3rd Edition Sec 2.3

Fundamental Models

FUNDAMENTAL MODELS

- What are we trying to capture?
  - Components (usually processes or objects)
  - interactions among components
    - process / message graph to represent it
    - complexity arguments to quantify it
      - (message passes are EXPENSIVE)
    - correctness arguments (ESTELLE, Lotus, FORTE)
  - failure modes
  - security

Interaction models

1] client-server

2] peer processes

3] manager - worker (within a server)

4] ??

Interaction models:
based on communication channels

- Latency:
  - propagation delay (approx 0.6 c, c = speed of light)
  - queuing delay for channel
  - queuing & service times in the OS (large)
  - Manning’s rule: a messgae pass takes a msec
- data rate
  - (24 Kb/s on dialup; 32 Tb/s on fibre)

Interaction models:

role of time

- No single clock & net delays unpredictable

- synchronous model: assume
  - process step execution times bounded above & below
  - message passing times bounded above
  - known drift rates of local clocks
- like synchronous computer design

Interaction models:
role of time

- Asynchronous model:
  - denies all of the above;
  - each may be arbitrarily large
  - usually more realistic

Interaction models:
event ordering

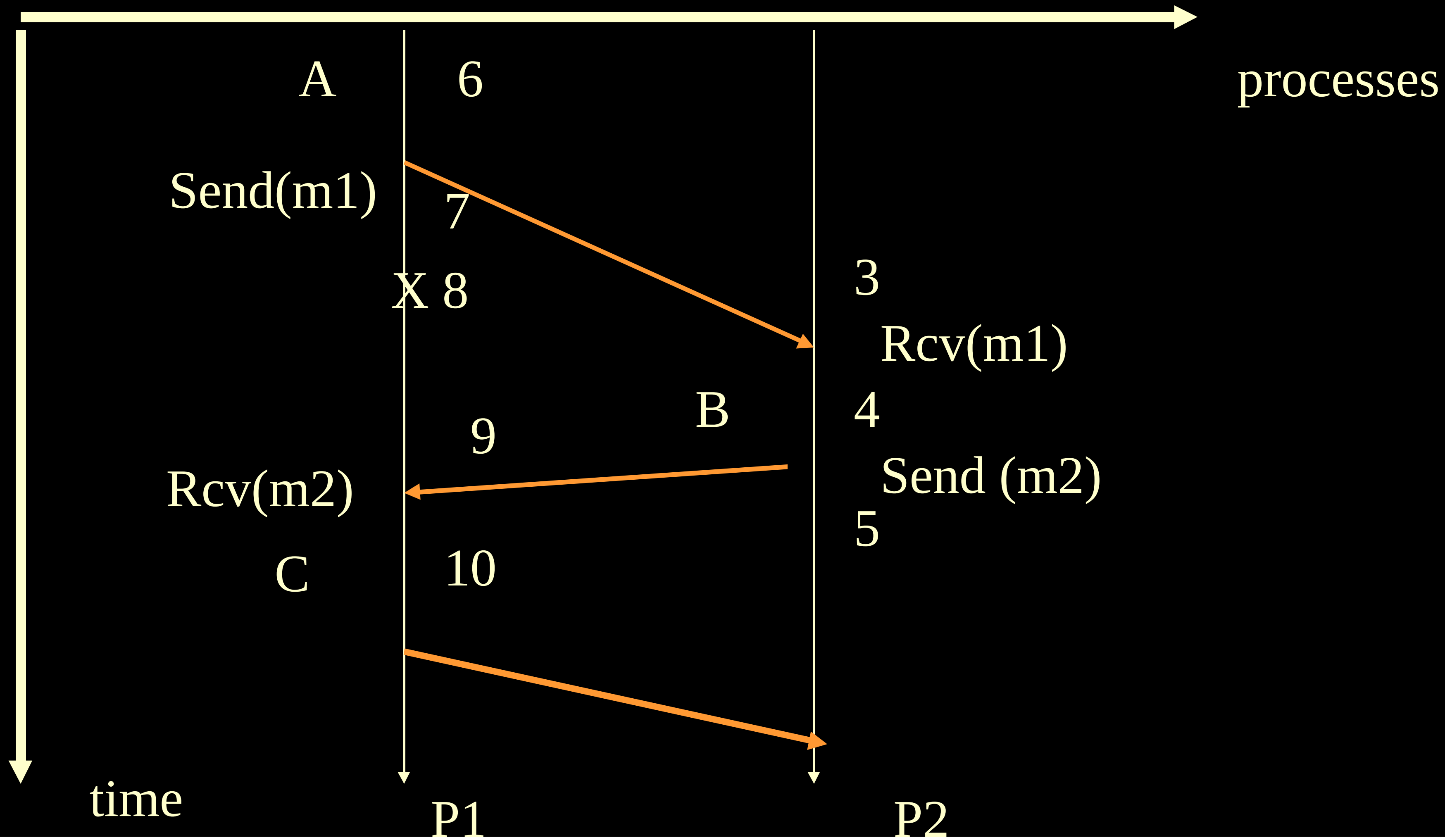
Establishing time ordering among events

  within a process

  among processes

despite the lack of a common time reference

Process time diagram:



event ordering

- Let h be the relation “happened before”

  a h b => a happened before b

  then

    a h b , b h c => a h c (transitive)

  What can we assert about the diagram?

assertions

6 h 7 A hapbef Send(m1), *same process*

7 h 3 send(m1) hapbef (rcv)m1

***above are axioms ***

6 h 4 deduce A hapbef B

8 ?? 3

8 ?? 4

Failure Model

Sec 2.3.2

Faults:

- The fundamental mechanisms that create wrong behaviour
- In hardware: transistors open or short
- In software, who knows?

Failures

- Elementary Wrong behaviour resulting from a fault
- in hardware: 0 -> 1 or 1 -> 0
- in software: see below

malfunction

- Complex behaviour resulting from failure
- in hardware: wrong system state

- in software: wrong system state

Plausible distsys failures

- Omission failure:
  - action a should have happened but didn’t
- process version:
  - crash
- channel version
  - dropped message (sendside, rcveside, or channel)

Plausible distsys failures

- Timing failure (synchronous model)
  - event a should happen before T but happened after
- masked failure
  - what redundancy should create

Properties of a channel with protocols

- Validity:
  - any msg sent is eventually received
- integrity:
  - message sent = message received