



African University of Science and Technology

Group Communication

Prof. Ousmane THIARE

[www.ousmanethiare.com]

September 12, 2025

Outline

Modes of communication

Implementing Group Communication Mechanisms

Reliability of multicasts

Message ordering

IP multicasting routing

Modes of communication

■ unicast

- $1 \leftrightarrow 1$
- Point-to-point

■ anycast

- $1 \rightarrow$ nearest 1 of several identical nodes
- Introduced with IPv6; used with BGP

■ netcast

- $1 \rightarrow$ many, 1 at a time

■ multicast

- $1 \rightarrow$ many
- group communication

■ broadcast

- $1 \rightarrow$ all

Groups

- Groups are dynamic
 - Created and destroyed
 - Processes can join or leave
 - May belong to 0 or more groups
- Send message to one entity
 - Deliver to entire group

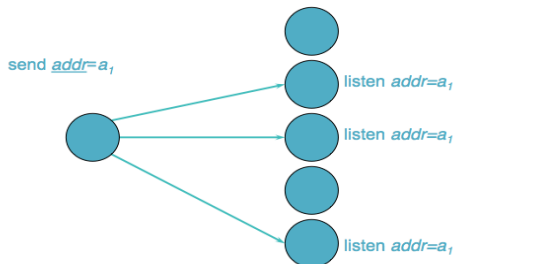
Deal with collection of processes as one abstraction

Design Issues

- Closed vs. Open
 - Closed: only group members can sent messages
- Peer vs. Hierarchical
 - Peer: each member communicates with group
 - Hierarchical: go through dedicated coordinator(s)
 - Diffusion: send to other servers & clients
- Managing membership & group creation/deletion
 - Distributed vs. centralized
- Leaving & joining must be synchronous
- Fault tolerance
 - Reliable message delivery? What about missing members?

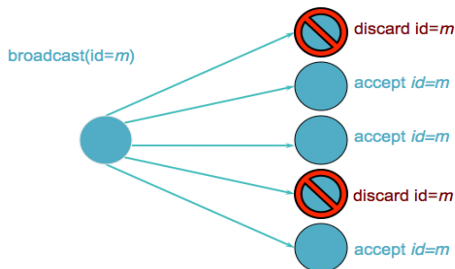
Hardware multicast

- Hardware support for multicast
 - Group members listen on network address



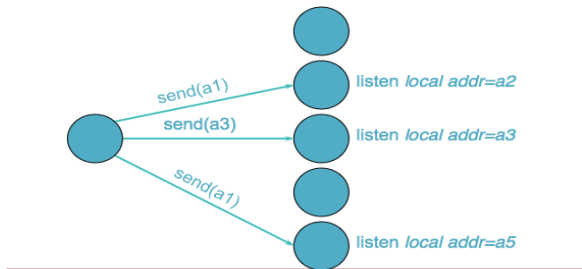
Hardware broadcast

- Hardware support for broadcast
 - Software filters multicast address • May be auxiliary address



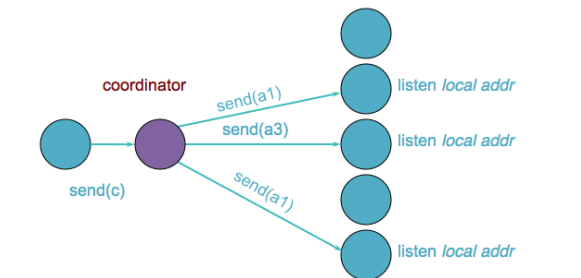
Software: netcast

- Multiple unicasts (netcast)
 - Sender knows group members



Software: hierarchical

- Multiple unicasts via group coordinator
 - coordinator knows group members



Atomic multicast

■ Atomicity

- Message sent to a group arrives at all group members
 - If it fails to arrive at any member, no member will process it.

■ Problems

Unreliable network

- Each message should be acknowledged
- Acknowledgements can be lost

Message sender might die

Achieving atomicity

Retry through network failures & system downtime

- **Send message to all group members**

- ☐ Each receiver acknowledges message
- ☐ Saves message and acknowledgement in log
- ☐ Does not pass message to application

- **Sender waits for all acknowledgements**

- ☐ Retransmits message to non-responding members
 - Again and again... until responses from all are received

- **Sender sends "go" message to all members**

- ☐ Each recipient delivers message to application
- ☐ Sends reply to server

Achieving atomicity

All members will eventually get the message

- Phase 1:

- Make sure that everyone gets the message

- Phase 2:

- Once everyone has confirmed receipt, let the application see it

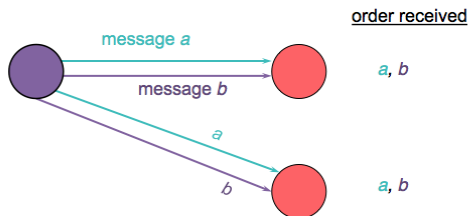
Reliable multicast

- All nonfaulty group members will receive the message
 - Assume sender & recipients will remain alive
 - Network may have glitches
 - Retransmit undelivered messages
- Acknowledgements
 - Send message to each group member
 - Wait for acknowledgement from each group member
 - Retransmit to non-responding members
 - Subject to feedback implosion
- Negative acknowledgements
 - Use a sequence # on each message
 - Receiver requests retransmission of a missed message
 - More efficient but requires sender to buffer messages indefinitely

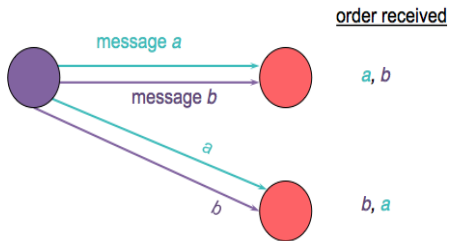
Unreliable multicast

- Basic multicast
- Hope it gets there

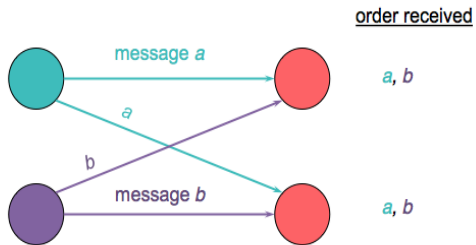
Good ordering



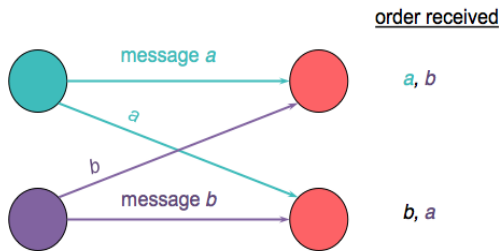
Bad ordering



Good ordering



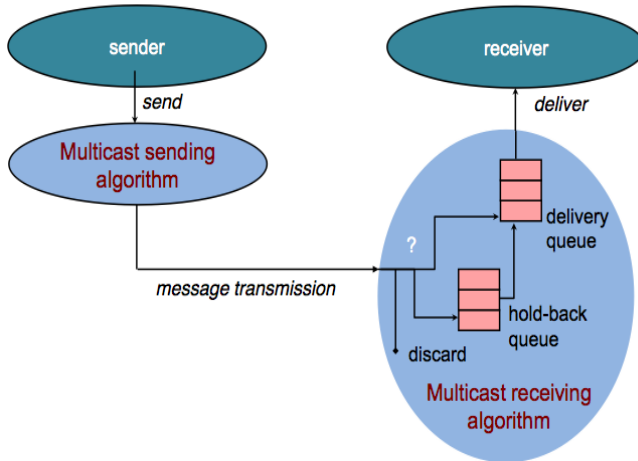
Bad ordering



Sending versus Delivering

- Multicast receiver algorithm decides when to **deliver** a message to the process.
- A received message may be:
 - **Delivered immediately**
 - Network may have glitches
(put on a delivery queue that the process reads)
 - **Placed on a hold-back queue**
(because we need to wait for an earlier message)
 - **Rejected/discarded**
(duplicate or earlier message that we no longer want)

Sending, delivering, holding back



Global time ordering

- All messages arrive in exact order sent
- Assumes two events never happen at the exact same time!
- Difficult (impossible) to achieve

Total ordering

- Consistent ordering everywhere
- All messages arrive at all group members in the same order
 - If a process sends m before m' then any other process that delivers m' will have delivered m .
 - If a process delivers m' before m'' then every other process will have delivered m' before m'' .
- Implementation:
 - Attach unique totally sequenced message ID
 - Receiver delivers a message to the application only if it has received all messages with a smaller ID

Total ordering

- Partial ordering
 - Messages sequenced by Lamport or Vector timestamps
 - If $\text{multicast}(G, m) \rightarrow \text{multicast}(G, m')$
then every process that delivers m' will have delivered m
- Implementation:
 - Deliver messages in timestamp order per-source.

Sync ordering

- Messages can arrive in any order
- Special message type
 - Synchronization primitive
 - Ensure all pending messages are delivered before any additional (post-sync) messages are accepted

FIFO ordering

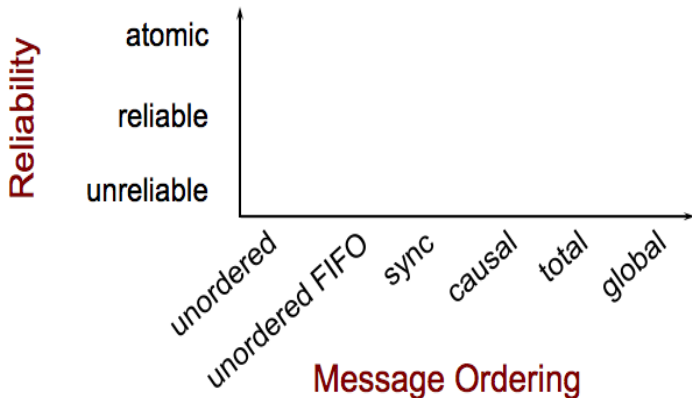
- Messages can be delivered in different order to different members
- Message m must be delivered before message m' iff m was sent before m' from the same host

If a process issues a multicast of m followed by m' , then every process that delivers m' will have already delivered m .

Unordered multicast

- Messages can be delivered in different order to different members
- Order per-source does not matter.

Multicasting considerations



IP multicasting routing

- Deliver messages to a subset of nodes
- How do we identify the recipients?
 - Enumerate them in the header?
 - What if we don't know?
 - What if we have thousands of recipients?
- Use a special address to identify a group of receivers
 - A copy of the packet is delivered to all receivers associated with that group
 - Host group=set of machines listening to a particular multicast address

IP multicasting

- Can span multiple physical networks
- Dynamic membership
 - Machine can join or leave at any time
- No restriction on number of hosts in a group
- Machine does not need to be a member to send messages
- Efficient: Packets are replicated only when necessary

IGMP

- Internet Group Management Protocol (IGMP)
 - Operates between a host and its attached router
 - Goal: allow a router to determine to which of its networks to forward IP multicast traffic
 - IP protocol
- Three message types
 - Membership_query
 - Sent by a router to all hosts on an interface to determine the set of all multicast groups that have been joined by the hosts on that interface
 - Membership_report
 - Host response to a query or an initial join or a group
 - Leave_group
 - Host indicates that it is no longer interested
 - Optional:router infers this if the host does not respond to a query

Multicast Forwarding

- IGMP allows a host to subscribe to a multicast stream
- What about the source?
 - There is no protocol for the source!
 - It just sends to a class D address
 - Routers have to do the work

Multicast Forwarding

- IGMP: Internet Group Management Protocol
 - Designed for routers to talk with hosts on directly connected networks
- PIM: Protocol Independent Multicast
 - Multicast Routing Protocol for delivering packets across routers
 - Topology discovery is handled by other protocols

Flooding: Dense Mode Multicast

- Relay multicast packet to all connected routers
 - Use a spanning tree and use reverse path forwarding (RPF) to avoid loops
 - Feedback & cut-off if there are no interested receivers on a link
 - A router sends a prune message.
 - Periodically, routers send messages to refresh the prune state
 - Flooding is initiated by the sender's router
- Reverse path forwarding (RPF): avoid routing loops
 - Packet is duplicated & forwarded ONLY IF it was received via the link that is the shortest path to the sender
 - Shortest path is found by checking the forwarding table to the source address

Flooding: Dense Mode Multicast

- Advantage:

- ☐ Simple
- ☐ Good if the packet is desired in most locations

- Disadvantage:

- ☐ wasteful on the network, wasteful extra state & packet duplication on routers

Sparse Mode Multicast

- Initiated by the routers at each receiver
- Each router needs to ask for a multicast feed with a PIM **Join** message
 - Initiated by a router at the destination that gets an IGMP join
 - Spanning tree constructed
 - **Join** messages propagate to a defined **rendezvous** point
 - Sender transmits only to the rendezvous point
 - A **Prune** message stops a feed
- Advantage:
 - Packets go only where needed
 - Creates extra state in routers only where needed