Group Communication

- A group is a collection of processes that act together in some system or user specified way.
- Several distributed applications requires that a message passing should also provide group communication.
- The key property that ,when a message is sent to the group; all the members of the group receive it.
- Depending upon the number of sender and receivers, group communication can be classified in following categories:
  - 1 One to many
  - 2. Many to one
  - 3. Many to many

One – to - many communication

- In this scheme, there are multiple receivers for a message sent by a single sender.
- One-to-many scheme is also known as multicast communication.
- Special case of multicast communication is broadcast communication in which a message is received by all the processors connected to the network.
- Multicast/broadcast communication is very useful for several practical applications. For example, consider a server manager managing a group of server processes all , providing the same type of service. The server manager can multicast a message to all the server processes, requesting that a free server volunteer to serve the current request.

- It then selects the first server that responds. The server manager does not have to keep track of the free servers.
- Similarly, to locate a processor providing a specific service, an inquiry message may be broadcast. In this case, it is not necessary to receive an answer from every processor; just finding one instance of the desired service is sufficient.

Broadcast communication /Multi cast communication

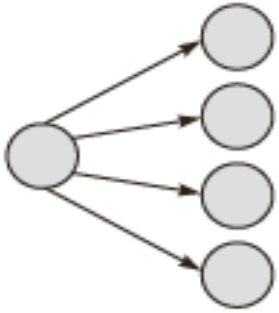

Figure 3.27 One-to-many communication

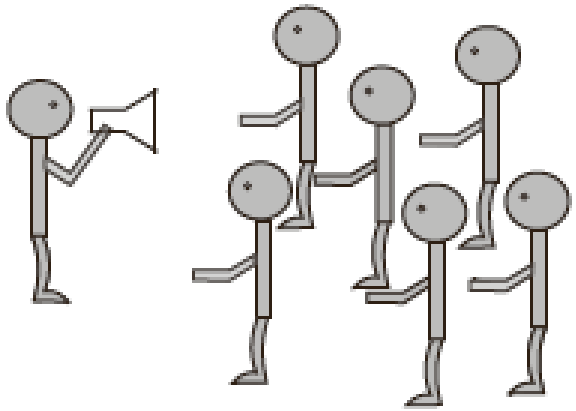

Figure 3-28 A broadcast sends a message to all the recipients

Group Management

- In case of group communication the communicating processes forms a group.
- Such a group may of either of two types-**close and open.**
- **Closed group** is the one, in which only members can send message, outside process cannot send a message to the group as a whole but can send to single member of a group.
- **Open group** is one in which any process in the system can send a message to the group as a whole.
- Message passing with group communication provides flexibility and create groups dynamically and to allow process join or leave groups dynamically.
- Centralized group server is used for this purpose but suffers from poor reliability and scalability.

Group management

- Centralized approach
- Distributed approach

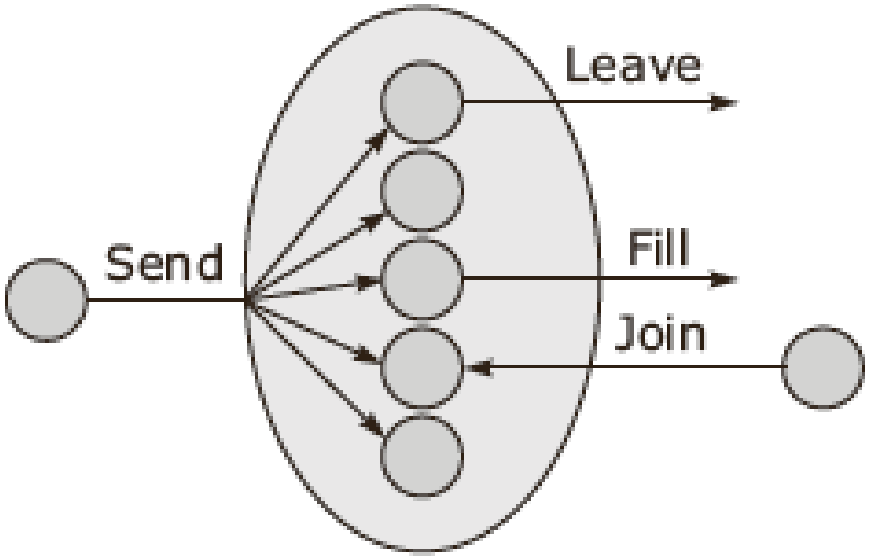

Figure 3-29 Group dynamics

Group Addressing

- A Two level naming scheme is used for group addressing.
- The high level group name is ASCII string that is independent of the location information on the process in the group.
- On the other hand, the low level group names depends on the underlying hardware.
- Multicast address is a special network address, the packet is delivered automatically to all machines listening to address. S
- ome networks do not have facility to create multicast address may have broadcast facility.
- Network with broadcast facility declares a certain address as a broadcast address and It broadcast message to all machines of the network.
- If network doesn't support any addressing among two then one to one communication is used to implement group communication. First two methods send single packet but one to one sends many, creating much traffic.

Message delivery to receiving process

- User application uses high level group names in program.The Centralized group server maintains a mapping of high level group names to their low level names, when sender sends message to the group. With high level name, kernel of server machine asks the group servers low level name and list of process identifiers of all the processes of each group.
- When a packet with a group address is sent then the kernel of the sending machine contacts the group server to obtain the low level name of the group and list of the processes belongs to that group and forwards message to those processes belonging to its own machine. If none of ID's is matching packet is discarded.

Buffered and unbuffered multicast

- In buffered multicast ,a copy of message is sent to each process of multicast group and message is buffered until it is accepted by process.
- In unbuffered multicast,message is not buffered for the receiving process.

Flexible Reliability in Multicast Communication.

- Different applications require different degrees of reliability.
- Thus, the Sending of a multicast message can specify the number of receivers from which a response message is expected.
- In one to many communication, the degree of reliability is expressed in 4 forms.
- **0 – Reliable: - No response is expected at all.**
- **1 – Reliable: - Sender expects response from any one of the receivers.**
- **M-out-of-n reliable :- Sender expects response from m of n receiver.**
- **All reliable : - Sender expects response from all.**

Reliability mechanism

- Classified based on number of receivers from which sender expects a response

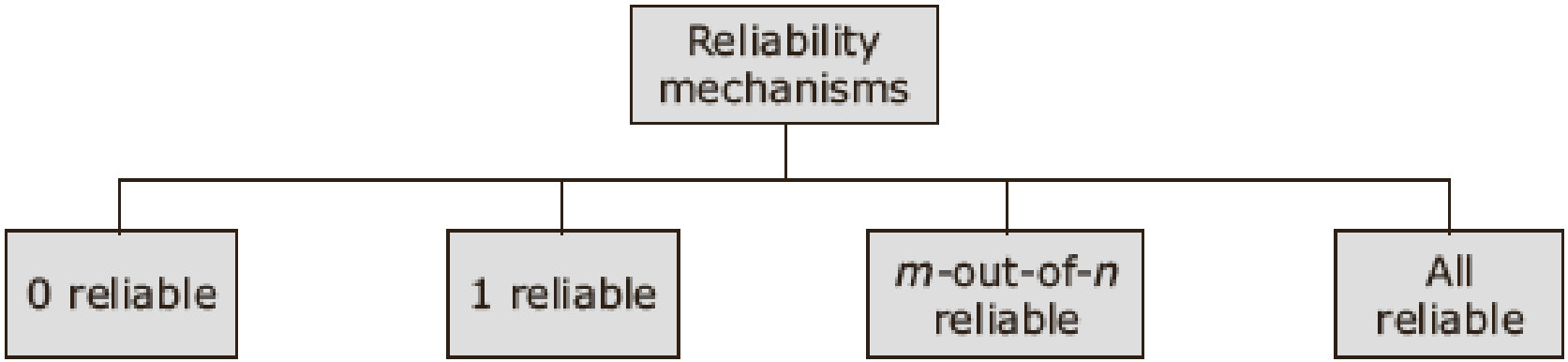

Figure 3-30 Reliability mechanisms

Atomic Multicast

- Atomic multicast has all or nothing property .That is ,when a message is sent to a group by atomic multicast,it is either received by all the processes that are a member of the group or else it is not received by any of them.all reliable form requires atomic multicast facility.

- Message passing system should support both atomic and non- atomic multicast facility. It should provide flexibility to sender to specify whether atomicity is required or not.It is difficult to implement atomic multicast if sender or receiver fails. Solution to this is fault tolerated atomic multicast protocol. In this protocol, each message has message identifier field to distinguish message and one field to indicate data message in atomic multicast message.

Many – to – one Communication

- In this type of communication, multiple sender sends message to a single receiver.
- Single receiver may be selective or non-selective.
- A Selective receiver specifies unique sender message exchange takes place only if the sender sends the message.
- Non-selective receiver specifies set of sender if any from that sends message then message exchange takes place.
- No determinism issues need to be handled here. Rest all factors are the same as for one – to – many communication.

Many –to – many

- In this scheme ,multiple senders sends messages to multiple recievers.
- One to many, many to one, are implicit in this scheme.
- Ordered message delivery ensures that all messages are delivered to all receivers in an order acceptable to the application.
- The commonly used semantics for ordered delivery of multicast messages are:
- 1) Absolute ordering
- 2) Consistent ordering
- 3) Casual

Message ordering

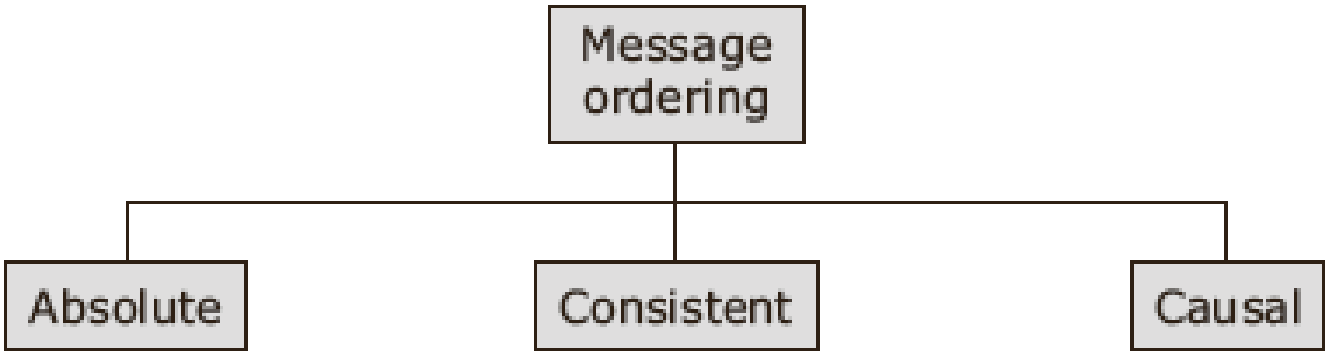

Figure 3-31 Types of message ordering

Message ordering: Absolute Ordering

It ensures that all messages delivered to all receiver in exact in which they are sent

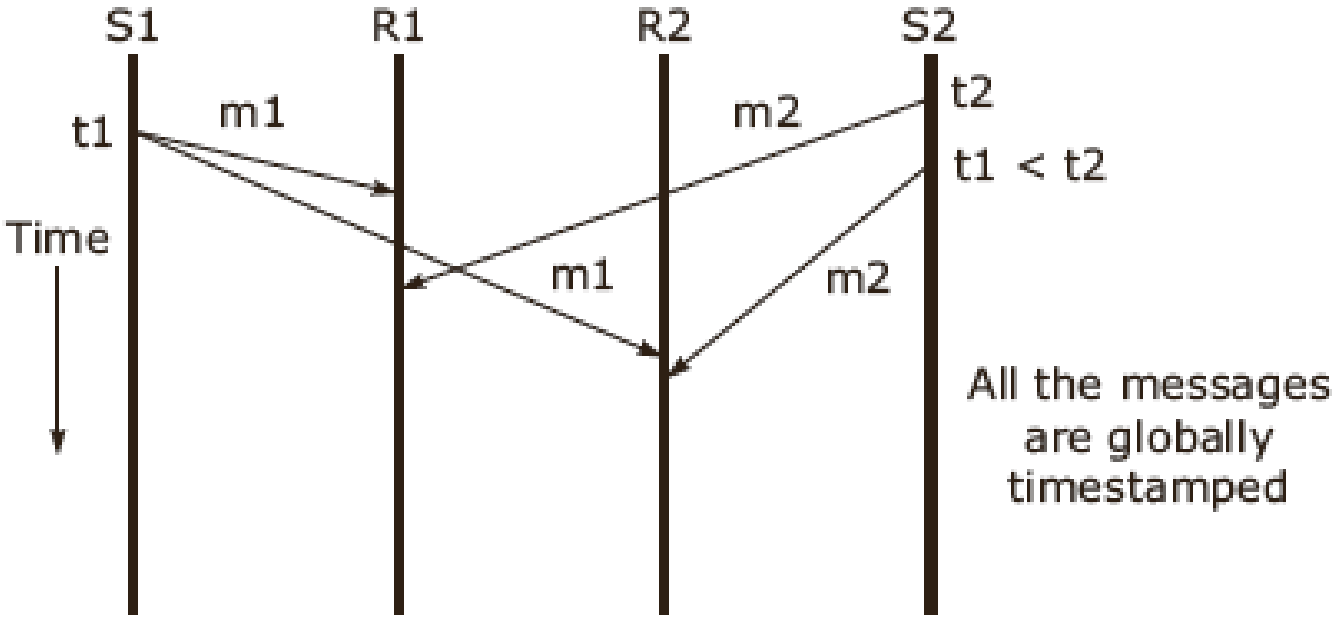

Figure 3-32 Absolute ordering

Message ordering: Consistent ordering

- Absolute ordering semantics requires globally synchronised clocks,which is difficult to implement.
- Moreover,many applications does not even need absolute according to function correctly.
- Consistent ordering semantics ensure that all messages are delivered to all receivers process in the same order and this order may be different from the order in which messages were sent.

Message ordering: Consistent ordering

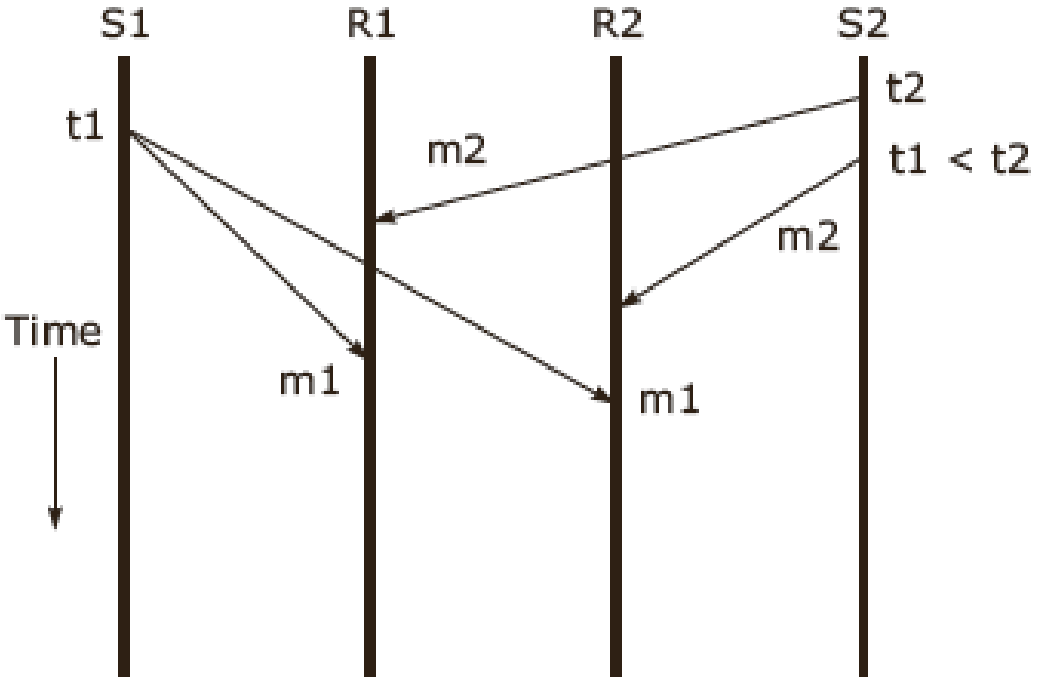

Figure 3-33 Consistent ordering

Causal ordering.

- For some application consistent ordering semantics is not necessary and even weaker semantics is acceptable.
- Therefore, an application can have better performance if the message passing system used supports a weaker ordering semantics that is acceptable to the application.
- One such ordering semantics is is called causal ordering.

Message ordering: Causal ordering

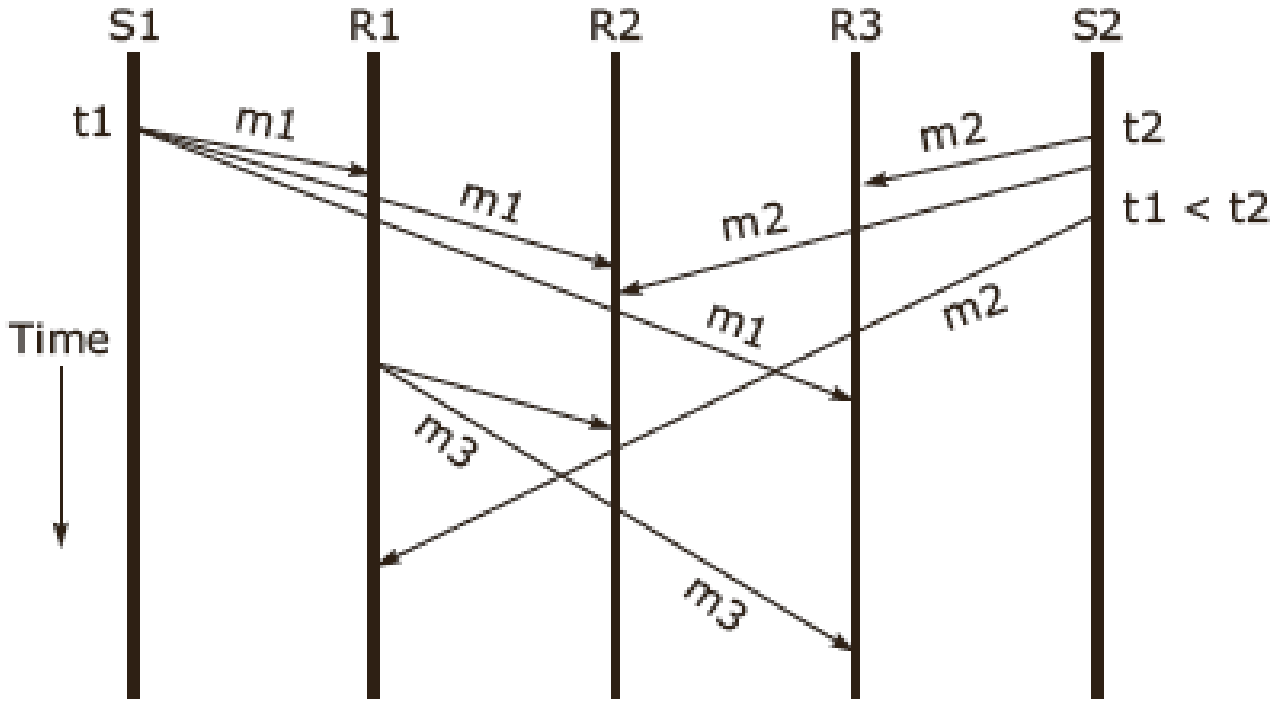

Figure 3-34 Causal ordering

Happens before relation:

1.If a and b are events in the same process, and a comes before b, then a → b is true.

2. If a :event of message sent

b : event of receipt of the same message

then a → b is trues

Happens before the relation is Transitive: If a → b and b → c then a → c.