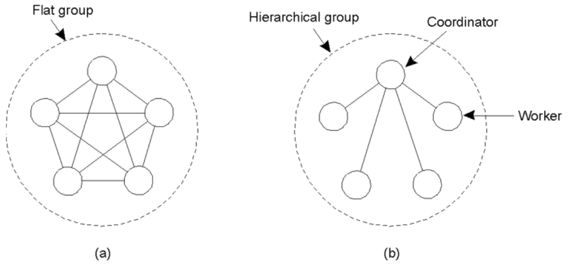
INFO SHEET

Fault Tolerance

- **Basic Concepts:**
  - Availability
  - Reliability
  - Safety
  - Maintainability
- **Failure Models**
  - Crash failure
  - Omission failure
  - Timing failure
  - Response failure
  - Arbitrary failure

Process Resilience

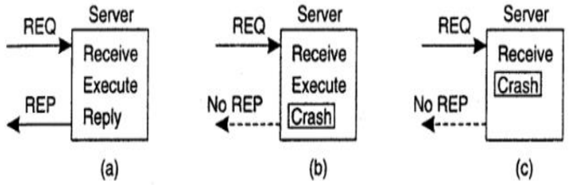


Communication in a flat group.

Communication in a simple hierarchical group

- **Flat groups**
  - symmetrical
  - no single point of failure
  - complicated decision making
- **Hierarchical groups**
  - the opposite properties
- **Group management issues**
  - join, leave
  - crash (no notification)

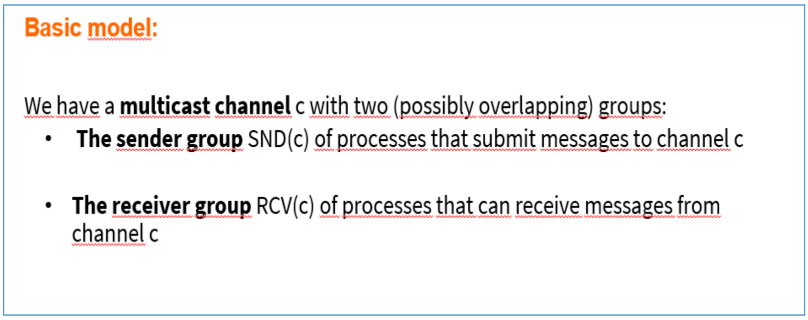
Client/Server communication constancy



→ We need to decide on what we expect from the server:

- At-least-once-semantics

Group communication constancy

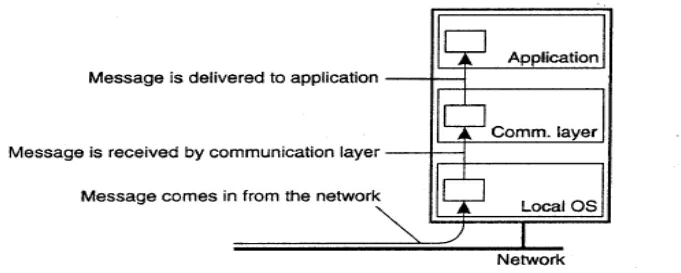


Basic model:

We have a **multicast channel** c with two (possibly overlapping) groups:

- **The sender group** SND(c) of processes that submit messages to channel c
- **The receiver group** RCV(c) of processes that can receive messages from channel c

- **Simple reliable:** if process P ∈ RCV(c) at the time messages m was submitted to c, and P does not leave RCV(c), m should be delivered to P
- **Atomic multicast:** How can we ensure that a message m submitted to channel c is delivered to process P ∈ RCV(c) only if m is delivered to all members of RCV(c)


Message is delivered to application
Message is received by communication layer
Message comes in from the network
Application
Comm. layer
Local OS
Network

Distributed commit

Model:

The client who initiated the computation acts as coordinator, processes required to commit are the participants

- **Phase 1a:** Coordinator sends vote-request to participants (also called a **pre-write**)
- **Phase 1b:** When participant receives vote-request it returns either vote-commit or vote-abort to coordinator. If it sends vote-abort, it aborts ist local computation
- **Phase 2a:** Coordinator collects all votes, if all are vote-commit, it sends global-commit to all participant, otherwise it sends global-abort
- **Phase 2b:** Each participant waits for global-commit or global-abort and handles accordingly.

Recovery

When a failure occurs, we need to bring the system into an error-free state:

- **Forward error recovery**: Find a new state from which the system can continue operation
- **Backward error recovery**: Bring the system back into a previous error-free state

→ Use backward error recovery, requiring that we etablish **recovery points**