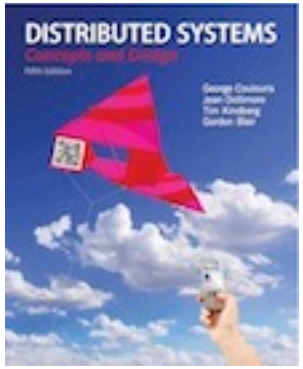
Indirect Communication



From **Coulouris, Dollimore, Kindberg and Blair**

Distributed Systems:
Concepts and Design

- Group Communication
- Publish-Subscribe
- Message Queues
- Share Memory

Objectives

- To understand the key concepts of group communications, publish-subscribe systems, message queue, tuple and distributed share memory

Indirect communication

- Indirect communication: communication through an intermediary with no direct coupling between the sender and the receiver(s).
  - Space uncoupling: the sender does not know or need to know the identity of the receiver(s), and vice versa
  - Time uncoupling: the sender and receiver(s) can have independent lifetimes
- It is often used in distributed systems where change is anticipated
- Its main disadvantage is the performance overhead introduced by the added level of indirection and it is more difficult to manage

Space and time coupling in distributed systems

	Time-coupled	*Time-uncoupled*
Space coupling	*Properties*: Communication directed towards a given receiver or receivers; receiver(s) must exist at that moment in time *Examples*: Message passing, remote invocation (see Chapters 4 and 5)	*Properties*: Communication directed towards a given receiver or receivers; sender(s) and receiver(s) can have independent lifetimes *Examples*: See Exercise 15.3
Space uncoupling	*Properties*: Sender does not need to know the identity of the receiver(s); receiver(s) must exist at that moment in time *Examples*: IP multicast (see Chapter 4)	*Properties*: Sender does not need to know the identity of the receiver(s); sender(s) and receiver(s) can have independent lifetimes *Examples*: Most indirect communication paradigms covered in this chapter

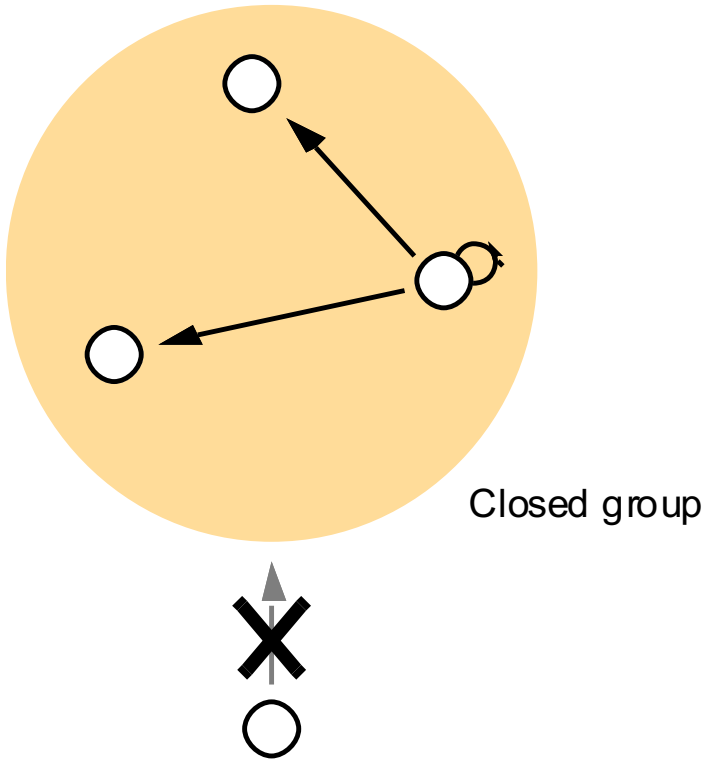
Group communication

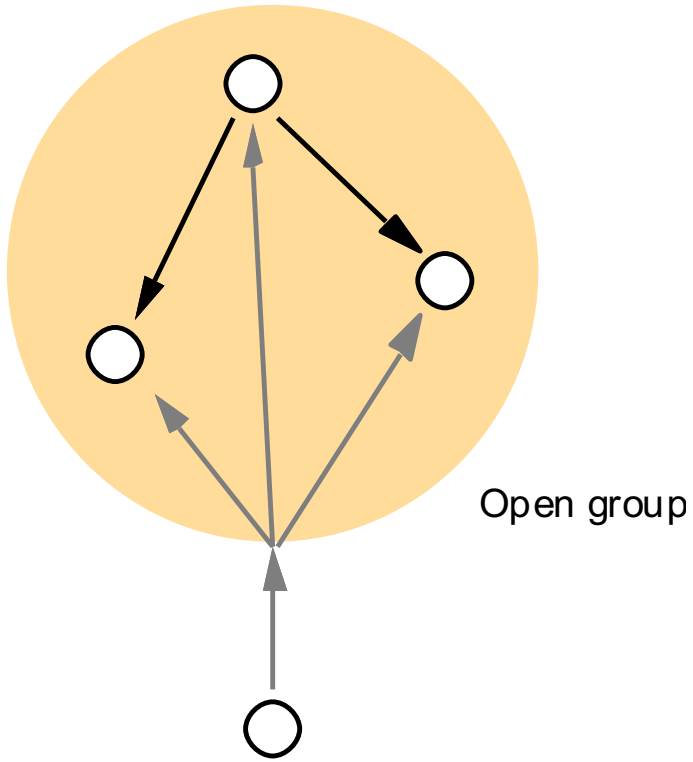
- It offers a service whereby a message is sent to a group and then this message is delivered to all members of the group
- The sender is not aware of the identities of the receivers
- Group communication represents an abstraction over multicast communication adding significant extra value in terms of managing group membership, detecting failures and providing reliability and ordering guarantees
- Areas of applications:
  - Reliable dissemination of information to a large numbers of clients
  - Support for collaborative applications
  - Support for a range of fault-tolerance strategies
  - Support for system monitoring and management,

The programming model

- The central concept is that of a group with associated group membership, whereby processes may join or leave the group
- Processes can then send a message to this group and have it propagated to all members of the group with certain guarantees in terms of reliability and ordering
- The essential feature of group communication is that a process issues only one multicast operation to send a message to each of a group of processes

Open and closed groups

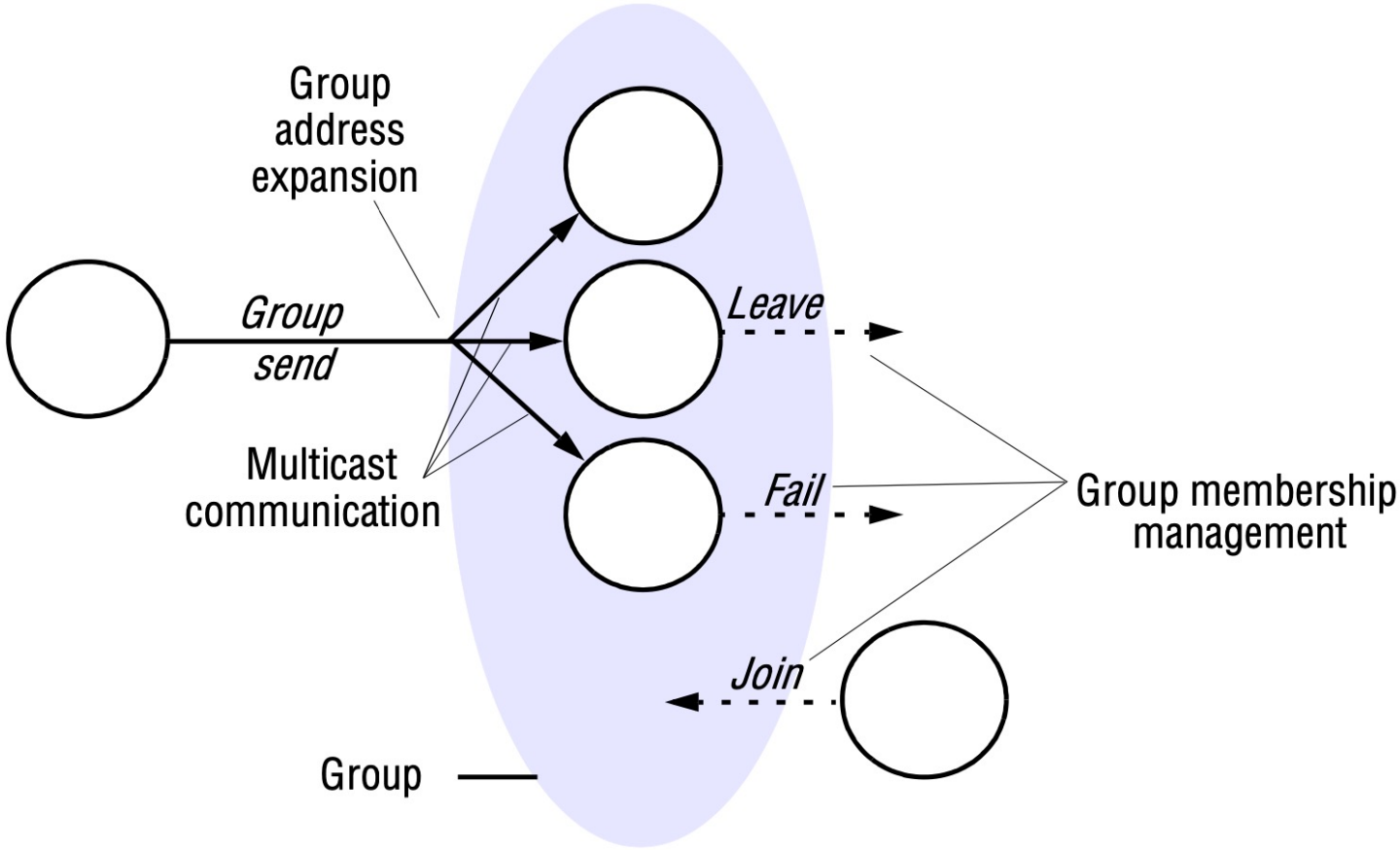




Implementation issues

- Reliability and ordering in multicast
  - Integrity: the message received is the same as the one sent, and no messages are delivered twice
  - Validity: guarantees that a message sent will eventually be delivered
  - Agreement: if the message is delivered to one process, then it is delivered to all processes in the group.
- Group communication services offer ordered multicast:
  - FIFO ordering
  - Causal ordering
  - Total ordering

Group membership management



Implementation issues

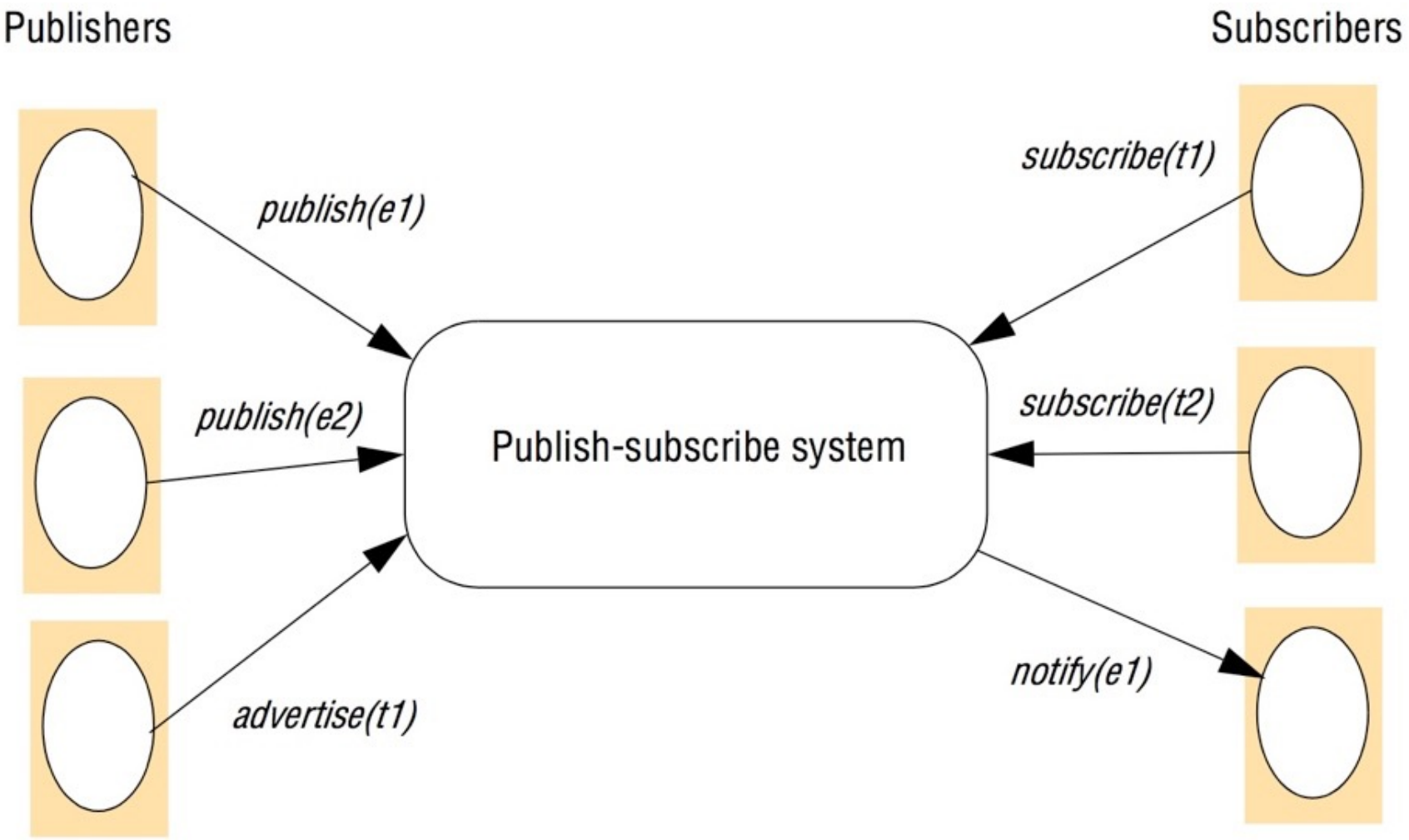
- Main tasks:
  - Providing an interface for group membership changes
  - Failure detection
  - Notifying members of group membership changes
  - Performing group address expansion. it can decide consistently where to deliver any given message, even though the membership may be changing during delivery.

- Group communication is most effective in small-scale and static systems and does not operate as well in larger-scale environments or environments with a high degree of volatility

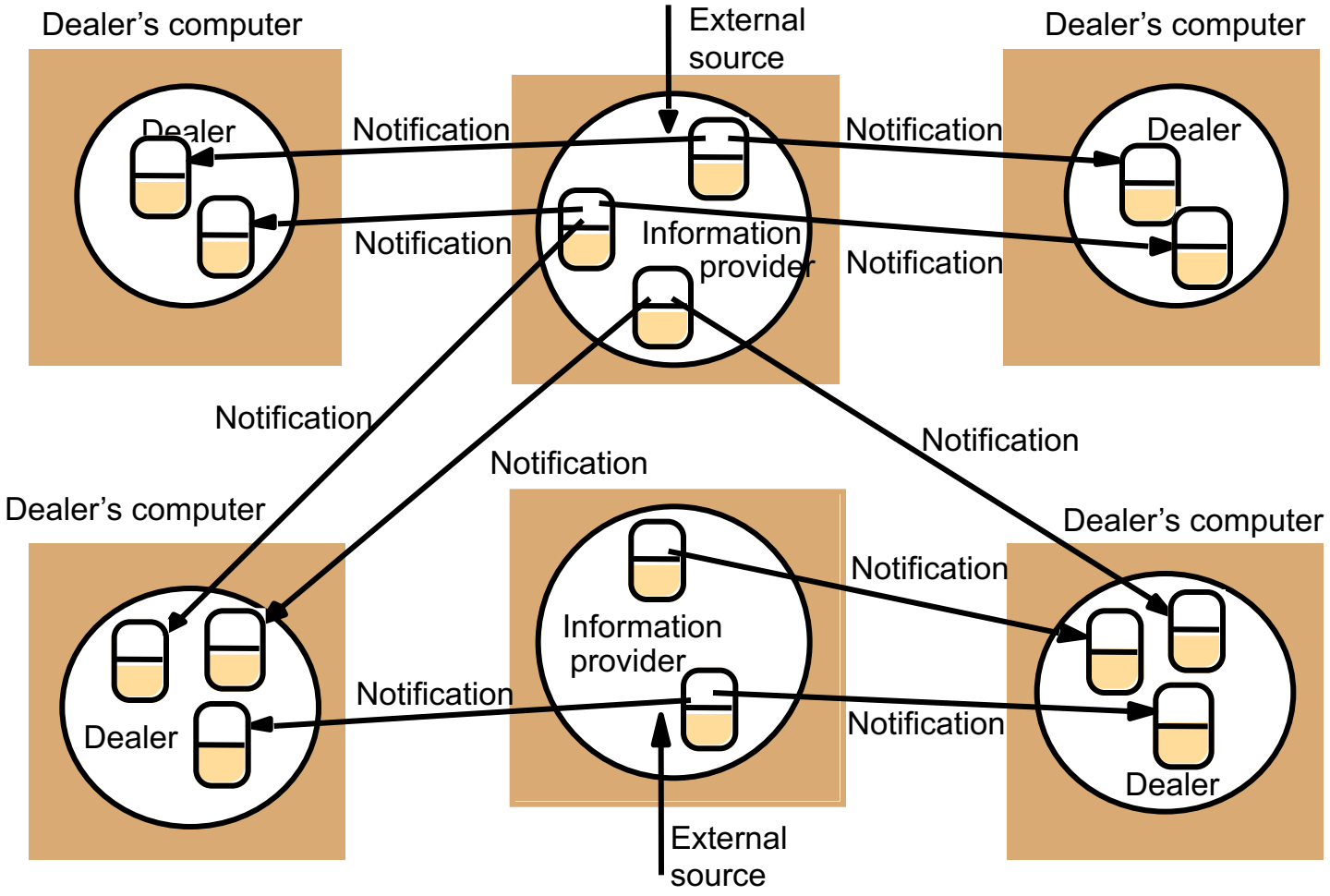
Publish-subscribe systems

- Also referred to as distributed event-based systems
- Publisher publish structured events to an event service
- Subscribers express interest in particular events through subscriptions which can be arbitrary patterns over the structured events
- Applications:
  - Financial information systems
  - Other areas with live feeds of real-time data
  - Support for cooperative working, where a number of participants need to be informed of events of shared interest
  - Support for ubiquitous computing
  - A broad set of monitoring applications

The publish-subscribe paradigm



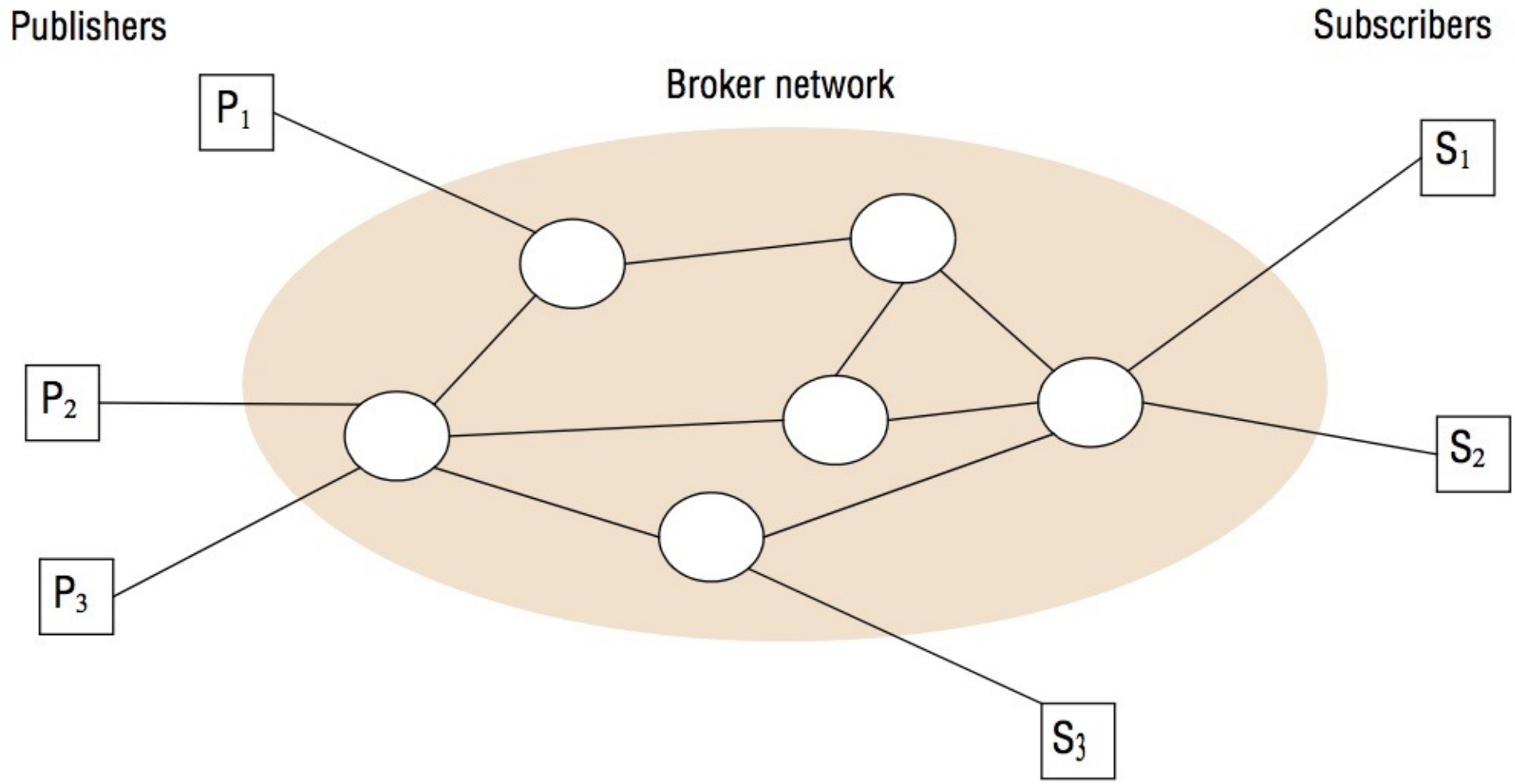
Example: Dealing room system



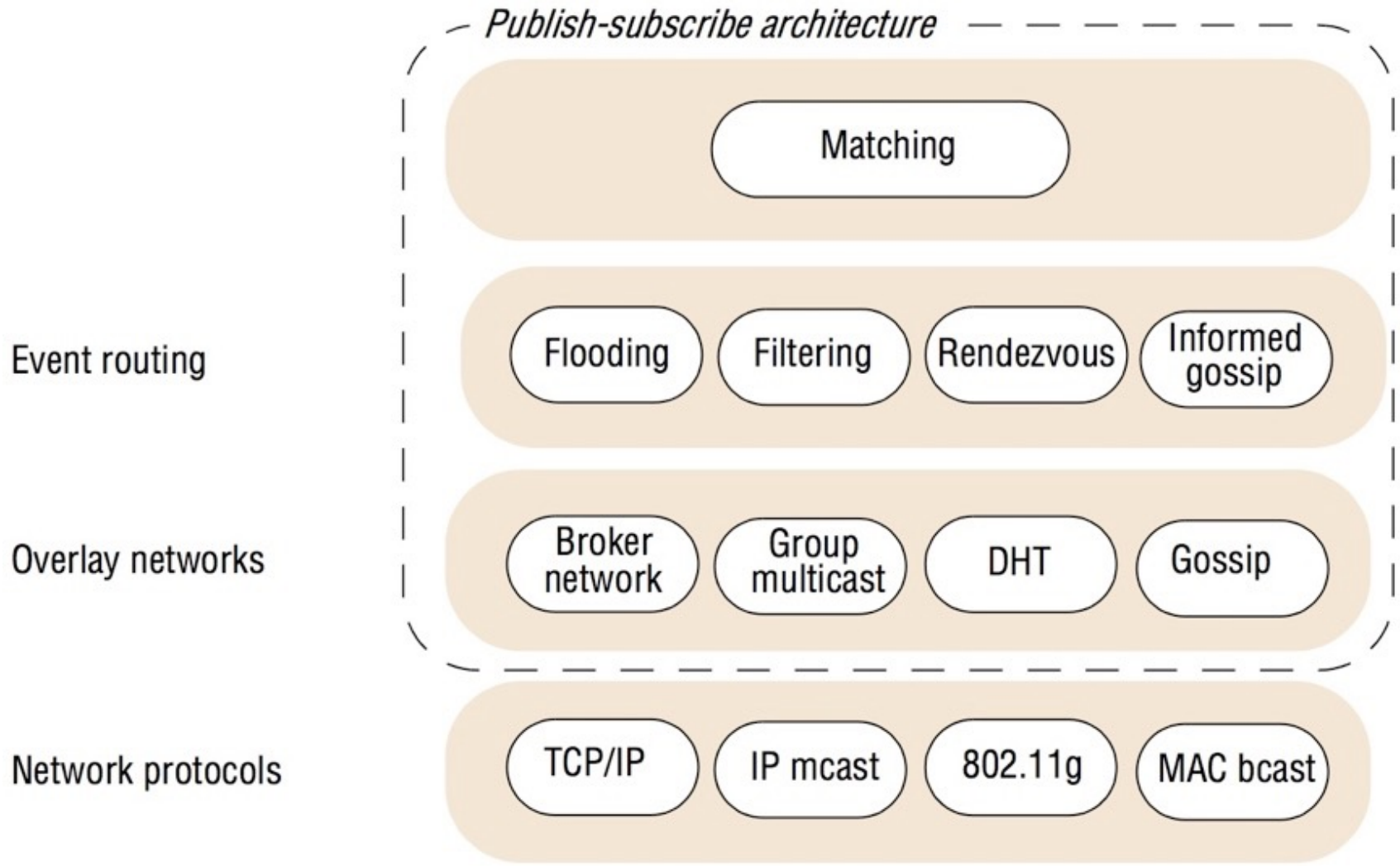
Implementation issues

- The task of a publish-subscribe system is to ensure that events are delivered efficiently to all subscribers that have filters defined that match the event
- Subscription filter model:
  - Channel-based
  - Topic-based
  - Content-based
  - Type-based
- Centralized versus distributed implementations

A network of brokers (Centralized)



The architecture of publish-subscribe systems



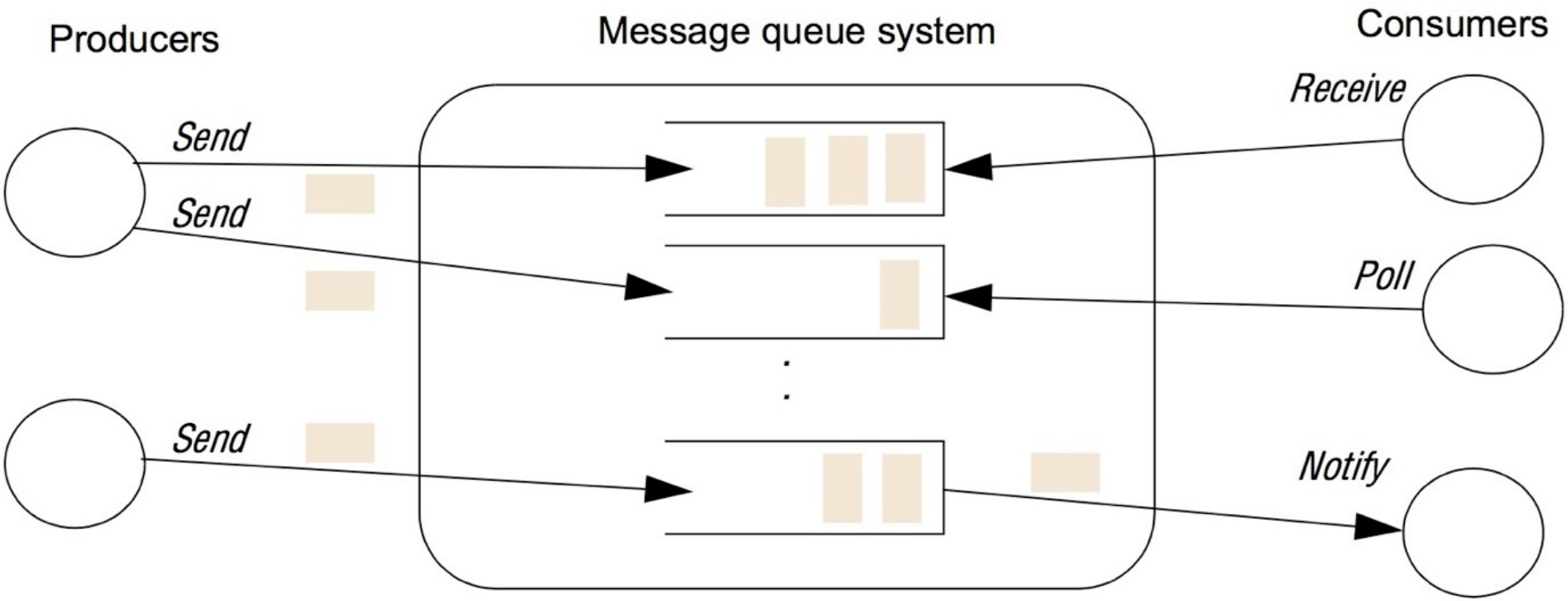
Example publish-subscribe systems

System (and further reading)	*Subscription model*	*Distribution model*	*Event routing*
CORBA Event Service (Chapter 8)	Channel-based	Centralized	-
TIB Rendezvouz [Oki *et al.* 1993]	Topic-based	Distributed	Ffiltering
Scribe [Castro *et al.* 2002b]	Topic-based	Peer-to-peer (DHT)	Rendezvous
TERA [Baldoni *et al.* 2007]	Topic-based	Peer-to-peer	Informed gossip
Siena [Carzaniga *et al.* 2001]	Content-based	Distributed	Filtering
Gryphon [www.research.ibm.com]	Content-based	Distributed	Filtering
Hermes [Pietzuch and Bacon 2002]	Topic- and content-based	Distributed	Rendezvous and filtering
MEDYM [Cao and Singh 2005]	Content-based	Distributed	Flooding
Meghdoot [Gupta *et al.* 2004]	Content-based	Peer-to-peer	Rendezvous
Structure-less CBR [Baldoni *et al.* 2005]	Content-based	Peer-to-peer	Informed gossip

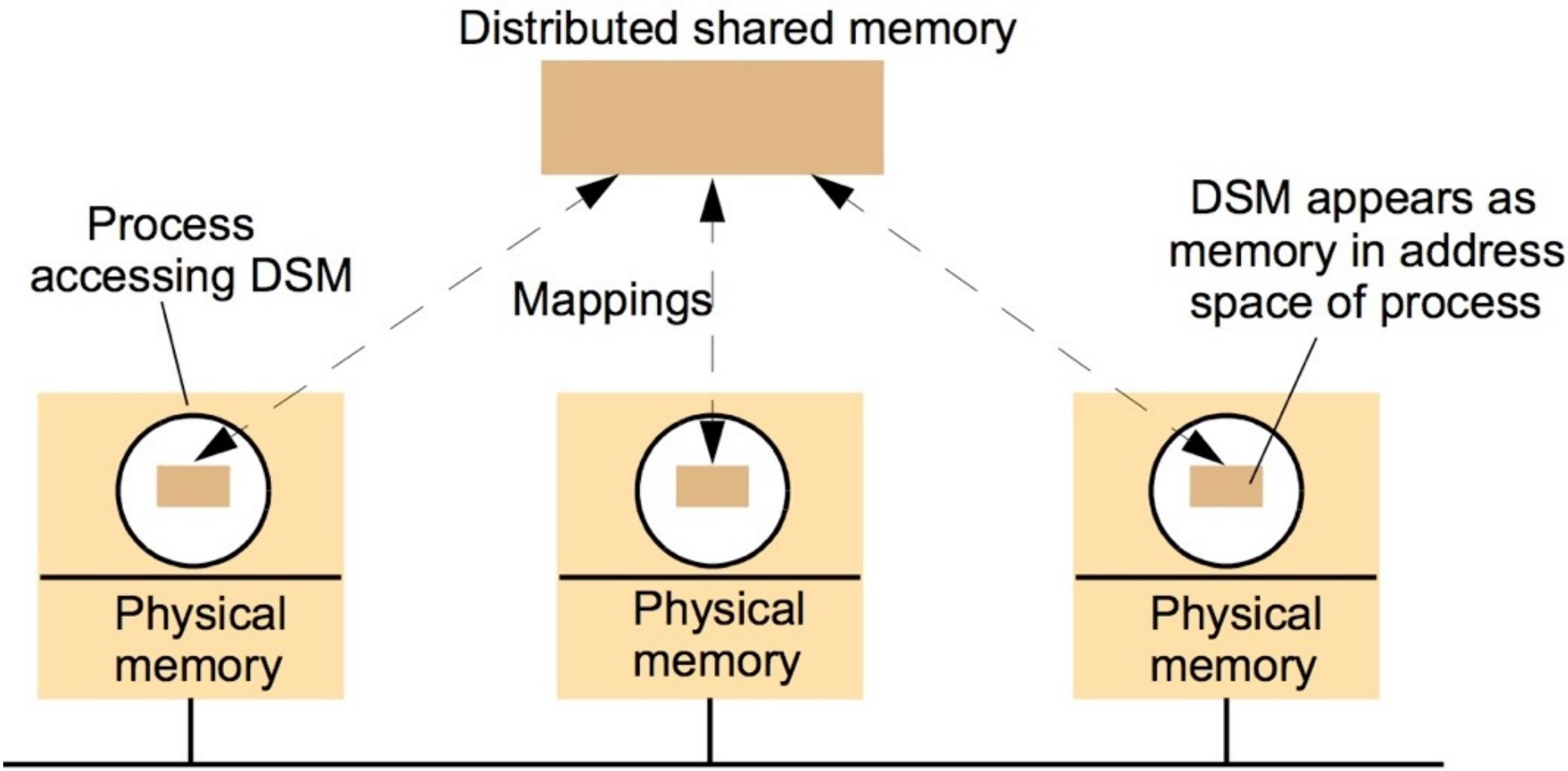
Message queues

- Message queues provide a point-to-point service using the concept of a message queue as an indirection
- The main use of such products is to achieve Enterprise Application Integration
- They are also extensively used as the basis for commercial transaction processing systems

The message queue paradigm



The distributed shared memory abstraction



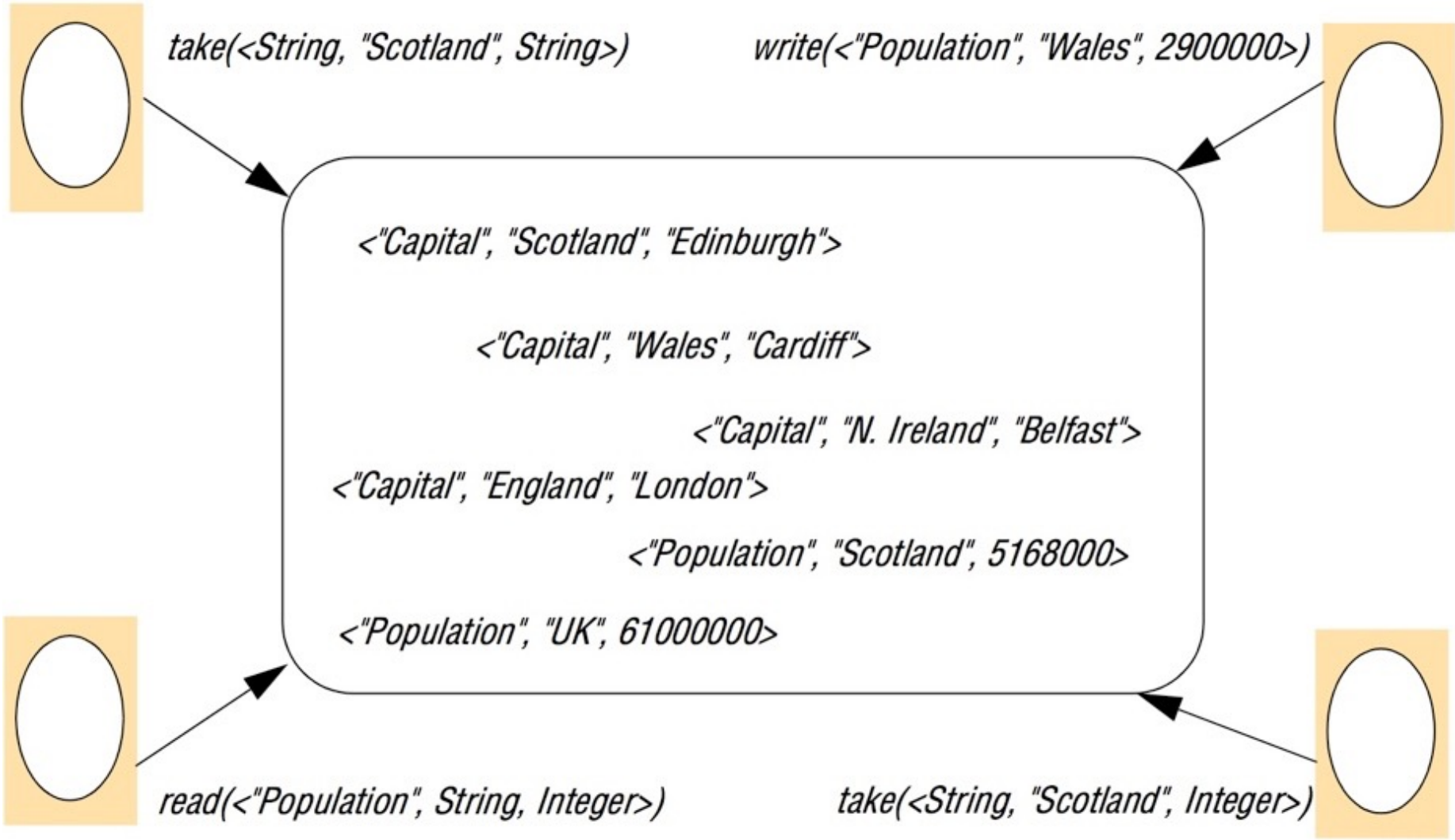
Shared memory approaches

- Distributed shared memory (DSM) is an abstraction used for sharing data between computers that do not share physical memory
- Processes access DSM by reads and updates to what appears to be ordinary memory within their address space
- It is as though the processes access a single shared memory, but in fact the physical memory is distributed
- DSM is primarily a tool for parallel applications or for any distributed application or group of applications in which individual shared data items can be accessed directly
- DSM is in general less appropriate in client-server systems, where clients normally view server-held resources as abstract data and access them by request

Tuple space communication

- processes communicate indirectly by placing tuples in a tuple space, from which other processes can read or remove them
- Tuples consist of a sequence of one or more typed data fields such as <"fred", 1958>, <"sid", 1964> and <4, 9.8, "Yes">
- Operations
  - Write
  - Read
  - Take
- Tuples are immutable

The tuple space abstraction



Summary of indirect communication styles

	Groups	*Publish-subscribe systems*	*Message queues*	*DSM*	*Tuple spaces*
Space-uncoupled	Yes	Yes	Yes	Yes	Yes
Time-uncoupled	Possible	Possible	Yes	Yes	Yes
Style of service	Communication-based	Communication-based	Communication-based	State-based	State-based
Communication pattern	1-to-many	1-to-many	1-to-1	1-to-many	1-1 or 1-to-many
Main intent	Reliable distributed computing	Information dissemination or EAI; mobile and ubiquitous systems	Information dissemination or EAI; commercial transaction processing	Parallel and distributed computation	Parallel and distributed computation; mobile and ubiquitous systems
Scalability	Limited	Possible	Possible	Limited	Limited
Associative	No	Content-based publish-subscribe only	No	No	Yes