

# Distributed Systems

## 7. Group Communication

Paul Krzyzanowski  
pxk@cs.rutgers.edu

# 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?

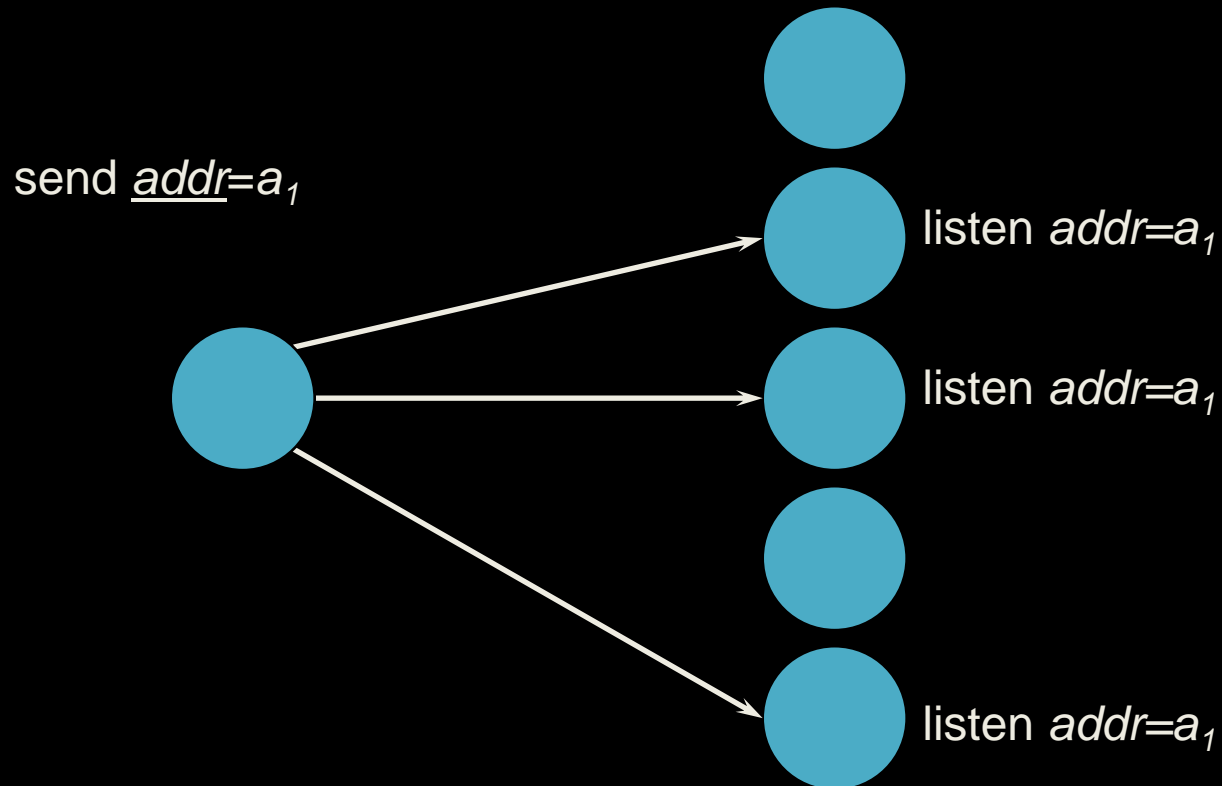
# Implementing Group Communication Mechanisms

# Hardware multicast

---

## Hardware support for multicast

- Group members listen on network address

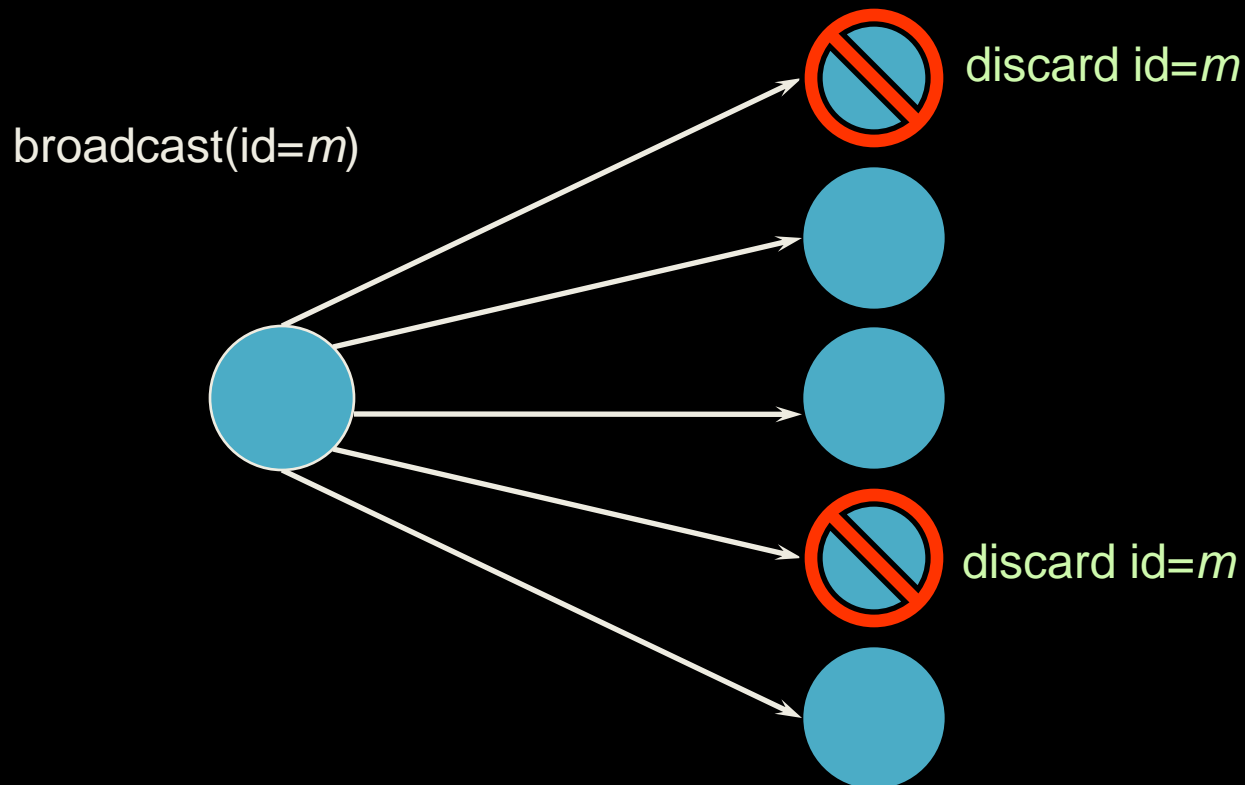


# Hardware broadcast

---

## Hardware support for broadcast

- Software filters multicast address
  - May be auxiliary address

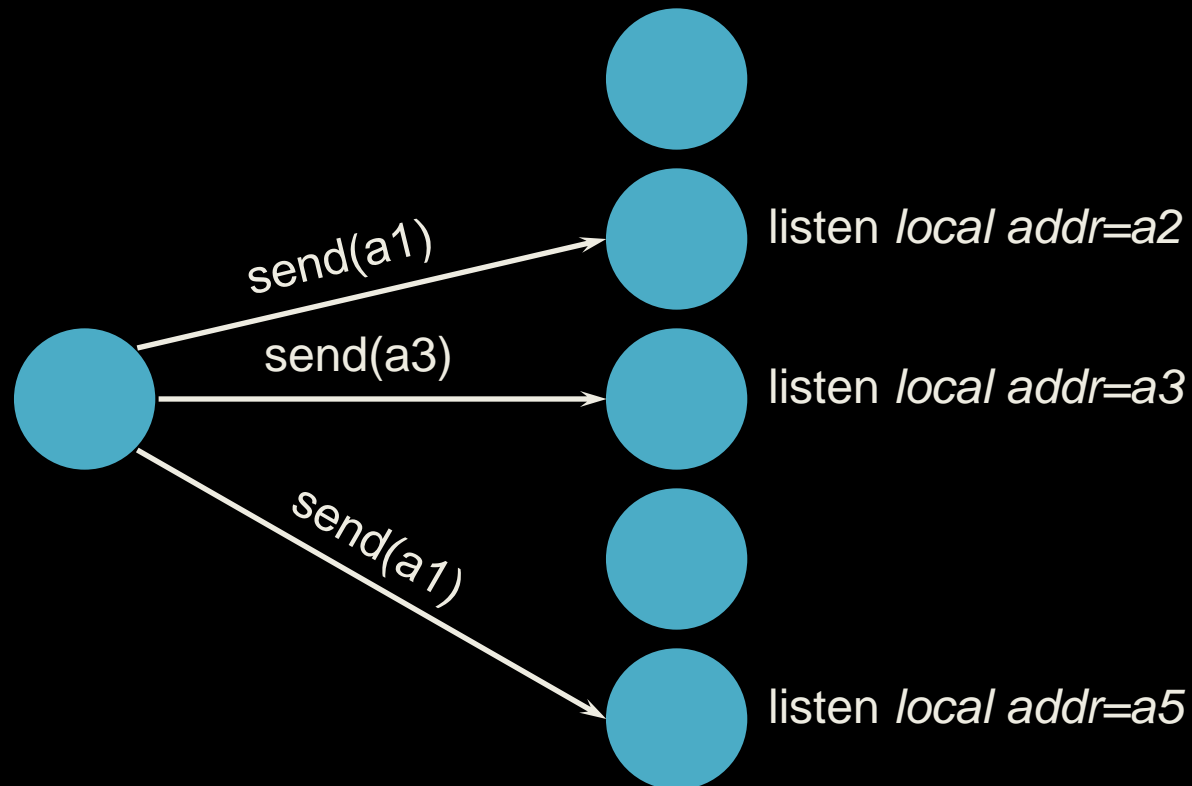


# Software: netcast

---

## Multiple unicasts (**netcast**)

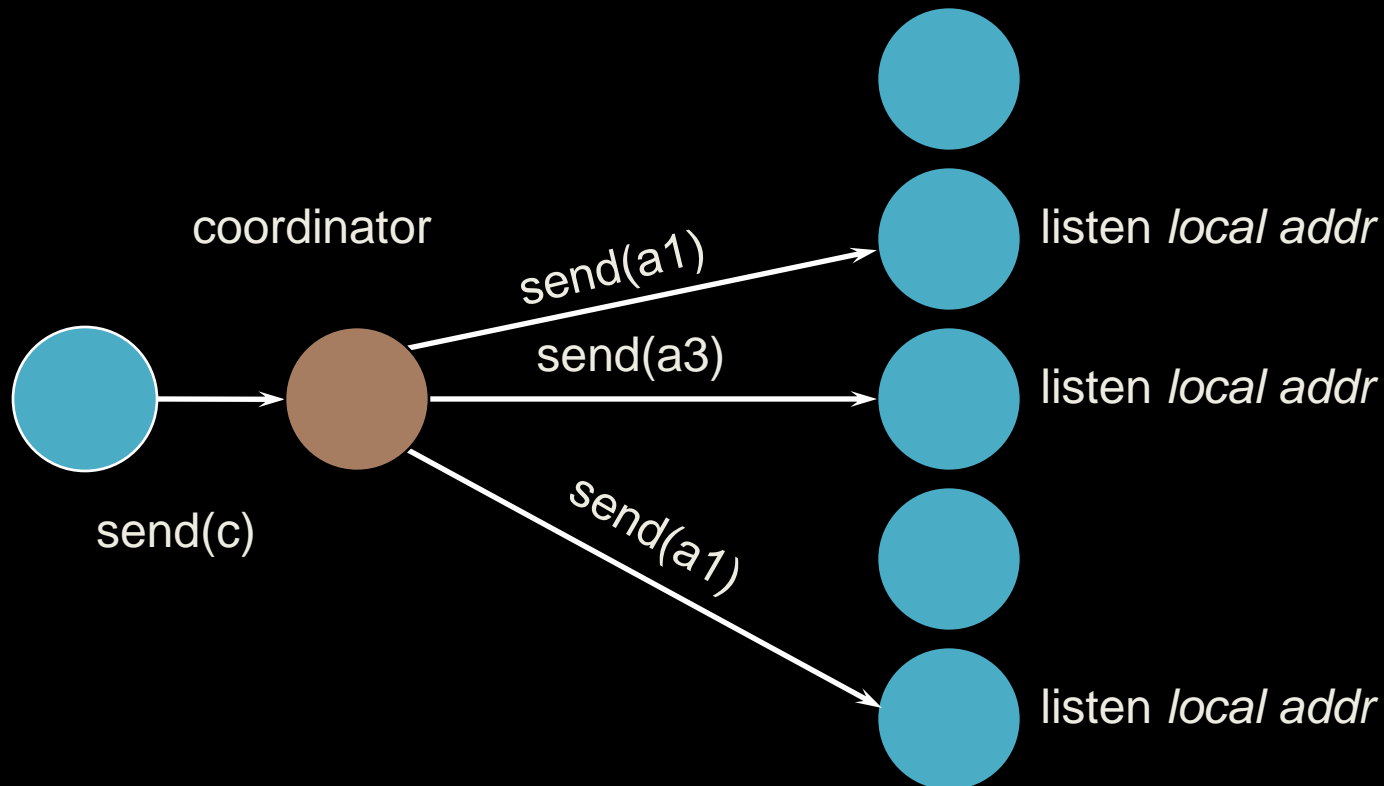
- Sender knows group members



# Software: hierarchical

## Multiple unicasts via group coordinator

- coordinator knows group members



# Reliability of multicasts

# 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 (2-phase commit variation)

---

## Retry through network failures & system downtime

Sender and receivers maintain **persistent log**

1. **Send message to all group members**
  - Each receiver acknowledges message
  - Saves message and acknowledgement in log
  - Does not pass message to application
2. **Sender waits for **all** acknowledgements**
  - Retransmits message to non-responding members
    - Again and again... until response received
3. **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

---

## Best effort

- Assume sender will remain alive
  - Retransmit undelivered messages
- 
- Send message
  - Wait for acknowledgement from each group member
  - Retransmit to non-responding members

# Unreliable multicast

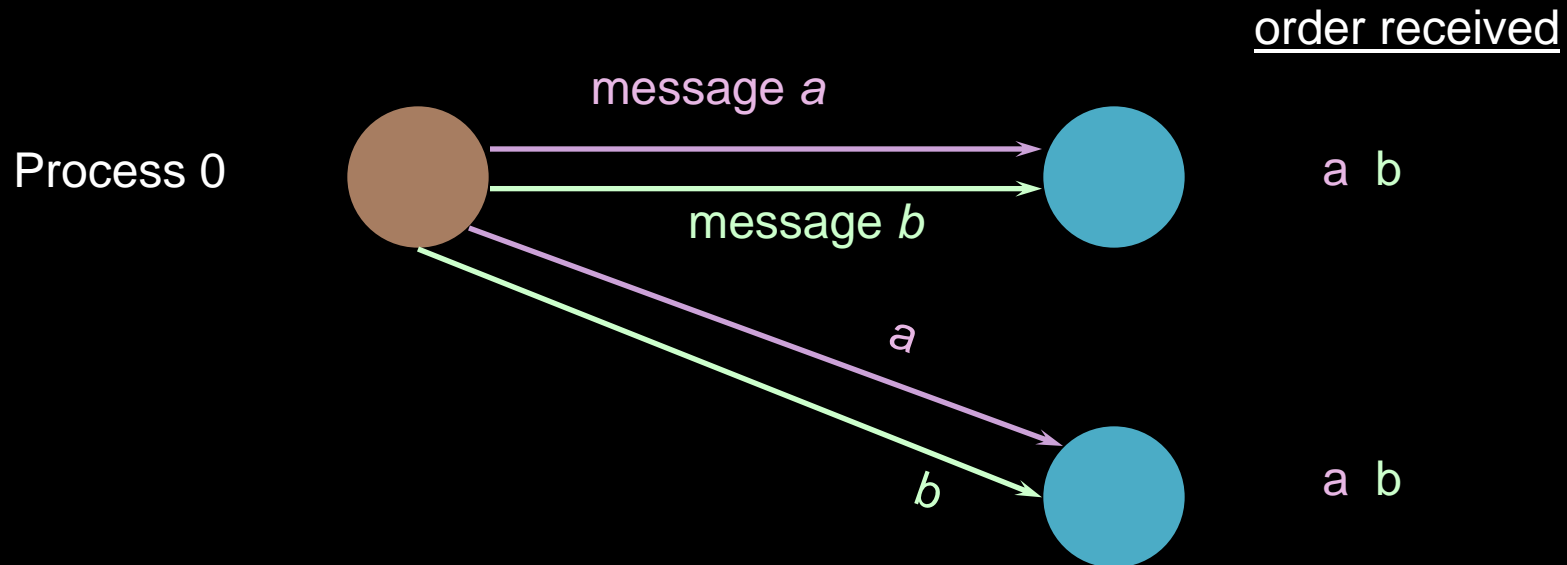
---

- Basic multicast
- Hope it gets there

# Message ordering

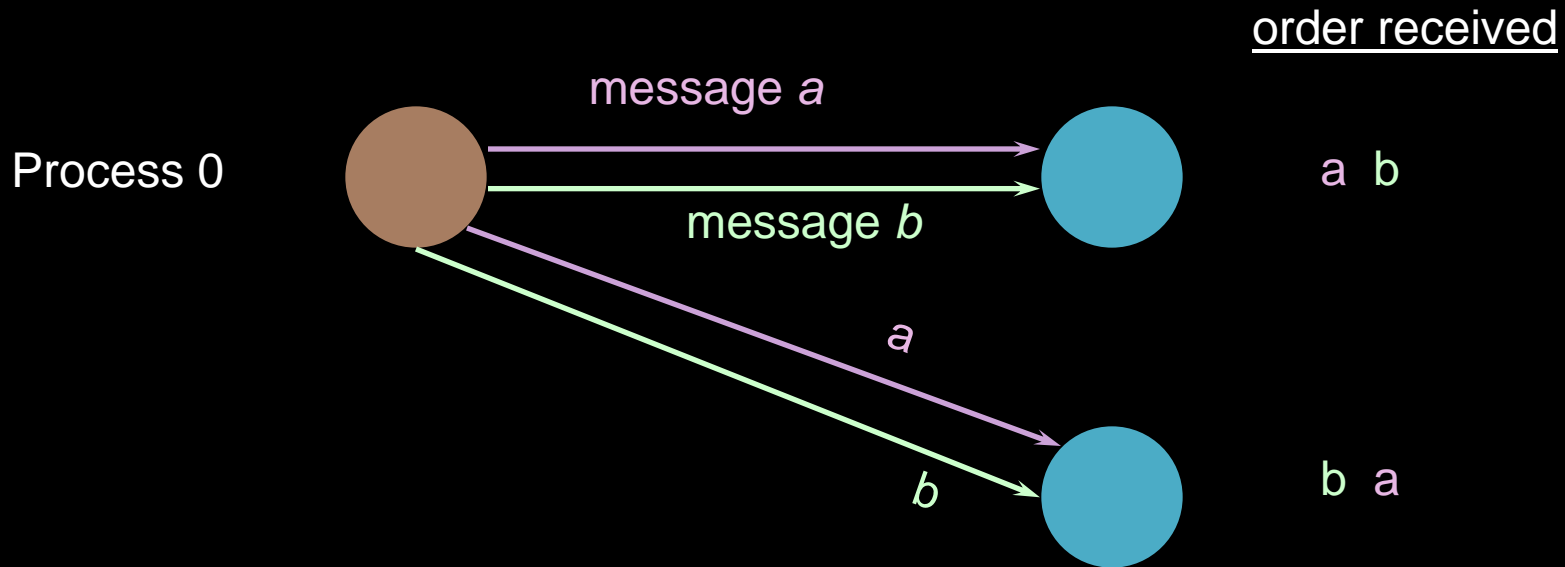
# Good Ordering

---



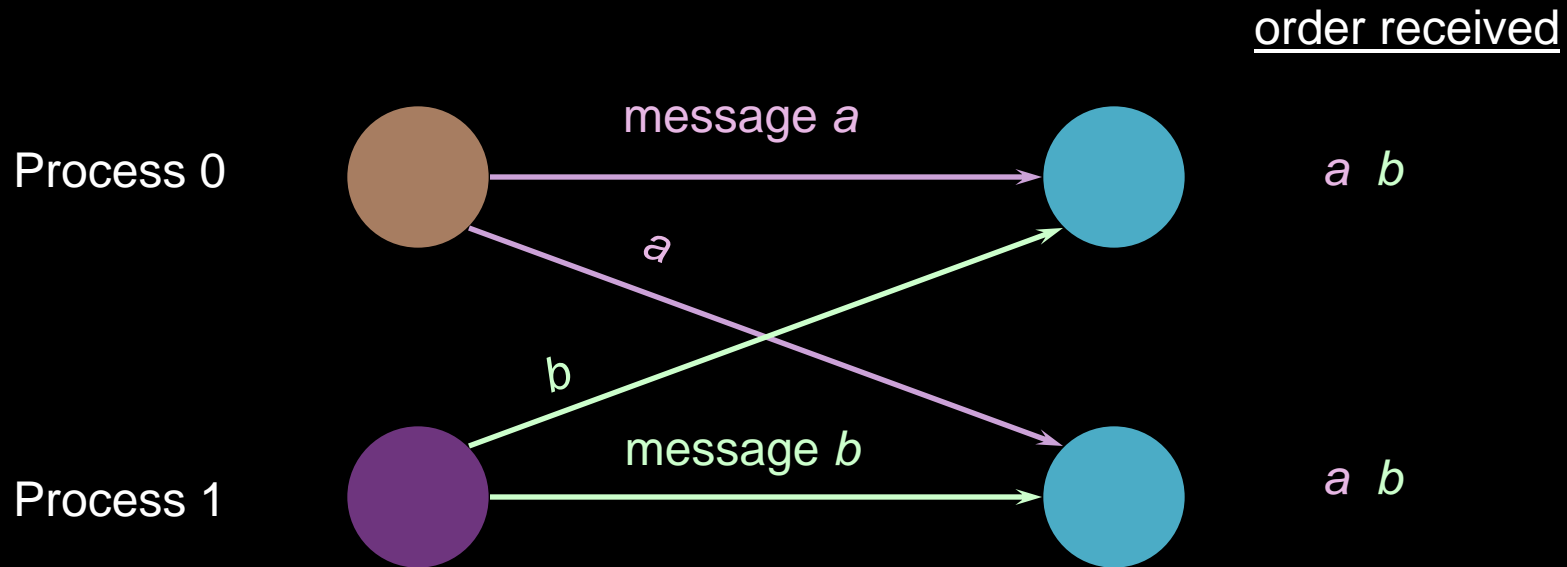
# Bad Ordering

---



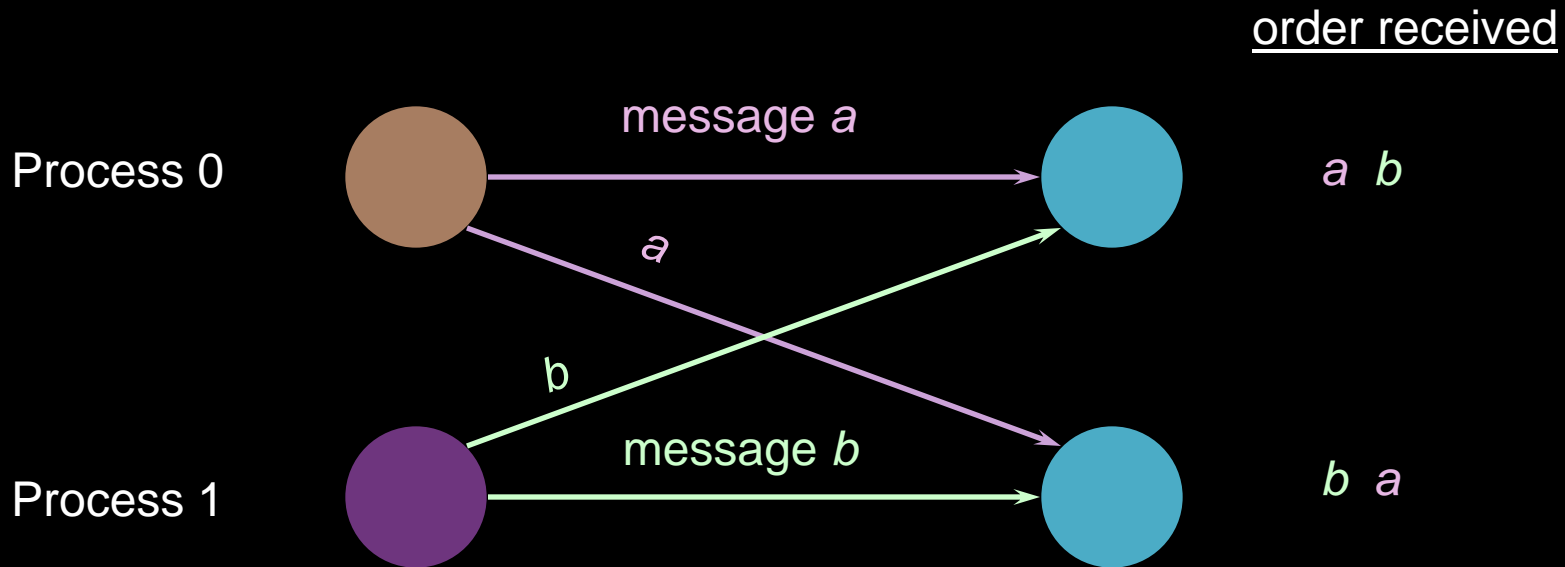
# Good Ordering

---



# Bad Ordering

---

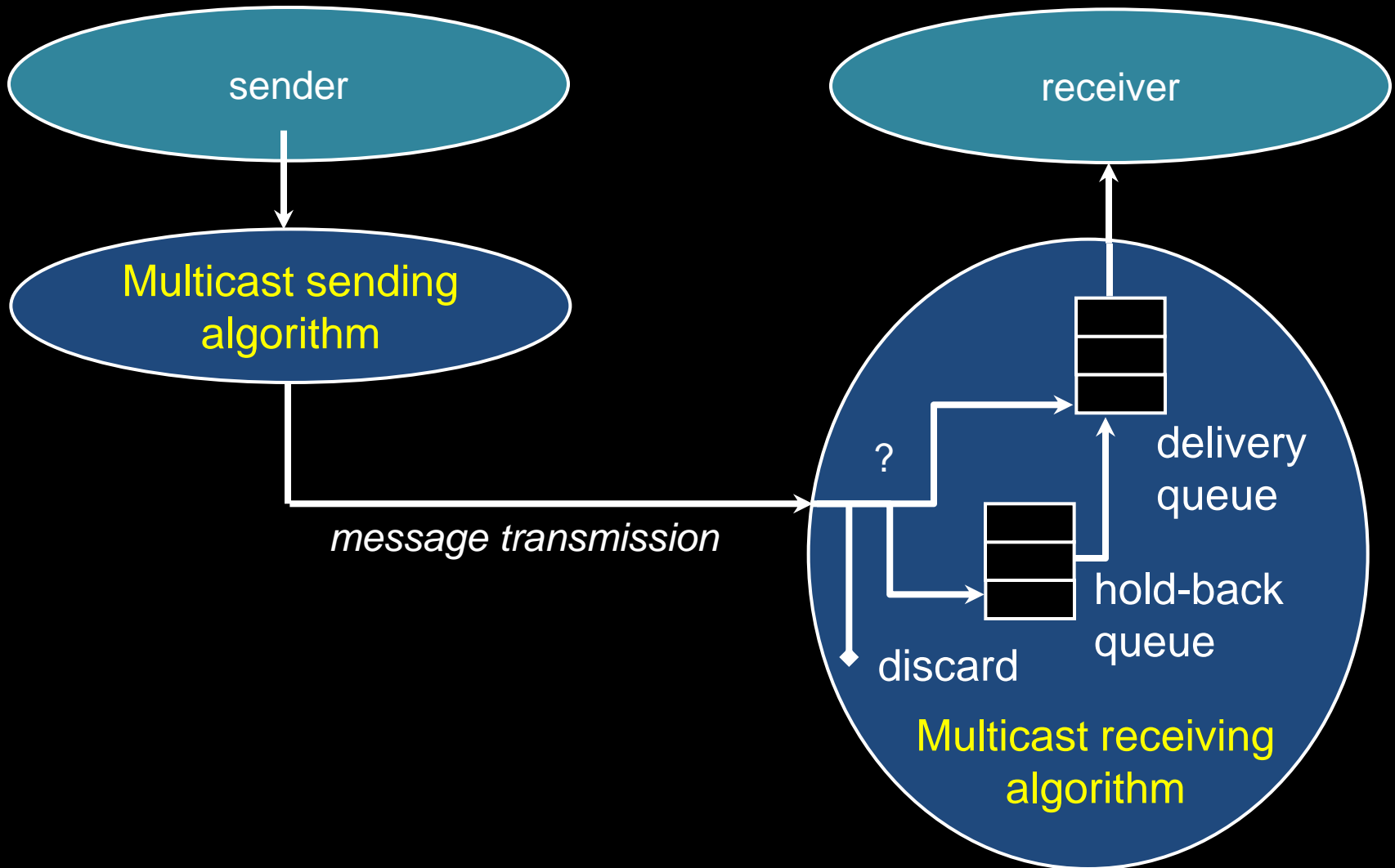


# Sending versus Delivering

---

- Multicast receiver algorithm decides when to *deliver* a message to the process.
- A received message may be:
  - **Delivered immediately**  
(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

1. If a process sends  $m$  before  $m'$  then any other process that delivers  $m'$  will have delivered  $m$ .
2. 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

# Causal 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  $m$ .

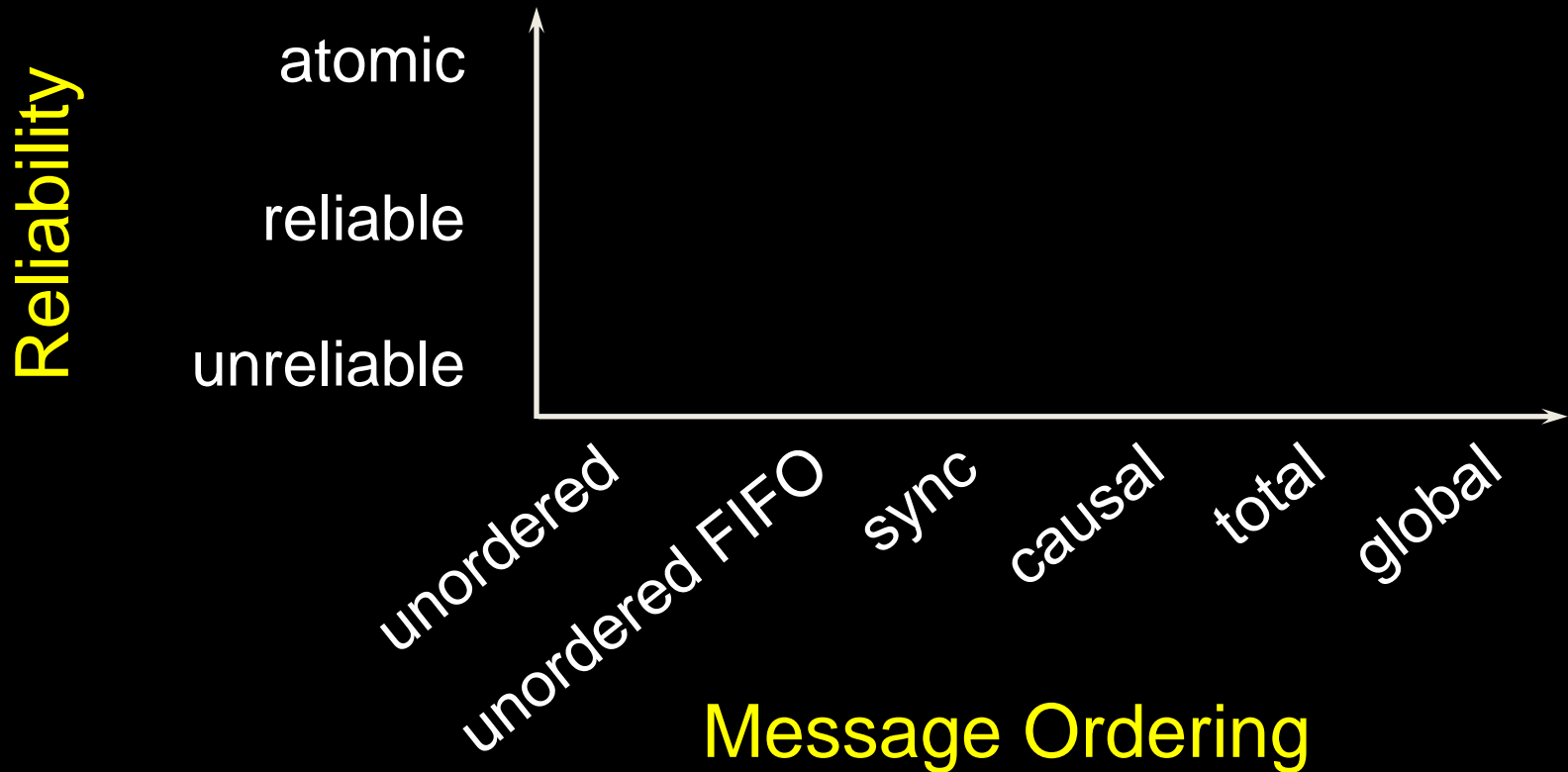
# Unordered multicast

---

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

# Multicasting considerations

---



# IP Multicasting

# IP Broadcasting

---

- 255.255.255.255
  - Limited broadcast: send to all connected networks
- Host bits all 1 (128.6.255.255, 192.168.0.255)
  - Directed broadcast on subnet

# IP Multicasting

---

Class D network created for IP multicasting



224.0.0.0/4

224.0.0.0 – 239.255.255.255

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

# IP multicast addresses

---

- Addresses chosen arbitrarily
  - Well-known addresses assigned by IANA
    - Internet Assigned Numbers Authority
    - Originally RFC 1340
    - See <http://www.iana.org/assignments/multicast-addresses/multicast-addresses.xml>
    - Similar to ports – service-based allocation
      - FTP: port 21, SMTP: port 25, HTTP: port 80
- |                |                                 |
|----------------|---------------------------------|
| 224.0.0.1:     | all systems on this subnet      |
| 224.0.0.2:     | all multicast routers on subnet |
| 224.0.23.173:  | Philips Health                  |
| 224.0.23.52:   | Amex Market Data                |
| 224.0.12.0-63: | Microsoft & MSNBC               |

# LAN (Ethernet) multicasting

---

- LAN cards support multicast in one (or both) of two ways:
  - Packets filtered based on *hash(multicast\_address)*
    - Some unwanted packets may pass through
    - Simplified circuitry
  - Exact match on small number of addresses
    - If host needs more, put LAN card in multicast promiscuous mode
      - Receive all hardware multicast packets
- In either case
  - Device driver must check to see if the packet is really targeted to the system

# LAN (Ethernet) multicasting example

---

## Intel 82546EB Dual Port Gigabit Ethernet Controller

- 10/100/1000 BaseT Ethernet

### Supports:


- 16 exact MAC address matches
- 4096-bit hash filter for multicast frames
- promiscuous unicast & promiscuous multicast transfer modes

# IP multicast on a LAN

---

- Sender specifies class D address in packet
- Driver must **translate** 28-bit IP multicast group to multicast Ethernet address
  - IANA allocated range of Ethernet MAC addresses for multicast
  - Copy least significant 23 bits of IP address to MAC address
    - 01:00:5e:xx:xx:xx
- Send out **multicast Ethernet packet**
  - Payload contains multicast IP packet

Bottom 23 bits  
of IP address



# IP multicast on a LAN

---

## Joining a multicast group

### Receiving process:

- Notifies IP layer that it wants to receive datagrams addressed to a certain host group
- Device driver must enable reception of Ethernet packets for that IP address
  - Then filter exact packets

# Beyond the physical network

---

Packets pass through routers which bridge networks together

**Multicast-aware router** needs to know:

- *are any hosts on a LAN that belong to a multicast group?*

## IGMP:

- Internet Group Management Protocol
- Designed to answer this question
- RFC 1112 (v1), 2236 (v2), 3376 (v3)

# IGMP v1

---

- Datagram-based protocol
- Fixed-size messages:
  - 20 bytes header, 8 bytes data
    - 4-bit version
    - 4-bit operation (1=query by router, 2=response)
    - 16-bit checksum
    - 32-bit IP class D address

# Joining multicast group with IGMP

---

- Machine sends IGMP report:
  - “I’m interested in this multicast address”
- Each multicast router broadcasts IGMP queries at regular intervals
  - See if any machines are still interested
  - One query per network interface
- When machine receives query
  - Send IGMP response packet for each group for which it is still interested in receiving packets

# Leaving a multicast group with IGMP

---

- No response to an IGMP query
  - Machine has no more processes which are interested
- Eventually router will stop forwarding packets to network when it gets no IGMP responses

# IGMP enhancements

---

- IGMP v2
  - *Leave group* messages added
  - Useful for high-bandwidth applications
- IGMP v3
  - Hosts can specify list of hosts from which they want to receive traffic.
  - Traffic from other (unwanted) hosts is blocked by the routers and hosts.

# IP Multicast in use

---

- Initially exciting:
  - Internet radio, NASA shuttle missions, collaborative gaming
- But:
  - Few ISPs enabled it
  - Required tapping into existing streams (not good for on-demand content)
  - Industry embraced unicast instead

# IP Multicast in use: IPTV

---

- IPTV is emerging as the biggest user of IP multicast
- Cable TV systems: aggregate bandwidth ~ 4.5 Gbps
  - Video streams: MPEG-2 or MPEG-4 (H.264)
  - MPEG-2 HD: ~30 Mbps
  - MPEG-4 HD: ~6-9 Mbps; DVD quality: ~2 Mbps

# IP Multicast in use: IPTV

---

- Traffic is within the provider's network
  - QoS: typically mix of ATM and/or IP
    - 2.5 Mbps VBR video
    - 256 kbps CBR voice
    - Remainder: ABR for IP traffic
  - Unicast for video on demand
  - Multicast for live content
    - When you select a channel, you join a multicast group via IGMPv2
      - Local office checks if you are authorized.
      - If yes, routers add the user to the group
    - Burst of unicast data to get the I-frame to ensure 150 msec channel switching times.

The End