*Indirect communication* is defined as communication between entities in a

distributed system through an intermediary with no direct coupling between the sender

and the receiver(s).

*Space uncoupling,* in which the sender does not know or need to know the identity

of the receiver(s)

*Time uncoupling* the sender and receiver(s) do not need to exist at the same

time to communicate.

Group communication:

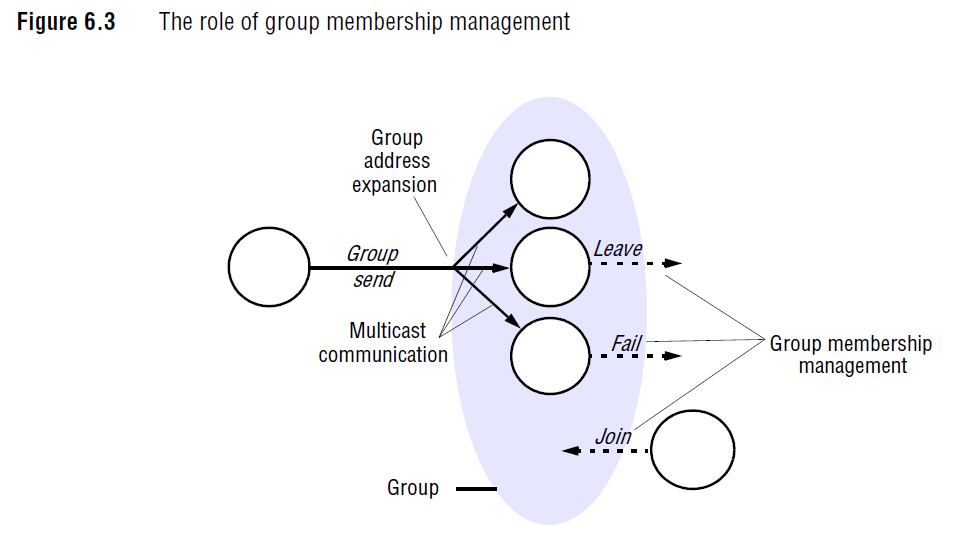
- Message is sent to a group and delivered to all members

- Sender doesn't know receiver's ID

- implemented over IP multicast (from one point to many)

- Message send to all members by a single operation

- There is a guarantee that message is delivered (no halfway cases)



To consider

- Open/closed groups: may processes outside the group send to it/ the close is the opposite

- Non/overlapping: entities may be members of multiple groups/ non is just in one group

Implementation issues

Reliability{

-Integrity: message is delivered without changes, no duplicate deliveries

-Validity: any message is delivered

-Agreement: if message is delivered to one client, it is delivered to all

}

Ordering

- FIFO

- Causal: considers causal relationships between messages

- Total: same delivery order for everyone

**Group membership management •**

*Providing an interface for group membership changes*:

*Failure detection*:

*Notifying members of group membership changes*

Pub Sub Systems:

- Sender sends messages to a broker

- Broker notifies remote observers

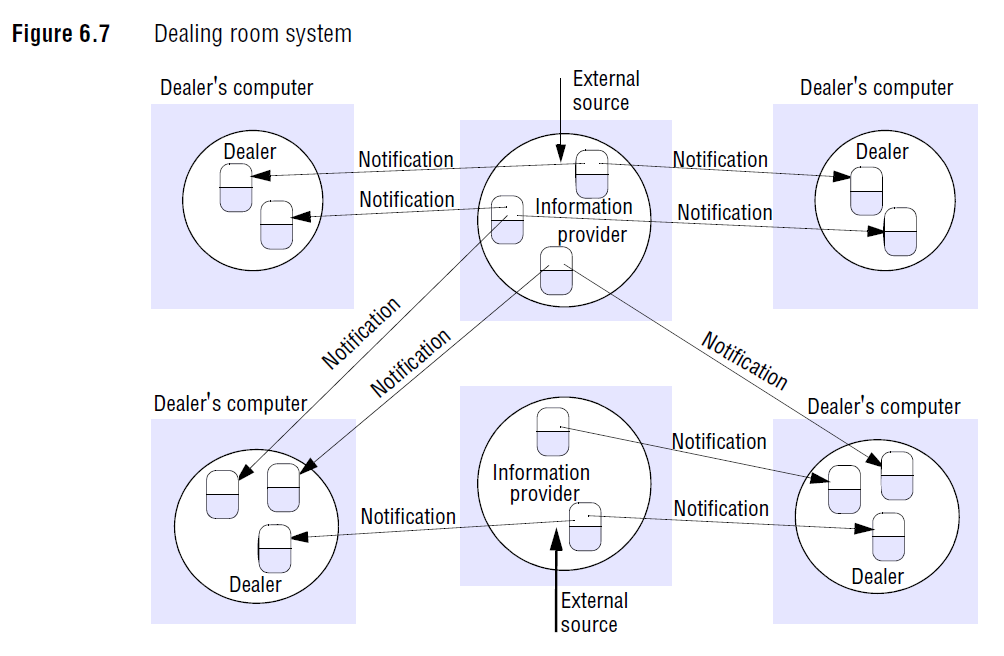
Observers can subscribe to:

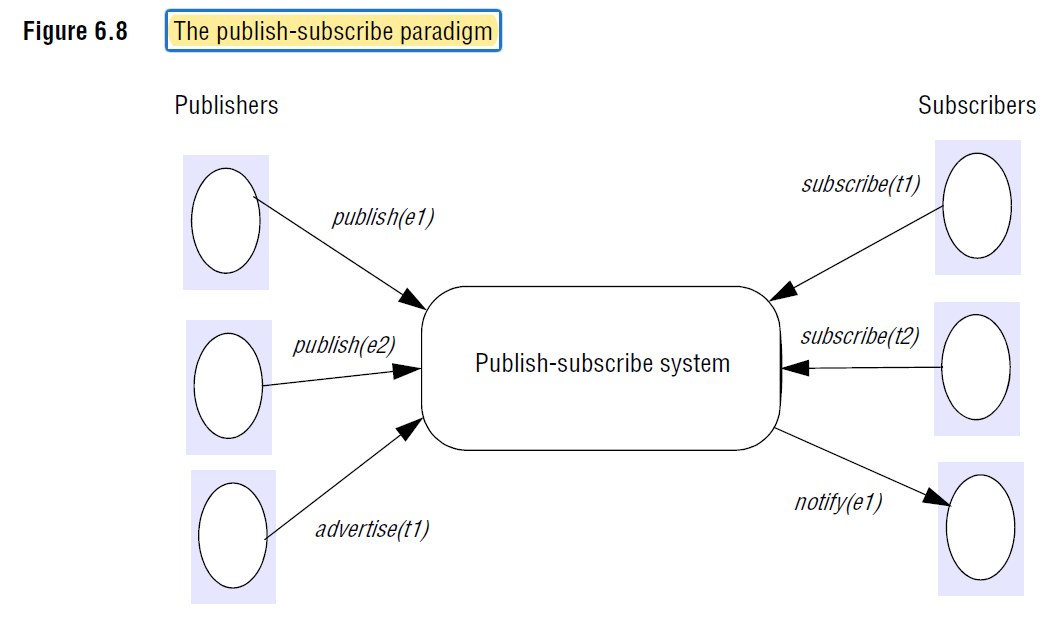
- Topic: based on one field in the message that comes from the publisher

- Channel: everything comes to the channel.

- Content based(event): based on range of fields

- Applications: RSS, financial info systems, monitoring, smart home





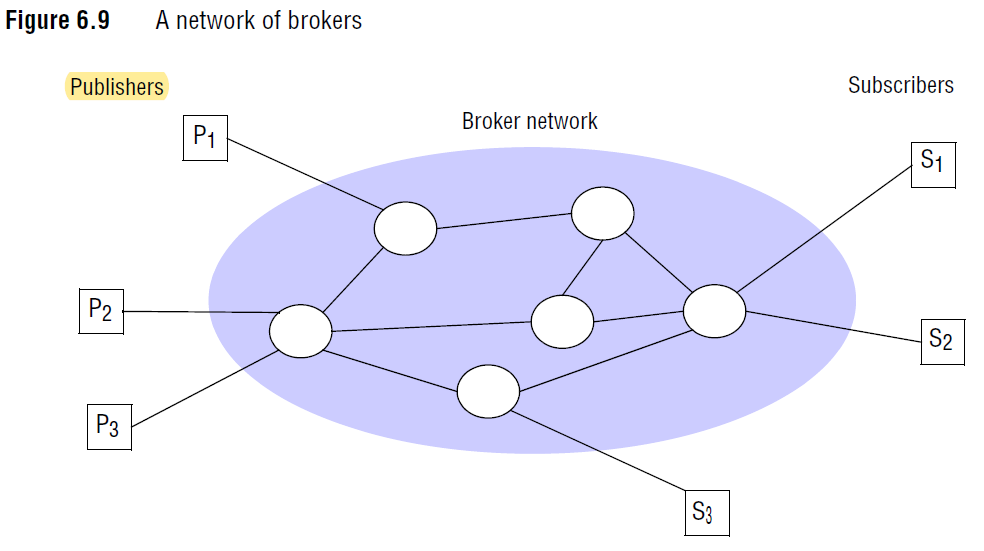
Characteristics

- Heterogeneity: subscribers can be Heterogeneity

- A synchronicity: publishers & subscribers are decoupled, time(possible) and space

Implementation issues

**Centralized versus distributed implementations**



Message Queues

- point-to-point

- sender places the message into the queue

- message is then removed by a single process(receiver).

Three style of receive:

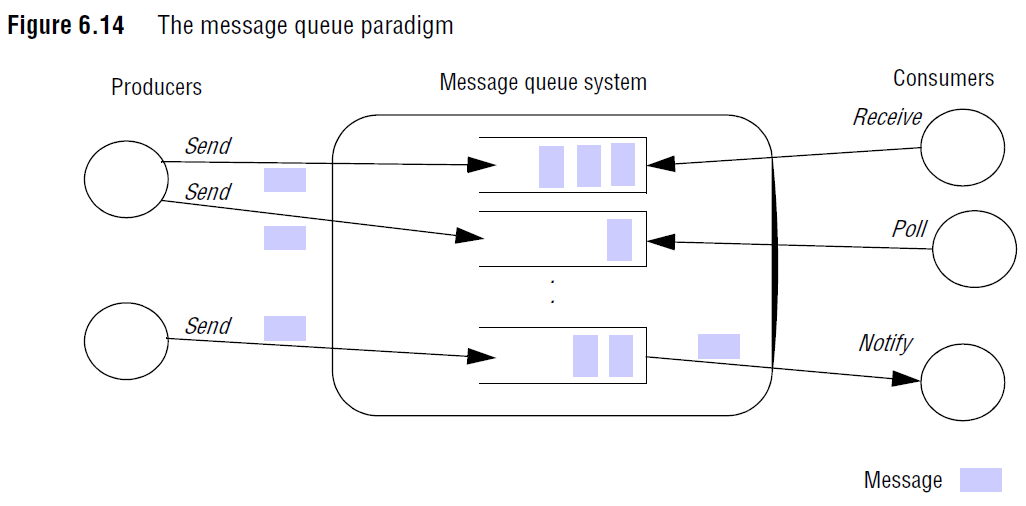
1. Queue notifies specific receiver(s)

2. Receivers poll queue repeatedly

3.Receive (blocking): the client blocks the server until the appropriate message is available

- You might get network timeouts

-This method is barely (not) used



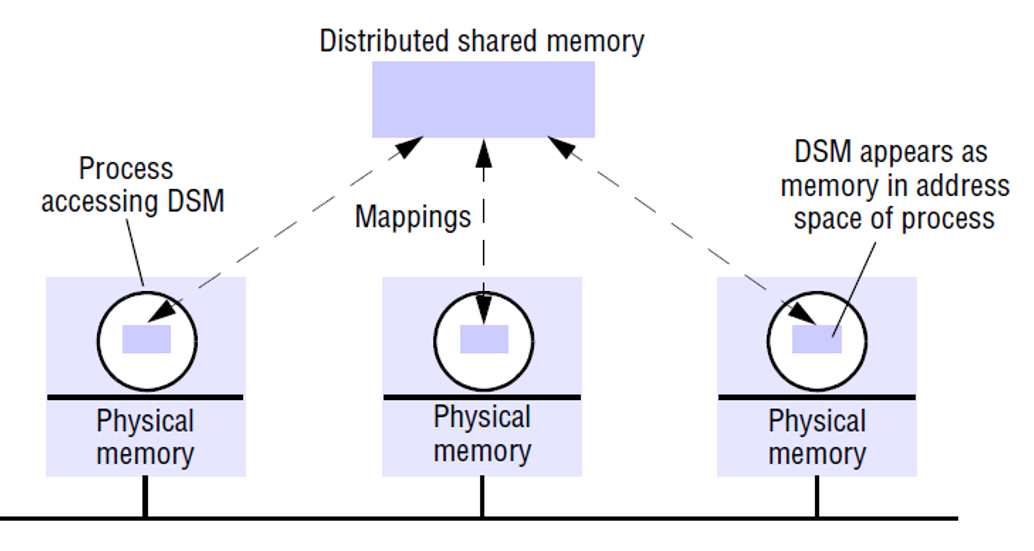
Distributed Shared Memory

(DSM) is an abstraction used for sharing data between computers that do not share physical memory

- RAM that everyone can read and write on

- Reader doesn't know when the information was written, or who left it there

- it is used for parallel systems, not so much for client-server



Tuple Spaces

Take means you will take the data

Read means the data will stay in the tuple

- Space & time uncoupled

