Group Communication

- Point-to-point vs. one-to-many
- Multicast communication
- Atomic multicast
- Virtual synchrony
- Group management
- ISIS

Reading:
- Coulouris: Distributed Systems, Addison Wesley, Chapter 4.4, Chapter 11

Group Communication: Introduction

- One-to-many communication
- Dynamic membership
- Groups can have various communication patterns
  - peer group
  - server group
  - client-server group
  - subscription (diffusion) group
  - hierarchical groups
Group Membership Management

Multiicast Communication

- Reliability guarantees:
  - *Unreliable* multicast: Attempt is made to transmit the message to all members without acknowledgement.
  - *(Reliable* multicast: Message may be delivered to some but not all group members.)
  - *Atomic* multicast: All members of the group receive message, or none of them do.

- Message *reception*: message has been received and buffered in the receiver machine. Not yet delivered to the application.
- Message *delivery*: The previously received message is delivered to the application.
Multicast Communication: Message Ordering

- *Globally (chronologically) ordered* multicast: All members are delivered messages in order they were sent.
- *Totally (consistently) ordered* multicast: Either \( m_1 \) is delivered before \( m_2 \) to all members, or \( m_2 \) is delivered before \( m_1 \) to all members.
- *Causally ordered* multicast: If the multicast of \( m_1 \) happened before the multicast of \( m_2 \), then \( m_1 \) is delivered before \( m_2 \) to all members.
- *Sync-ordered* multicast: If \( m_1 \) is sent with sync-ordered multicast primitive, and \( m_2 \) is sent with *any ordered* multicast primitive, then either \( m_1 \) is delivered before \( m_2 \) at all members, or \( m_2 \) is delivered before \( m_1 \) at all members.
- *Unordered* multicast: no particular order is required on how messages are delivered.

Message Ordering: Examples

[Diagram showing message ordering examples]
Atomic Multicast

- Simple multicast algorithm: Send a message to every process in the multicast group, using reliable message passing mechanism (e.g. TCP).
  - Is not atomic: does not handle processor failures.

- “Fix” to simple multicast algorithm: Use **2-phase-commit** (2PC) technique and treat multicast as transaction.
  - Works, but correctness guarantees stronger than necessary
  - 1. If sending process $s$ fails to obtain ack from process $p$, $s$ must abort delivery of message.
  - 2. If $s$ fails after delivering $m$ to all processors, but before sending “commit” message, delivery of $m$ is blocked until $s$ recovers.

- 2PC protocol does more work than is really necessary.

---

2-Phase-Commit Protocol

- Protocol for atomic commit.

![2-Phase-Commit Protocol Diagram](image-url)