Table of Contents

- Introduction and Motivation
- Theoretical Foundations
- Distributed Programming Languages
- Distributed Operating Systems
- Distributed Communication
- Distributed Data Management
- Reliability
- Applications
- Conclusions
- Appendix

Type of Faults

- Types of **faults:**
  - **Hardware** faults
  - **Software** faults
  - **Communication** faults
  - **Timing** faults
- Schneider's classification:
  - Omission failure
  - Failstop failure (detectable)
  - Crash failure (undetectable)
  - Crash and link failure
  - Byzantine failure

Redundancy

- **Hardware redundancy**: extra PE's, I/O's
- **Software redundancy**: extra version of software modules
- **Information redundancy**: error detecting code
- **Time redundancy**: additional time used to perform a function

Fault Handling Methods

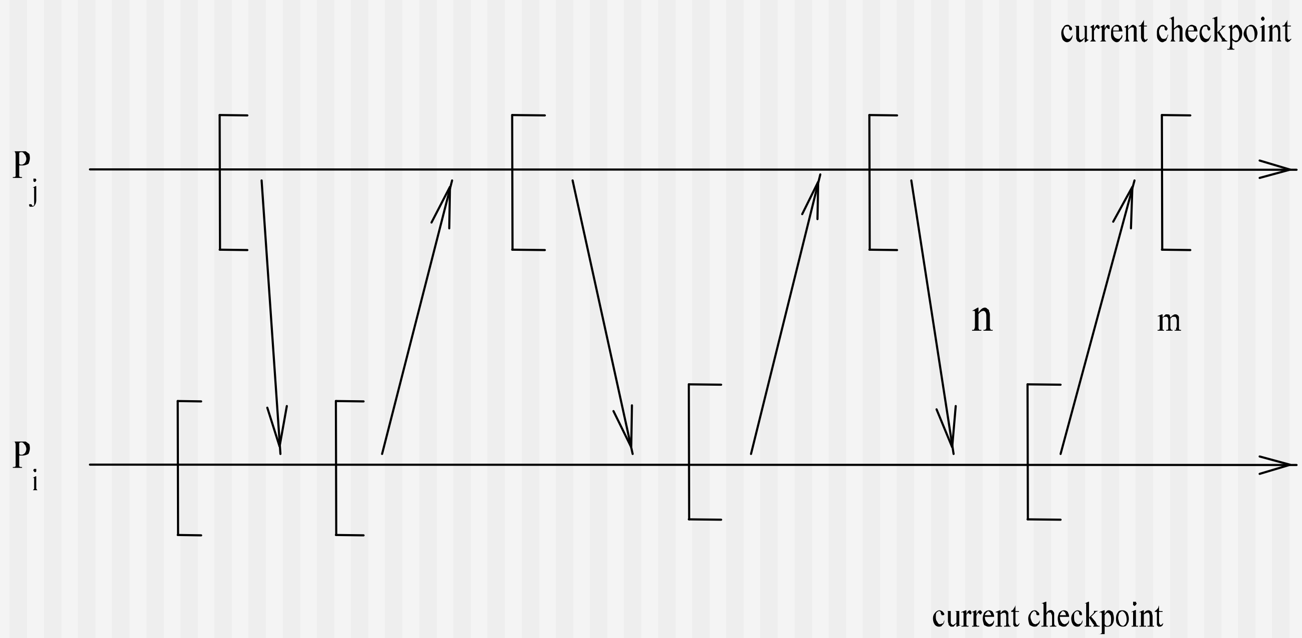
- **Active** replication
- **Passive** replication
- **Semi-active** replication

Building Blocks of Fault-Tolerant Design

- **Stable storage** is a logical abstraction for a special storage that can survive system failure.
- **Fail-stop processors** do not perform any incorrect action and simply cease to function.
- An **atomic action** is a set of operations which are executed indivisibly by hardware. That is, either operations are completed successfully and fully or the state of the system remains unchanged (operations are not executed at all).

Domino Effect

- Storage of checkpoints
- Checkpointing methods: coordinated set up



An example of domino effect.

Focus 21: Byzantine Faults

- Several divisions of the Byzantine army camp outside an enemy city. Each division commanded by its own general. Generals from different divisions communicate only through messengers. Some of the generals may be traitors. After observing the enemy, the generals must decide upon a common battle plan.

Two Requirements

- All loyal generals decide upon *the same plan* of action
- A small number of traitors *cannot* cause the loyal generals to *adopt a wrong plan*

Note

- Loyal generals may start with different decisions, but end up the same decision.
- The final decision must come from at least one loyal general's initial decision.

Focus 21 (Cont'd)

- Theoretical result
  - Consensus is reachable if $n \geq 3m + 1$, where n is the total number of generals and m is the number of traitors.
- (m+1)-round of consensus algorithm
  - At the first round, each node, including traitors, broadcasts its initial decision
  - At the (i+1)th round, each node broadcasts all messages received at ith round

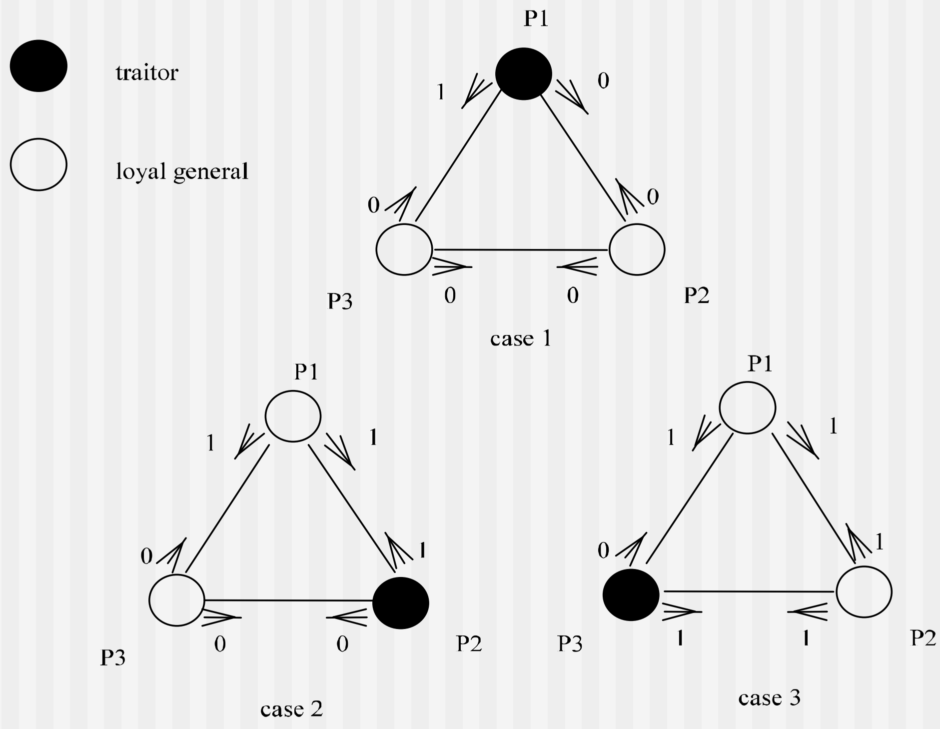
Agreement Protocol

(superscript: version number, d: don't care)

P1* (traitor)	P2	P3	P4
First round: $(-, v_2, v_3, v_4)$	$(v_1^2, -, v_3, v_4)$	$(v_1^1, v_2, -, v_4)$	$(v_1^3, v_2, v_3, -)$
Second round: $(v_1^2, -, v_3, v_4)$ $(v_1^1, v_2, -, v_4)$ $(v_1^3, v_2, v_3, -)$	$(v_1^1, v_2, -, v_4)$ $(v_1^3, v_2, v_3, -)$ $(-, v_2^4, v_3^4, v_4^4)$	$(v_1^2, -, v_3, v_4)$ $(v_1^3, v_2, v_3, -)$ $(-, v_2^5, v_3^5, v_4^5)$	$(v_1^2, -, v_3, v_4)$ $(v_1^1, v_2, -, v_4)$ $(-, v_2^6, v_3^6, v_4^6)$
(d_1, d_2, d_3, d_4)	(v_1^7, v_2, v_3, v_4)	(v_1^7, v_2, v_3, v_4)	(v_1^7, v_2, v_3, v_4)

An algorithm for reaching agreement.

No-Agreement Among Three Processes



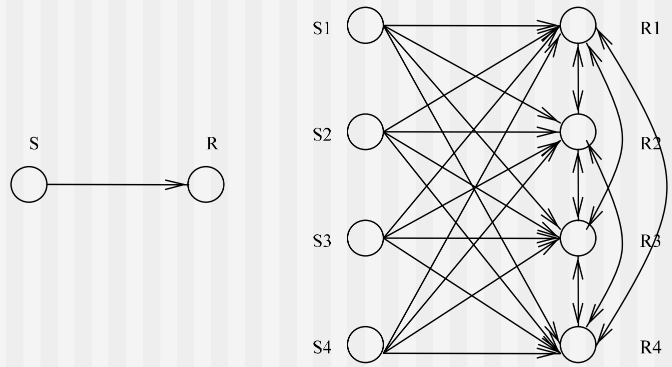
Cases leading to failure of the Byzantine agreement.

Extended Agreement Protocols

- Boolean values or arbitrary real values for the decisions.
- *Unauthenticated or authenticated messages*.
- Synchronous or asynchronous.
- *Completely connected network or partially connected networks*.
- Deterministic or randomized.
- Byzantine faults or fail-stop faults.
- Non-totally decentralized control system and, in particular, hierarchical control systems.

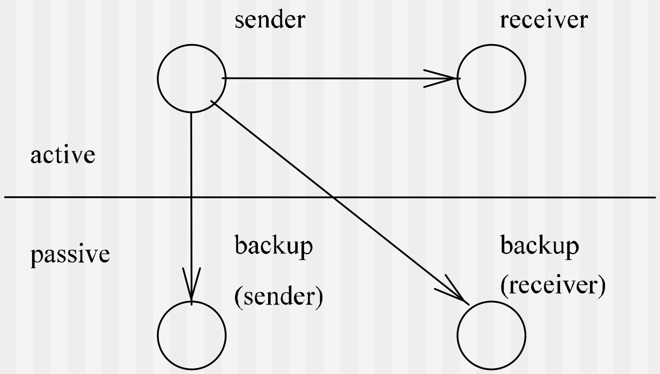
Reliable Process

Active model



(a) (b)

Passive model



Reliable Communication

- Acknowledgement with time out:

  Timeout mechanism with an acknowledge for the receipt of each packet.
- TCP:

  Transport protocol for reliable point-to-point comm
- Efficiency:

  TCP allows multiple data packets to be transmitted before the ack of the first packet is received

  TCP can also estimate the round-trip time and limit data rates to clear out congestions

Reliable Group Communication

- **Feedback suppression**: multicast or broadcast each positive (or negative) acknowledge.
- Negative acknowledgement
  - Signal for a missing packet.
  - Pros: better scalability (without positive acknowledgement).
  - Cons: sender is forced to keep each packet in the buffer forever.
- Combination of positive and negative acknowledgements

Example 26: Positive and Negative Acknowledgements in Multiple Broadcasting

Let A be a packet and a ($\underline{a}$) the positive (negative) acknowledgement for A.

E.g., A,Ba,Cb,Db,Ec,F $\underline{c}$d,Cb,G$\underline{d}$ef

1.Message A is sent first, acknowledged by the sender of B, which is in turn acknowledged by the senders of C and D.

2. The sender of E acknowledges C and the sender of F acknowledges the receipt of D but a negative acknowledgment of C .

3. Some node (not necessarily the original sender) retransmits C .

4. The sender of G acknowledges both E and F but sends a negative acknowledgment of D (after receiving F).

Multicasting Basics

- IP multicast
  - Source trees (shortest path trees)
  - Shared trees (also core trees)
- Reserve path forwarding
  - It is tree-based with upstream (toward the source) and downstream (away from the source) links
- Application layer multicast
  - Overlay networks, such as P2P

Different Types of Multicasting

- Atomic multicast: all-or-thing semantics (basic requirement)
- Reliable multicast: reliable, but no message ordering
- Ordered multicast
  - FIFO multicast: FIFO-ordered delivery (also local order)
  - Causal multicast: causal-ordered delivery
  - Total order multicast: global total order

Reliable Multicasting

- Each receiver multicasts the received message to others
- Node crash (suppose sequential send) or link failure

Atomic condition is also satisfied:

- If the sender crashes before sending a single message, it is ok.
- If the sender crashes at the middle, the one received the message will re-multicast the message to others.

The cost can be reduced if the failure is bounded

- Can reduce the number of re-multicast

Reliable Multicasting

- Each receiver multicasts the received message to others
- Node crash (suppose sequential send) or link failure

Atomic condition is also satisfied:

- If the sender crashes before sending a single message, it is ok.
- If the sender crashes at the middle, the one received the message will re-multicast the message to others.

The cost can be reduced if the failure is bounded

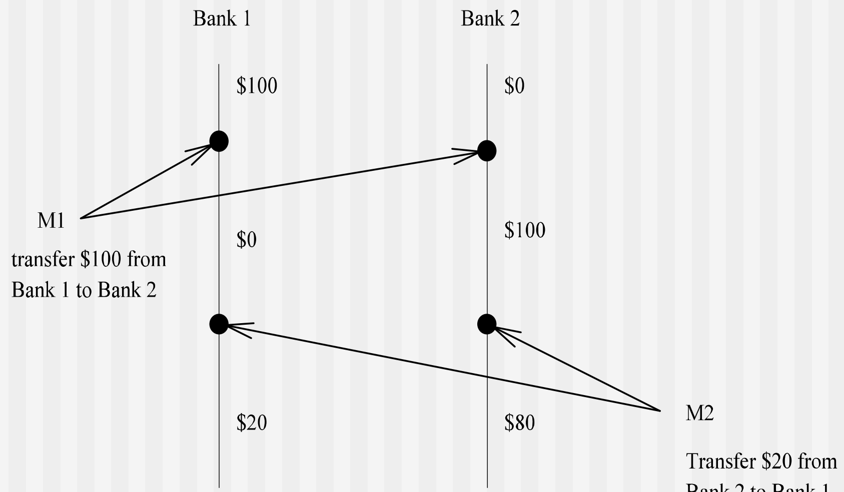
- Can reduce the number of re-multicast

Causal Order Multicast

- Can use vector time stamps
- Similar to mutual exclusion protocol

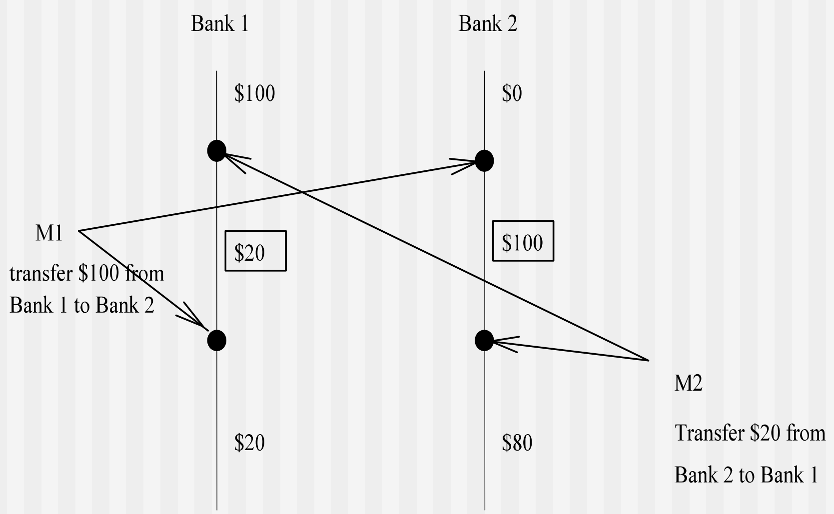
Multicast with Total Order

Multicast with total order



$\mathrm{seq}(m_1) < \mathrm{seq}(m_2)$

Multicast without total order



Neither $\mathrm{seq}(m_1) < \mathrm{seq}(m_2)$ nor $\mathrm{seq}(m_2) < \mathrm{seq}(m_1)$

Focus 22: Total-Ordered Multicasting

- **Total-ordered multicasting**
  - Each transfer order (message) can be assigned a global sequence number.
  - There exists a global sequence.
- **Sequencer**
  - The sender sends message to a sequencer
  - The sequencer allocates a global sequence number to the message.
  - The message is delivered by every destination based on the order.

Implementations of Sequencer

- Privilege-based (token circulated among the senders)
- Fixed sequencer (a fixed third party)
- Moving sequencer(token circulated among the third-party nodes)

Reliable Order Multicast

- In an asynchronous distributed system, total order multicasts cannot be implemented when even a single process crashes.

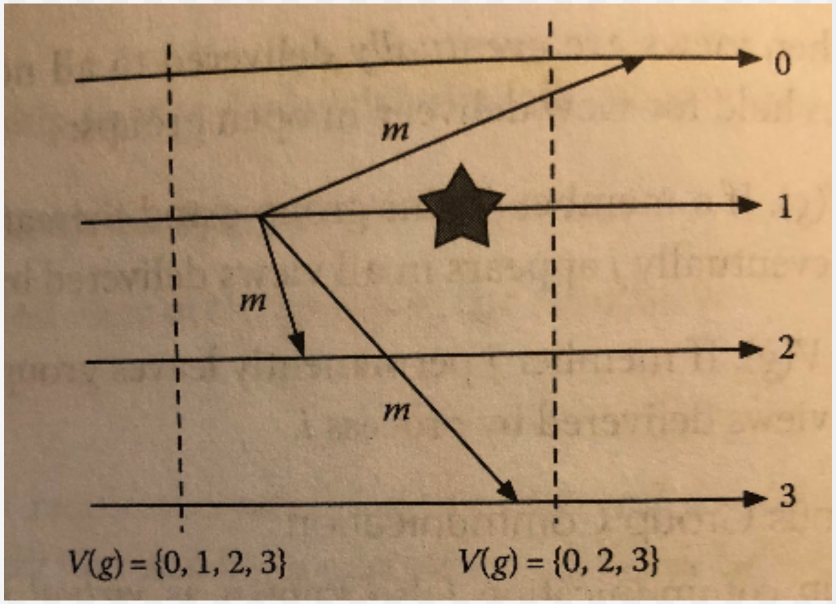
Similar to the asynchronous consensus problem

Open Groups and Issues

- Open groups: multicast members are allowed to spontaneously join and leave

View changes: {0, 1, 2, 3} to {1, 2, 3}

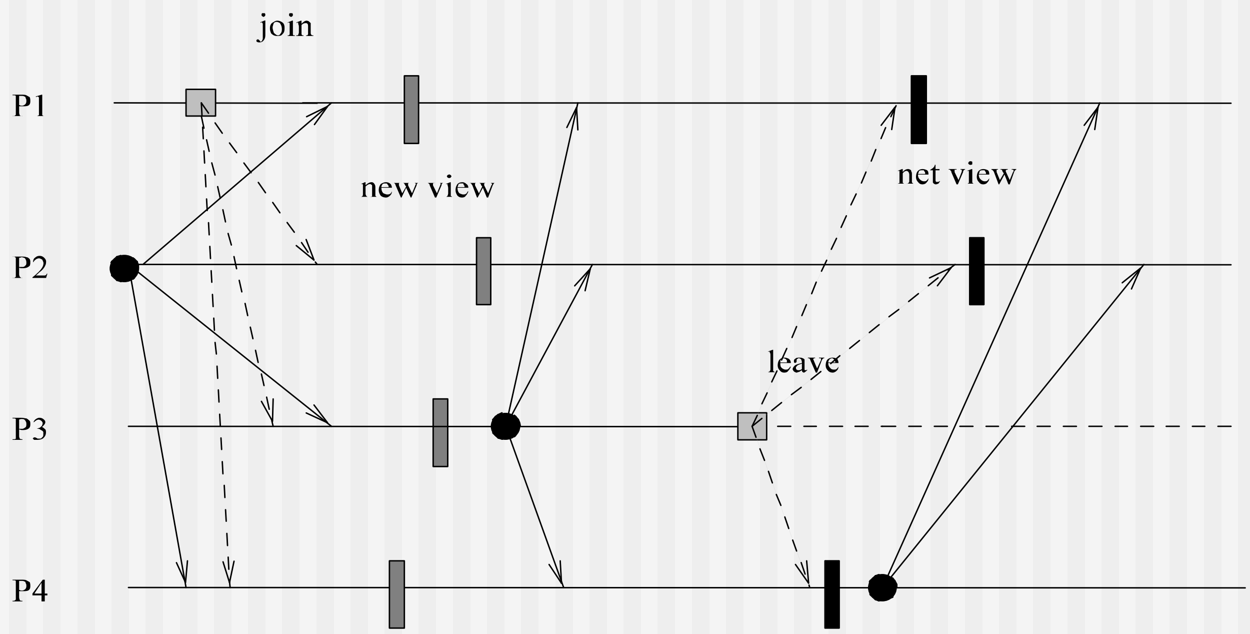
Inconsistency: 2 and 3 delivery m, not 1 in new view



Focus 23: Birman's Virtual Synchrony

- Virtual synchrony: reliable multicast with a special property.
- View: a multicast group.
- View change: (a) a new process joins, (b) a process leaves, and (c) a process crashes.
- Each view change is multicast to members in the group.
- Special property: each view change acts as a barrier across which no multicast can pass. (Application: distributed debugging.)

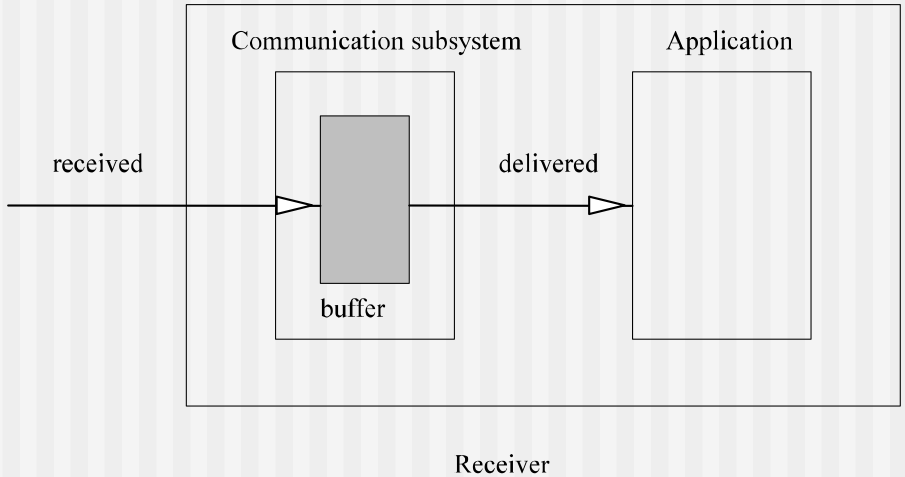
Focus 23 (Cont'd)



Virtual synchrony.

Implementing a Virtual Synchronous Reliable Multicast

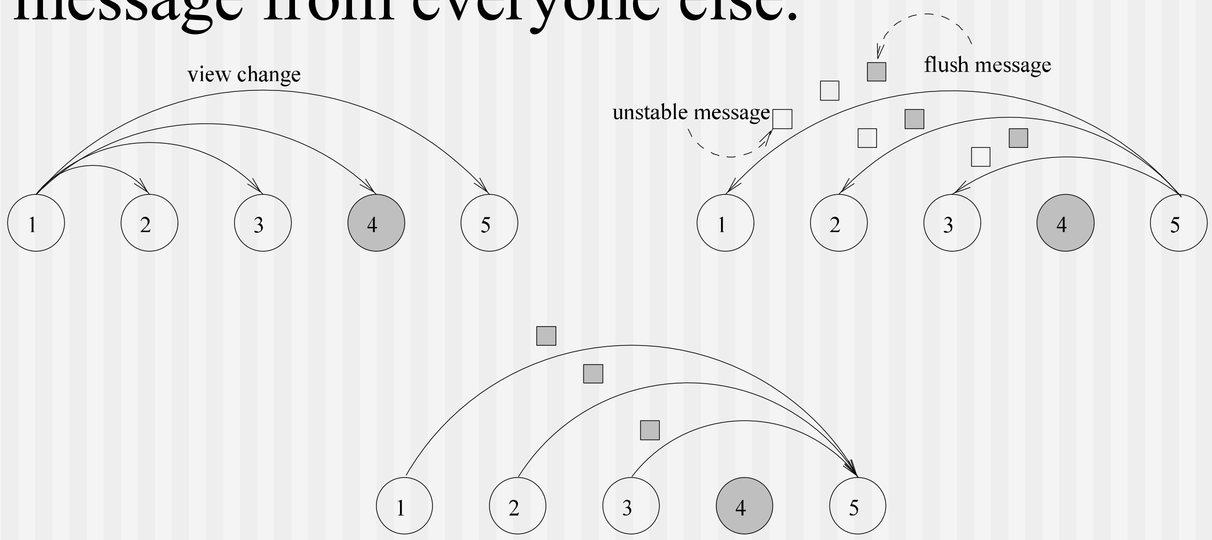
- Message received versus message delivered.
- If message *m* has been delivered to all members in the group, *m* is called **stable**.
- Point-to-point communication is reliable (TCP).
- Sender may crash before completing the multicasting. (Some members received the message but others did not.)



Message receipt versus message delivery.

Implementing a Virtual Synchronous Reliable Multicast (Cont'd)

- At group view G_i, a view changed is multicast.
- When a process receives the view-change message for G_{i+1}, it multicasts to G_{i+1} a copy of unstable messages for G_i followed by a **flush message**.
- A process installs the new view G_{i+1} when it has received a flush message from everyone else.



Virtual synchrony.

Exercise 7

1. Explain the difference between FIFO order, casual order, and total order multicast with examples.

2. Show how to implement a causal order multicast using vector time stamps.

3. In Byzantine agreement protocol $k + 1$ rounds of message exchanges are needed to tolerant k faults. The number of processes n is at least $3k + 1$. Assume P_1 and P_2 are faulty in a system of $n = 7$ processes.

- (a) Show the messages P_3 receives in first, second, and third round.
- (b) Demonstrate the correctness of the protocol by showing the final result vector (after a majority voting) for P_3.
- (c) Briefly show that result vectors for other non-faulty processes are the same.