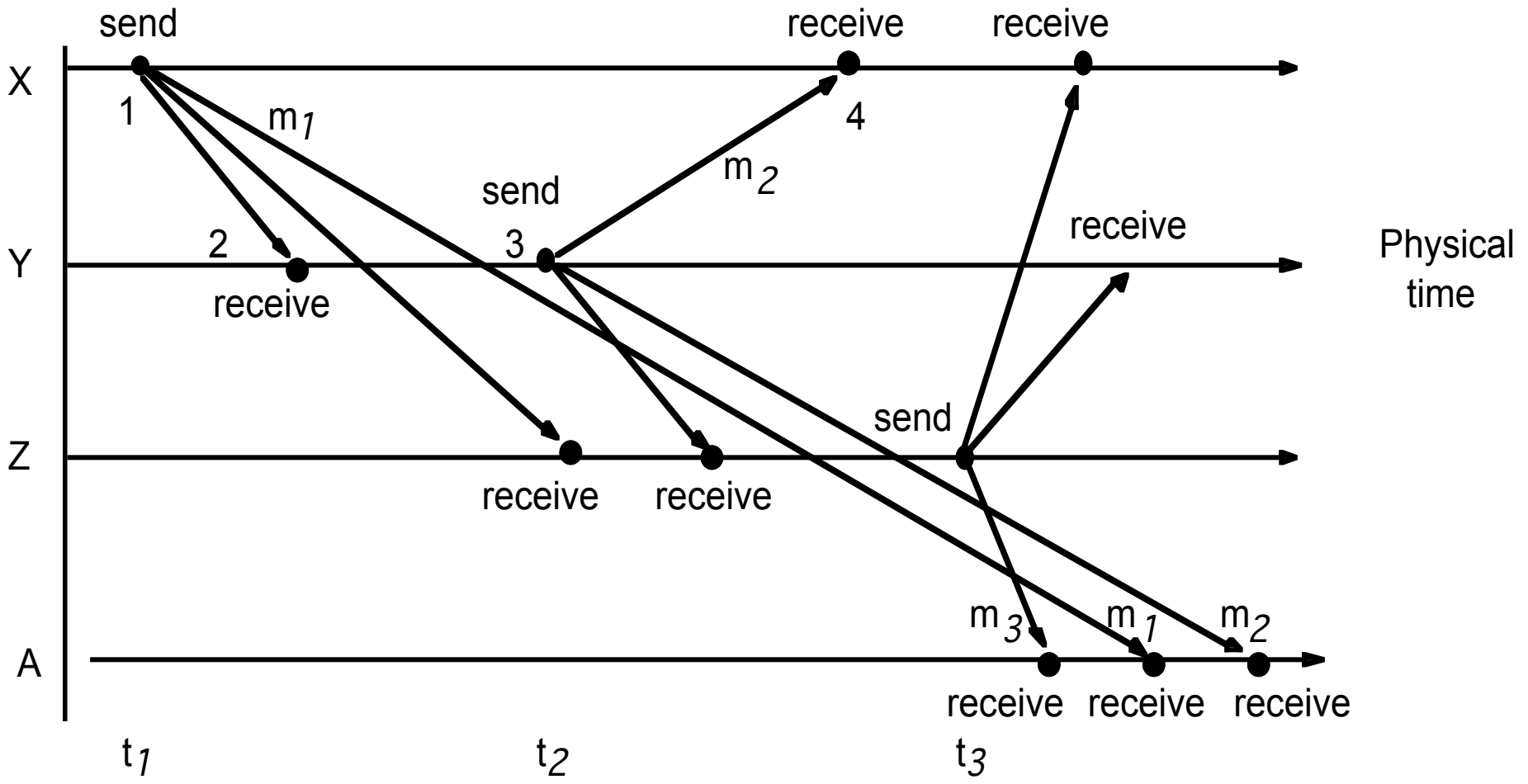
16

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

17

Example: asynchronous email



18

Interaction model

- Performance
  - Latency: Δ time transmission begins and beginning of receipt
  - Bandwith: 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

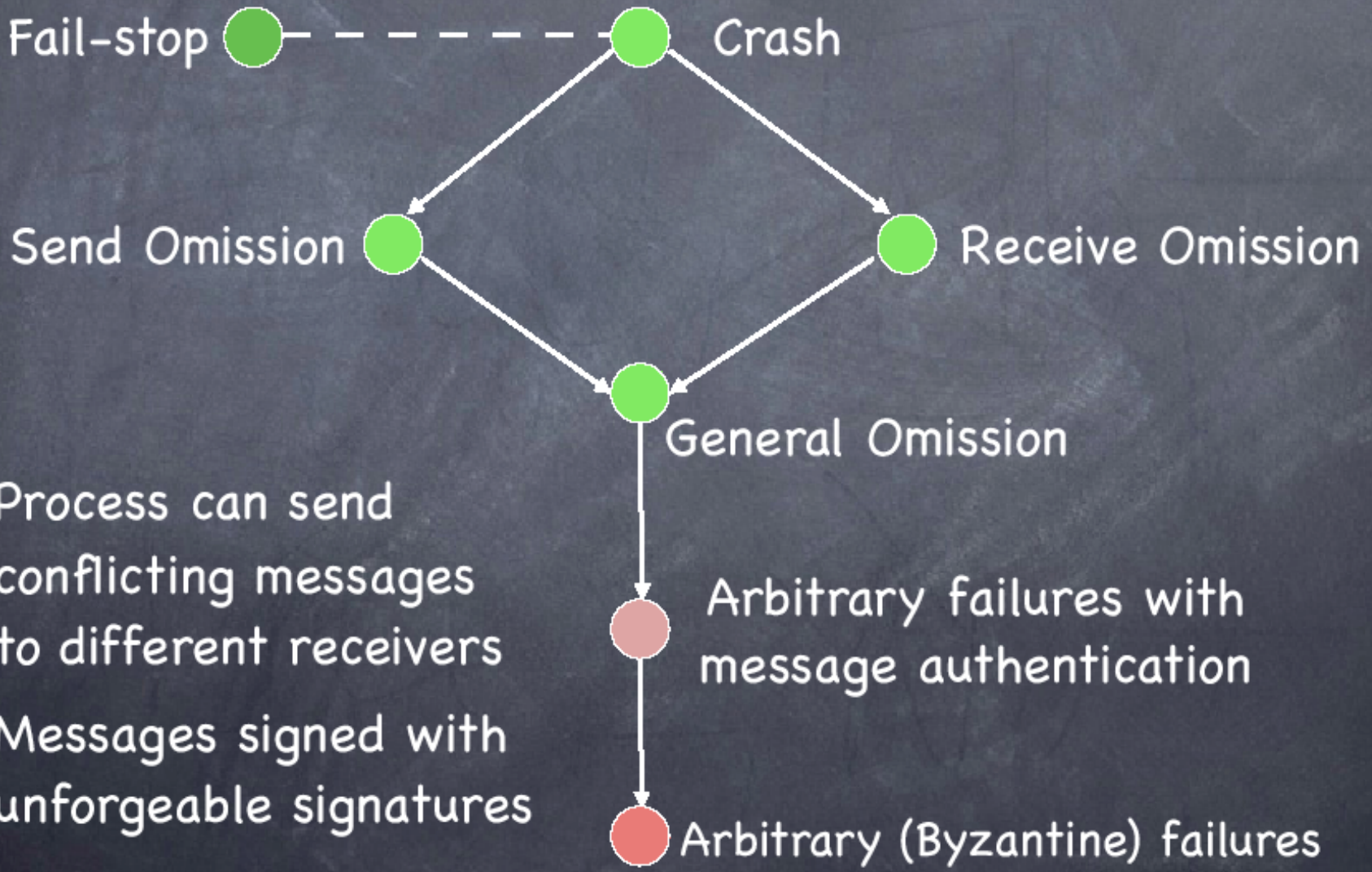
19

Failure model

- Defines ways in which a failure may occur in order to provide an understanding of the effects of failures.

20

Arbitrary failures with message authentication



- Process can send conflicting messages to different receivers
- Messages signed with unforgeable signatures

Security model

- Protecting objects
- Securing processes and their interactions
- Model the enemy's attacks

- Solutions:
  - Cryptography
  - Authentication
  - Secure channels

22