Chapter 4: Communication

Fundamentals

Introduction

- In a distributed system, processes run on different machines.
- Processes can only exchange information through message passing.
  - harder to program than shared memory communication
- Successful distributed systems depend on communication models that hide or simplify message passing

Middleware Protocols

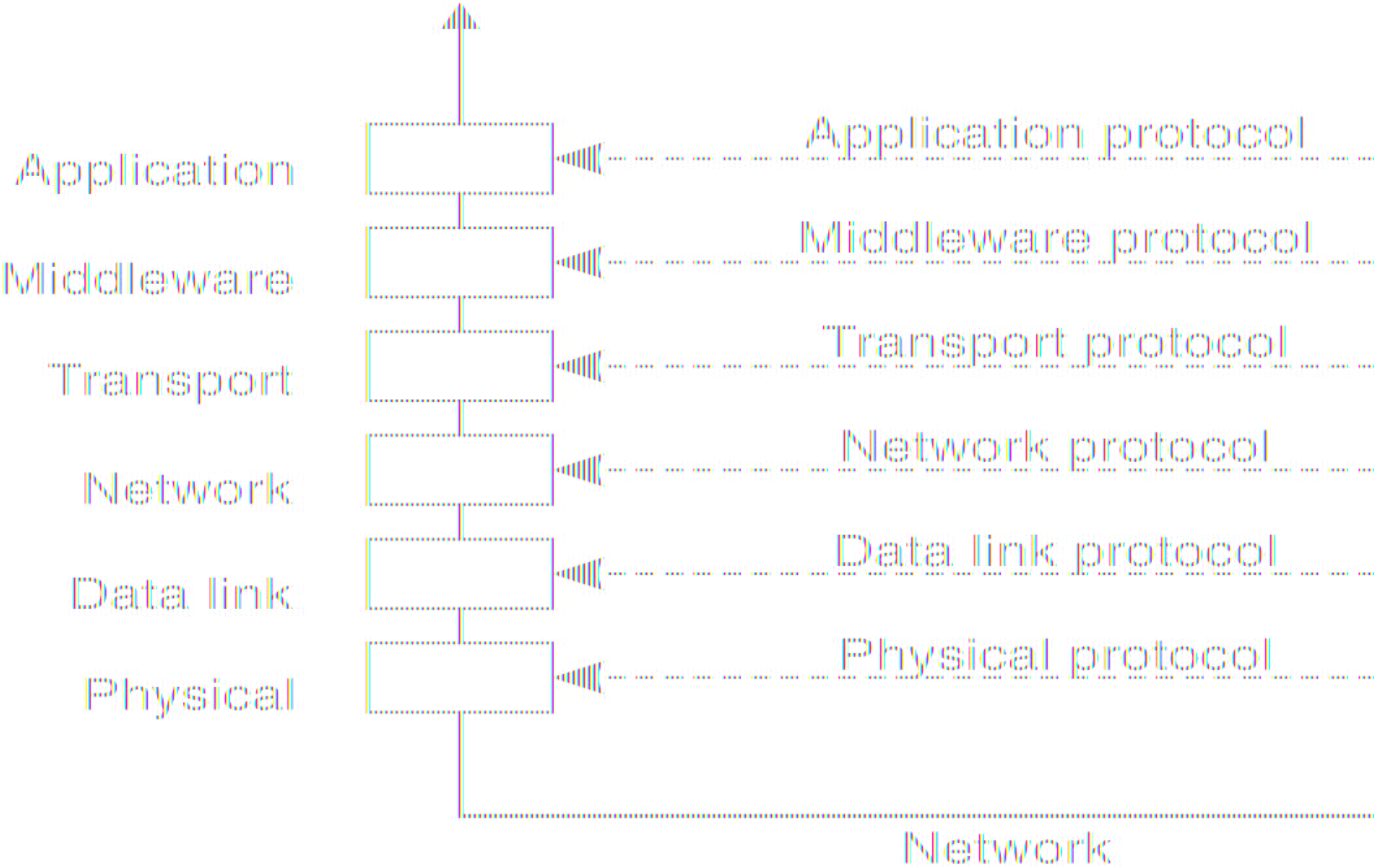


Figure 4-3. An adapted reference model for networked communication.

Protocols to Support Services

- Authentication protocols, to prove identity
- Authorization protocols, to grant resource access to authorized users
- Distributed commit protocols, used to allow a group of processes to decided to commit or abort a transaction (ensure atomicity) or in fault tolerant applications.
- Locking protocols to ensure mutual exclusion on a shared resource in a distributed environment.

Middleware Protocols to Support Communication

- Protocols for remote procedure call (RPC) or remote method invocation (RMI)
- Protocols to support message-oriented services
- Protocols to support streaming real-time data, as for multimedia applications
- Protocols to support reliable multicast service across a wide-area network

These protocols would be built on top of low-level message passing, as supported by the transport layer.

Messages

- Transport layer message passing consists of two types of primitives: send and receive
  - May be implemented in the OS or through add-on libraries
- Messages are composed in user space and sent via a send() primitive.
- When processes are expecting a message they execute a receive() primitive.
  - Receives are often blocking

Types of Communication

- Persistent versus transient
- Synchronous versus asynchronous
- Discrete versus streaming

Persistent versus Transient Communication

- **Persistent**: messages are held by the middleware comm. service until they can be delivered. (Think email)
  - Sender can terminate after executing send
  - Receiver will get message next time it runs
- **Transient**: Messages exist only while the sender and receiver are running
  - Communication errors or inactive receiver cause the message to be discarded.
  - Transport-level communication is transient

Asynchronous v Synchronous Communication

- **Asynchronous**: (non-blocking) sender resumes execution as soon as the message is passed to the communication/middleware software
  - Message is buffered temporarily by the middleware until sent/received
- **Synchronous**: sender is blocked until
  - The OS or middleware notifies acceptance of the message, *or*
  - The message has been delivered to the receiver, *or*
  - The receiver processes it & returns a response. (Also called a rendezvous) –this is what we've been calling synchronous up until now.

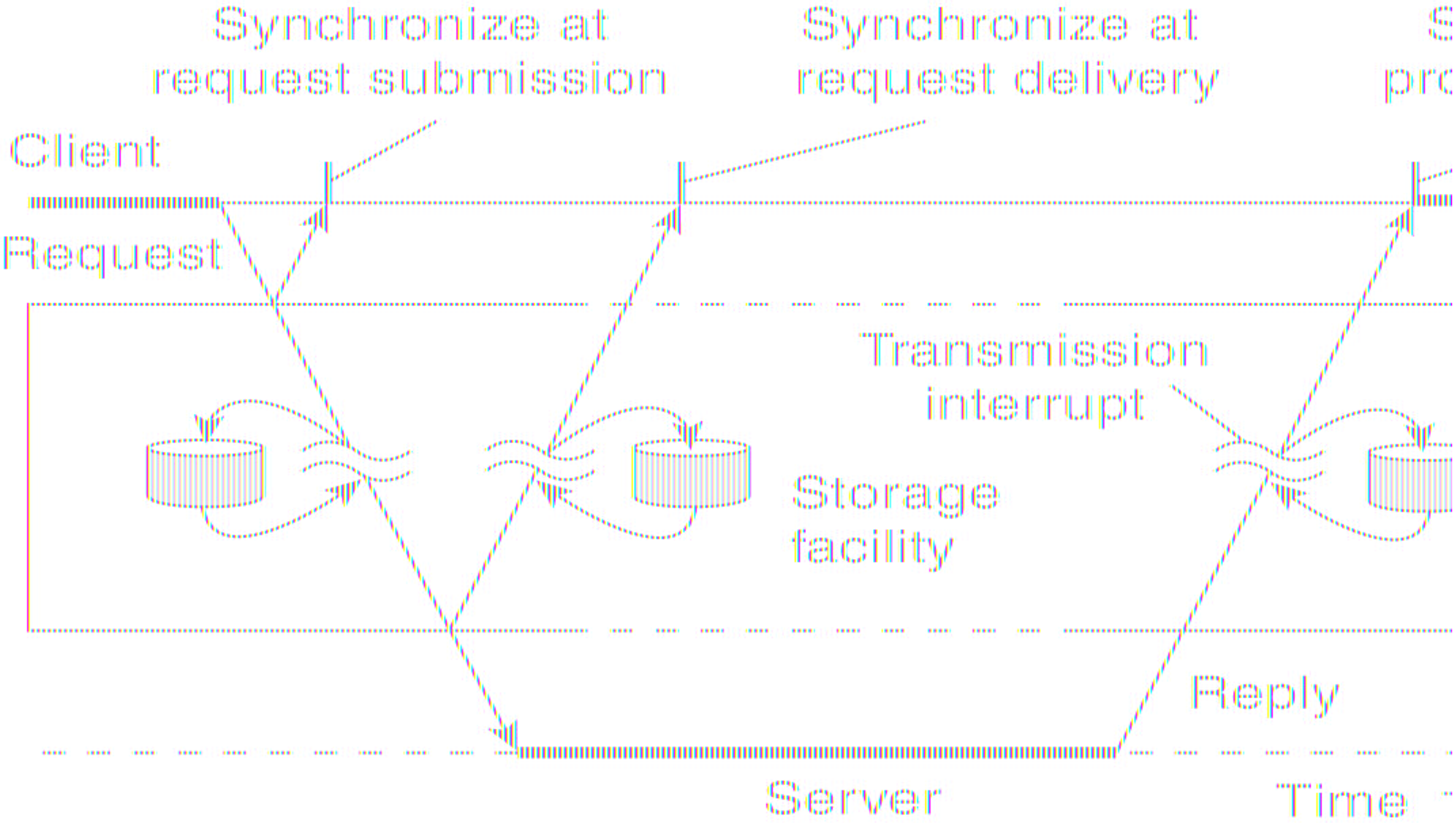


Figure. Viewing middleware as an intermediate (distributed) service in application-level communication.

Middleware Communication Techniques

- Remote Procedure Call
- Remote Method Invocation
- Message-Oriented Communication
- Stream-Oriented Communication
- Multicast Communication

Remote Procedure Calls

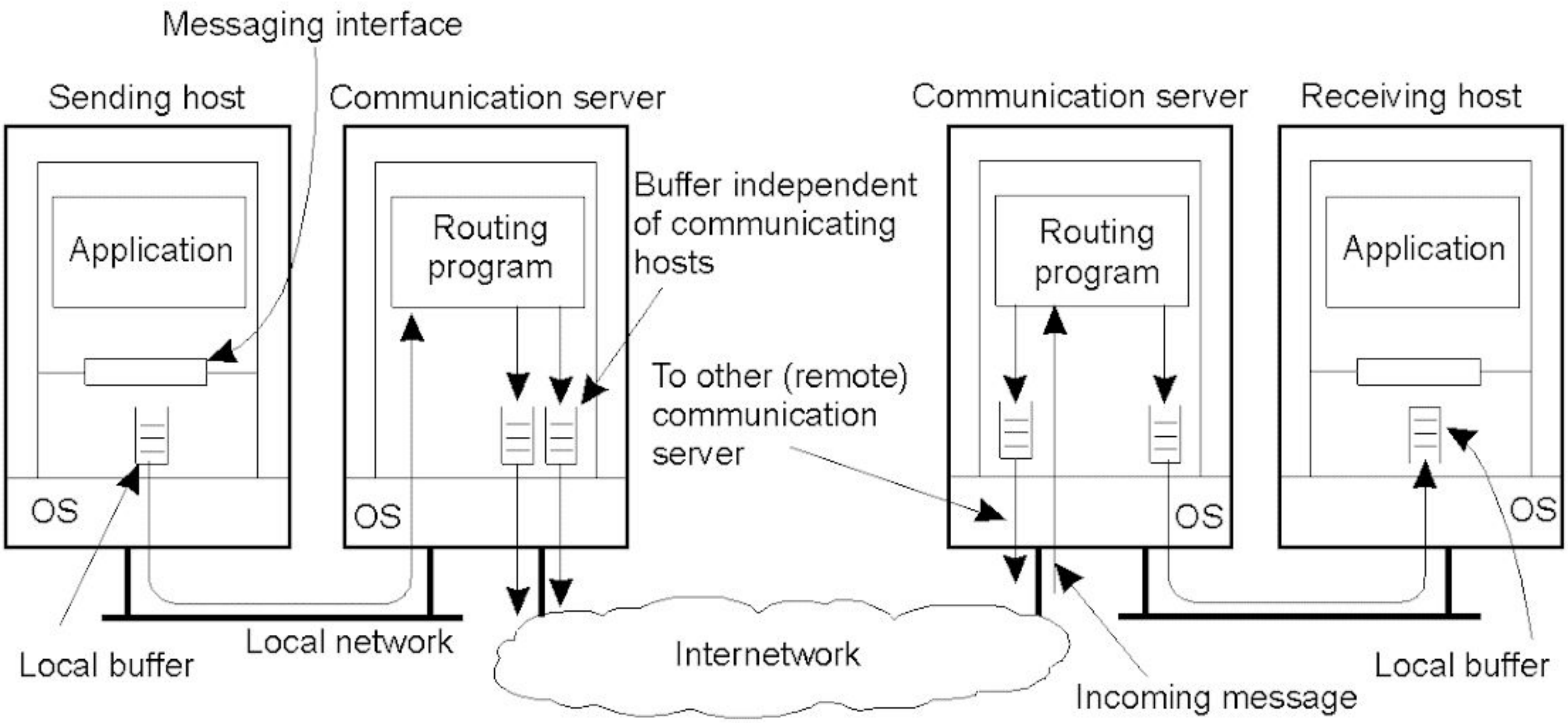
- Basic operation of RPC parallels same-process procedure calling
- Caller process executes the remote call and is suspended until called function completes and results are returned.
- Parameters are passed to the machine where the procedure will execute.
- When procedure completes, results are passed back to the caller and the client process resumes execution at that time.

Message Oriented Communication

- RPC and RMI support access transparency, but aren’t always appropriate
- Message-oriented communication is more flexible
- Built on transport layer protocols.
- Standardized interfaces to the transport layer include **sockets** (Berkeley UNIX) and XTI (X/Open Transport Interface), formerly known as TLI (AT&T model)

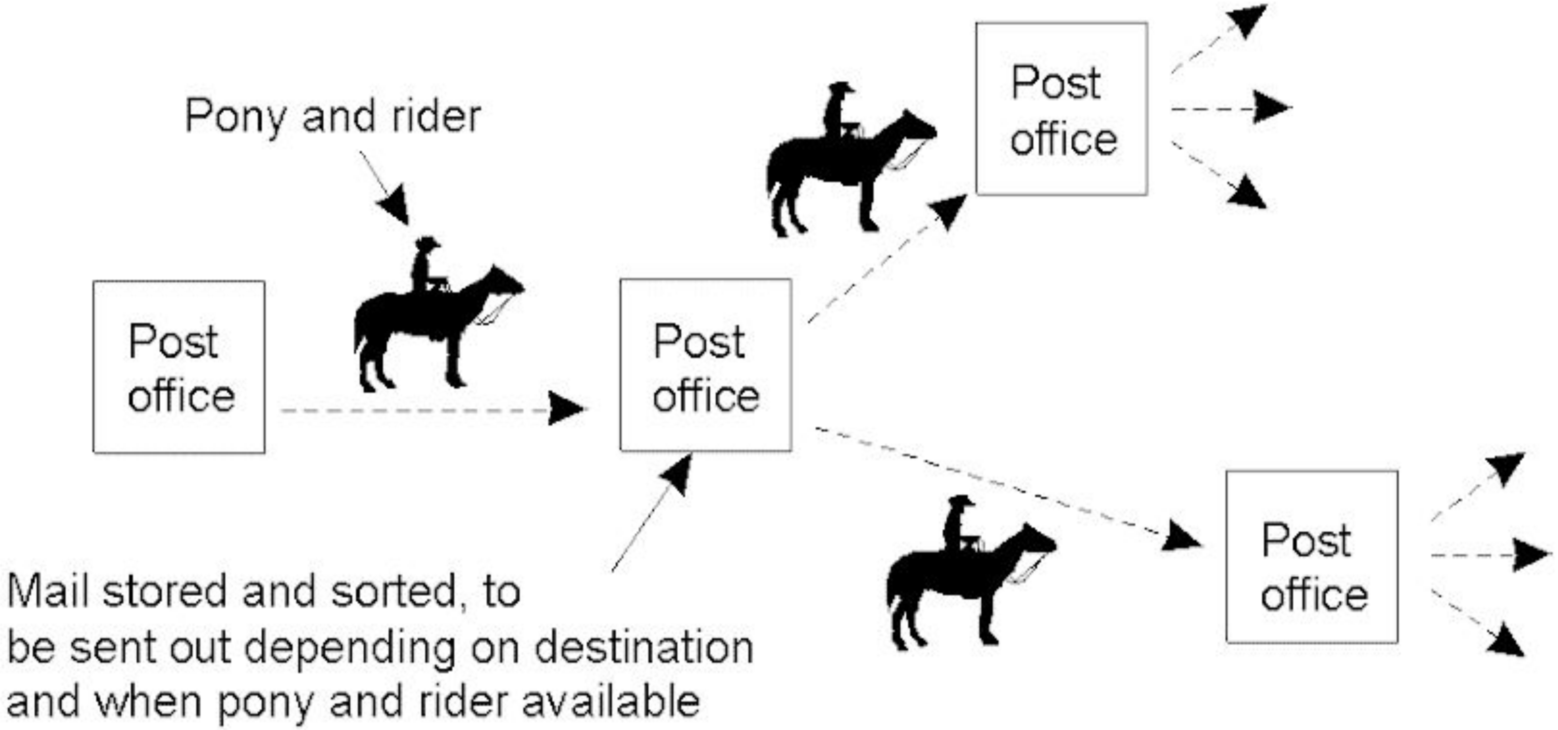
Persistence and Synchronicity in Communication

- **General communication system connected through a network**
  - Each host is connected to a single communication server.
  - Hosts and communication servers can have buffers.



Persistence and Synchronicity in Communication(2)

- **Persistent communication**
  - An example of persistent communication – Letter back in the days of Pony Express


Pony and rider
Post office
Post office
Post office
Post office
Mail stored and sorted, to be sent out depending on destination and when pony and rider available

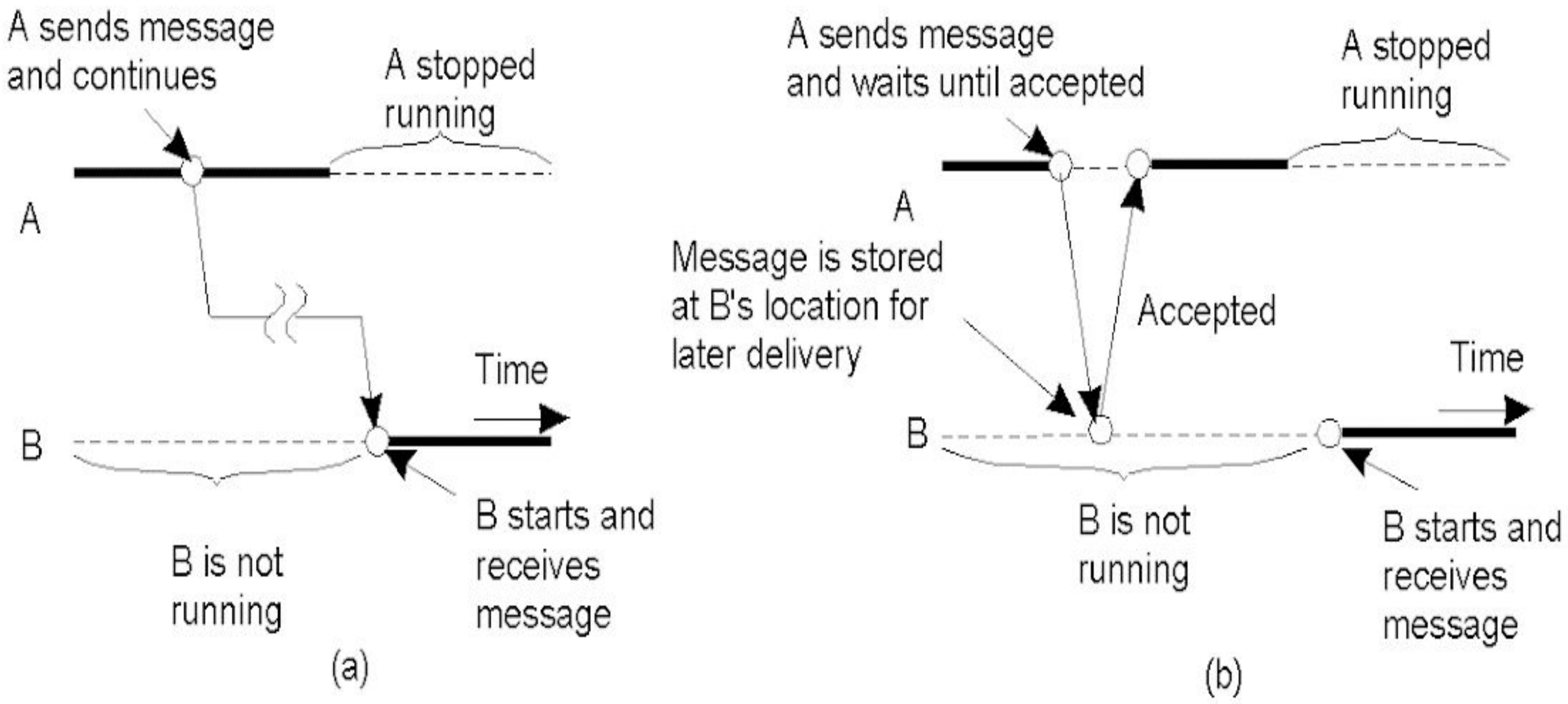
Persistence and Synchronicity in Communication(3)

- **Persistent vs. Transient**
  - Persistent messages are stored as long as necessary by the communication system (e.g., e-mail)
  - Transient messages are discarded when they cannot be delivered (e.g., TCP/IP)

- **Synchronous vs. Asynchronous**
  - Asynchronous implies sender proceeds as soon as it sends the message no blocking
  - Synchronous implies sender blocks till the receiving host buffers the message

Persistence and Synchronicity in Communication(4)

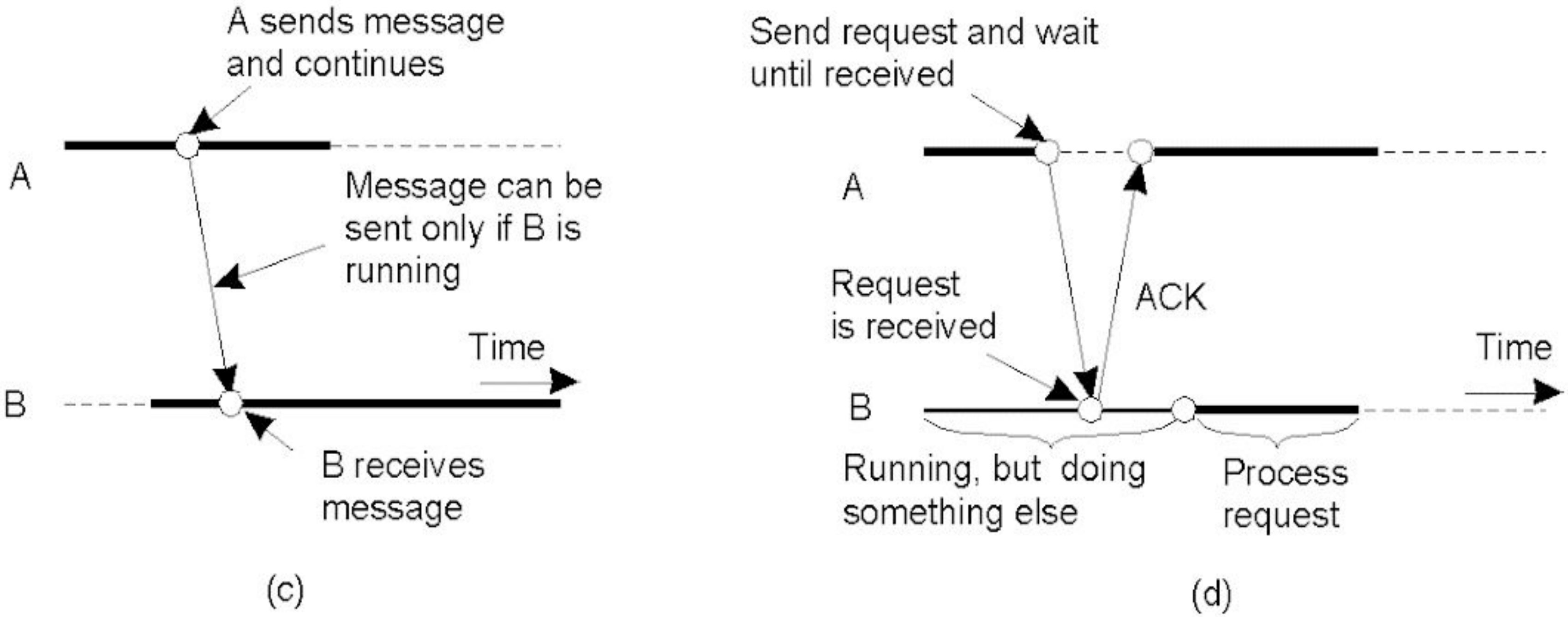
- Persistent asynchronous/synchronous


A sends message and continues
A stopped running
A
Time
B
B is not running
B starts and receives message
(a)
A sends message and waits until accepted
A stopped running
A
Message is stored at B's location for later delivery
Accepted
Time
B
B is not running
B starts and receives message
(b)

(a) Persistent asynchronous communication / (b) Persistent synchronous communication

Persistence and Synchronicity in Communication(5)

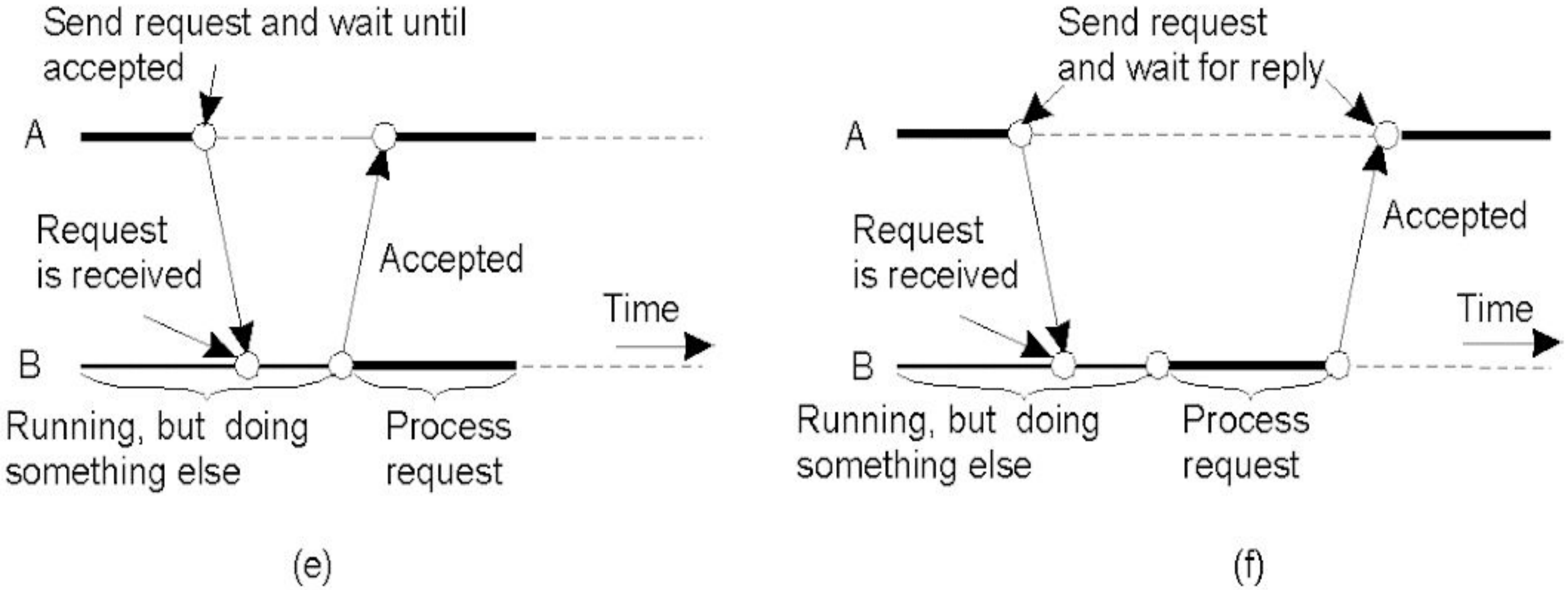
- Transient asynchronous/Receipt-based transient synchronous communication


A sends message and continues
A
Message can be sent only if B is running
Time
B
B receives message
(c)
Send request and wait until received
A
Request is received
ACK
Time
B
Running, but doing something else
Process request
(d)

(c) Transient asynchronous communication / (d) receipt-based transient synchronous communication

Persistence and Synchronicity in Communication(6)

- Other transient synchronous communications



Send request and wait until accepted
A
Request is received
Accepted
Time
B
Running, but doing something else
Process request
(e)
Send request and wait for reply
A
Accepted
Request is received
Time
B
Running, but doing something else
Process request
(f)

(e) Delivery-based transient synchronous communication at message delivery
(f) Response-based transient synchronous communication

Message-Oriented Transient Communication

- Message-Oriented Model
  - Many distributed systems and applications are built on top of the simple message-oriented model
  - These models are offered by Transport Layer
  - Message-oriented models
    - Berkeley Sockets: Socket interface as introduced in Berkeley UNIX
    - The Message-Passing Interface(MPI): designed for parallel applications and as such in tailored to transient communication

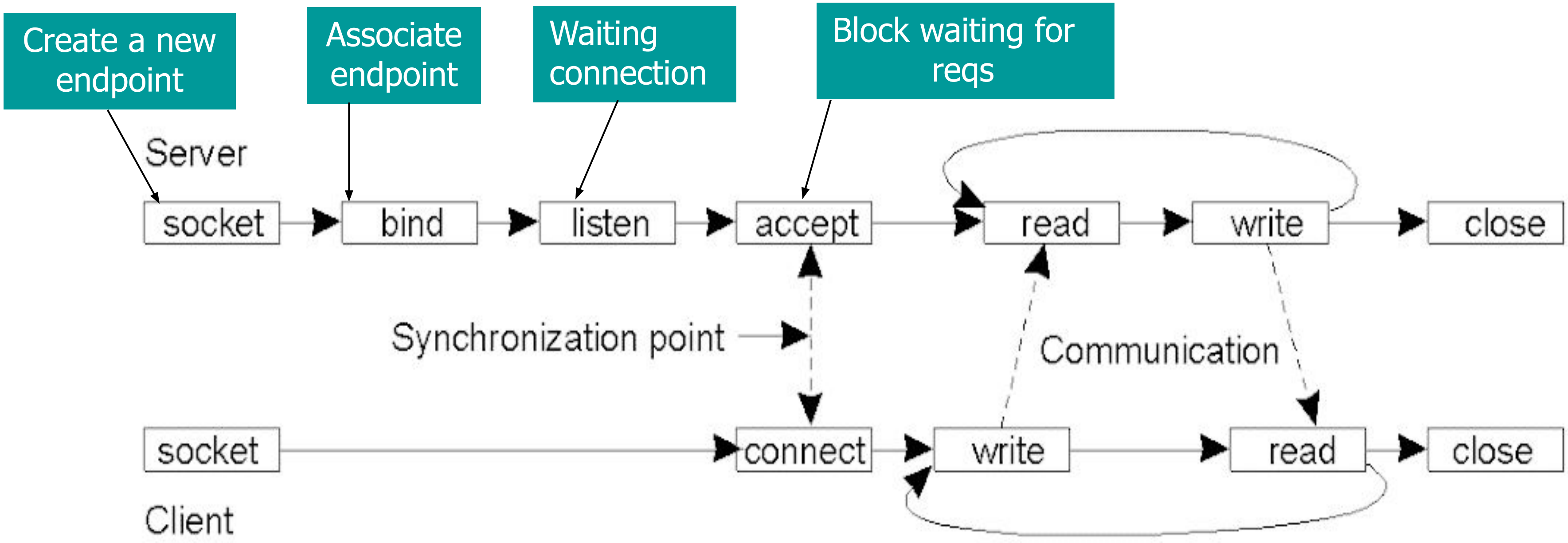
Message-Oriented Transient Communication

- **Berkeley Sockets(1)**
  - Meaning of Socket: a communication endpoint to which an application can write data (be sent to network) and read incoming data
  - The socket primitives for TCP/IP

Primitive	**Meaning**
Socket	Create a new communication endpoint
Bind	Attach a local address to a socket
Listen	Announce willingness to accept connections
Accept	Block caller until a connection request arrives
Connect	Actively attempt to establish a connection
Send	Send some data over the connection
Receive	Receive some data over the connection
Close	Release the connection

Message-Oriented Transient Communication

- **Berkeley Sockets(2)**
  - Connection-oriented communication pattern using sockets
  - Sockets considered insufficient because:
    - Support only send and receive primitives
    - Designed for communication using general-purpose protocol such as TCP/IP



Create a new endpoint

Associate endpoint

Waiting connection

Block waiting for reqs

Message-Orient Transient Communication(4)

- The Message-Passing Interface(MPI)(1)
  - Designed for multiprocessor machines and high-performance parallel programming
  - Provides a **high-level of abstraction** than sockets
  - Support diverse forms of **buffering and synchronization** (over 100 functions)

Message-Orient Transient Communication(5)

- **The Message-Passing Interface(MPI)**
  - Some of the most intuitive message-passing primitives of MPI

Primitive	**Meaning**
MPI_bsend	Append outgoing message to a local send buffer
MPI_send	Send a message and wait until copied to local or remote buffer
MPI_ssend	Send a message and wait until receipt starts
MPI_sendrecv	Send a message and wait for reply
MPI_isend	Pass reference to outgoing message, and continue
MPI_issend	Pass reference to outgoing message, and wait until receipt starts
MPI_recv	Receive a message; block if there are none
MPI_irecv	Check if there is an incoming message, but do not block

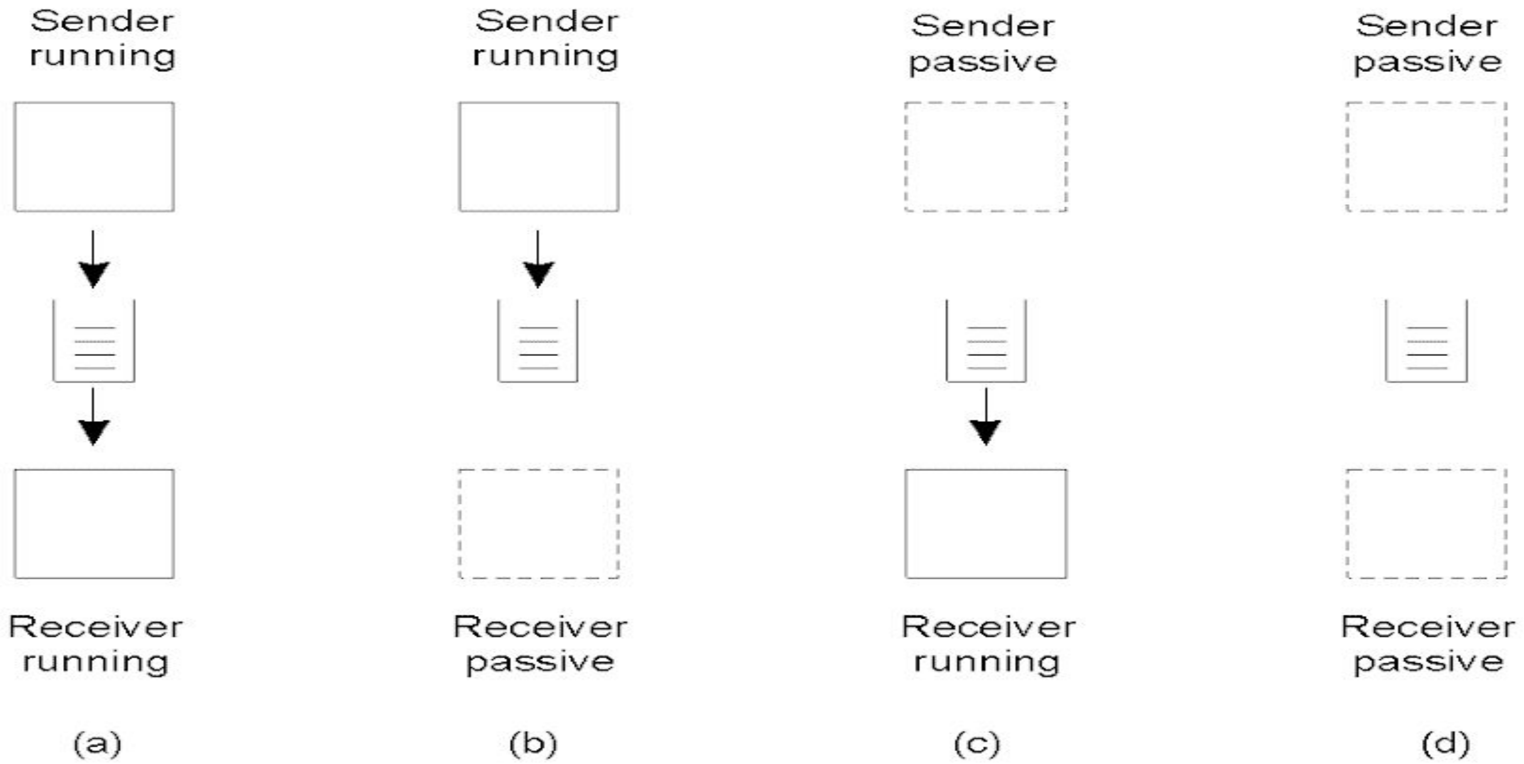
Message-Oriented Persistent Communication

- **Message-Queuing Model(1)**
  - Apps communicate by inserting messages in specific queues
    - Loosely-coupled communication
  - Support for:
    - Persistent asynchronous communication
    - Longer message transfers(e.g., e-mail systems)
  - Basic interface to a queue in a message-queuing system:

Primitive	Meaning
Put	**Append a message to a specified queue**
Get	**Block until the specified queue is nonempty, and remove the first message**
Poll	**Check a specified queue for messages, and remove the first. Never block**
Notify	**Install a handler to be called when a message is put into the specified queue**

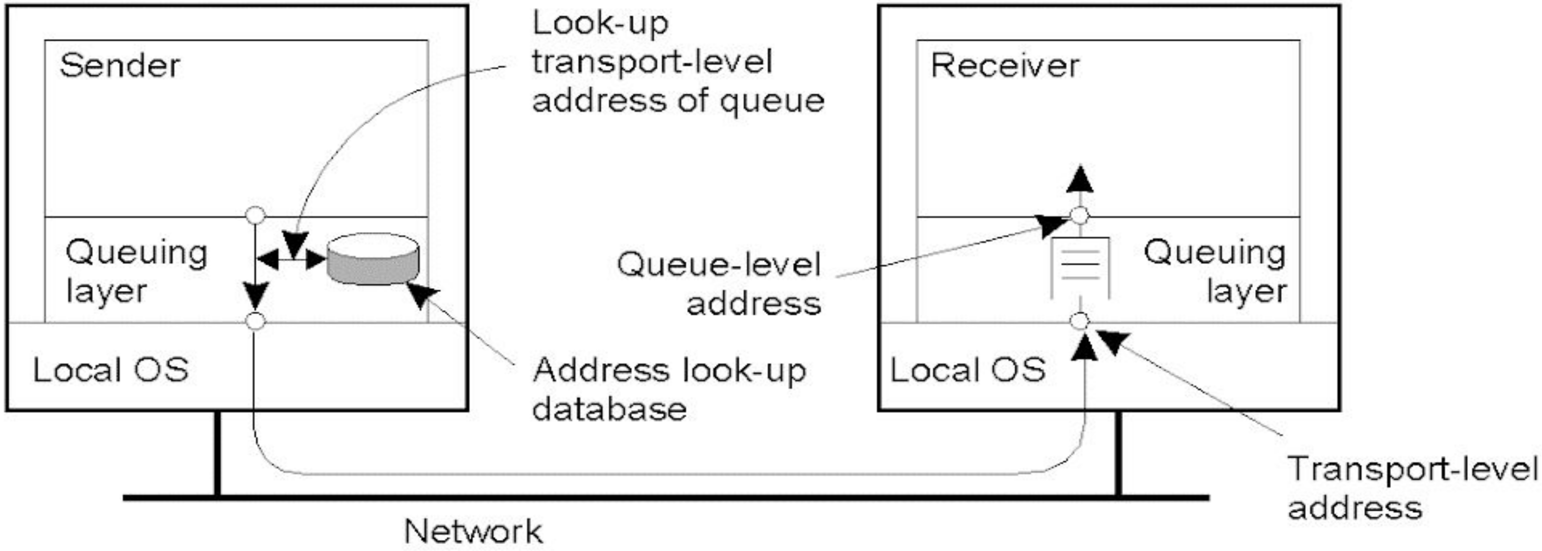
Message-Orient Persistent Communication(2)

- Message-Queuing Model(2)
  - Four combinations for loosely-coupled communication using queues:



Message-Orient Persistent Communication(3)

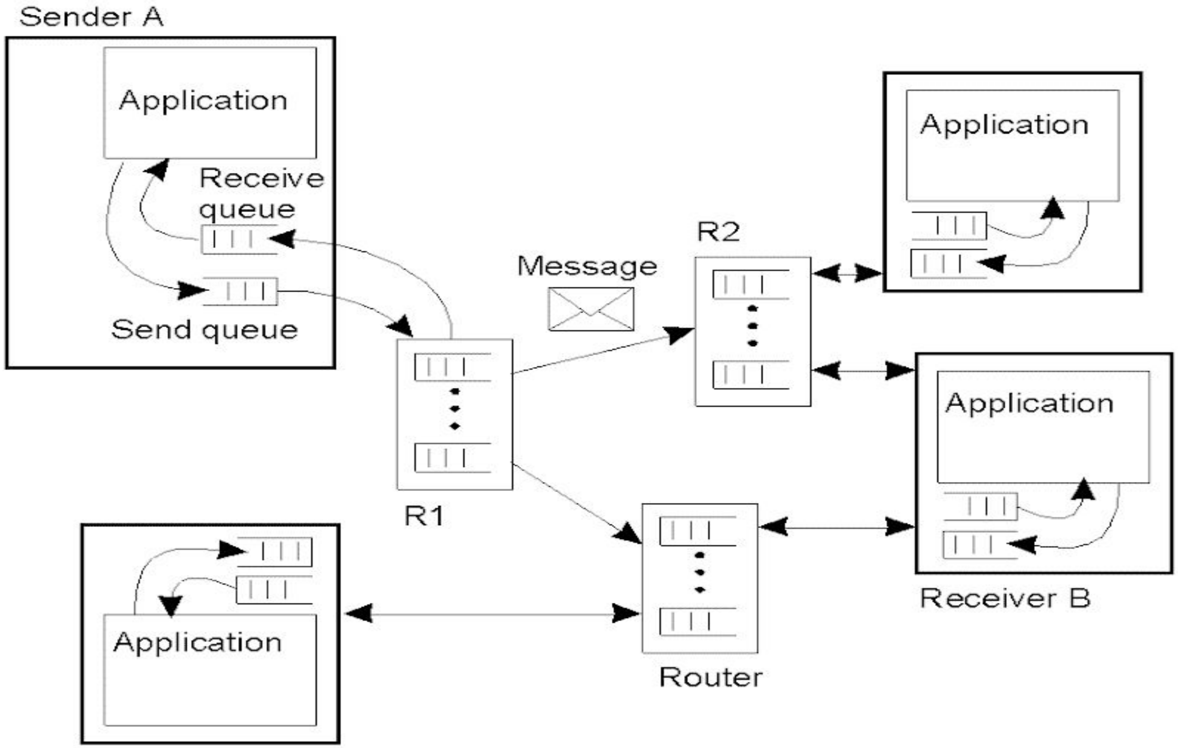
- General architecture of a Message-Queuing System(1)
  - Messages can only be put and received from local queues
  - Ensure transmitting the messages between the source queues and destination queues, meanwhile storing the messages as long as necessary
  - Each queue is maintained by a queue manager



The relationship between queue-level addressing and network-level addressing

Message-Orient Persistent Communication(4)

- General architecture of a Message-Queuing System(2)
  - Queue managers are not only responsible for directly interacting with applications but are also responsible for acting as relays (or routers)



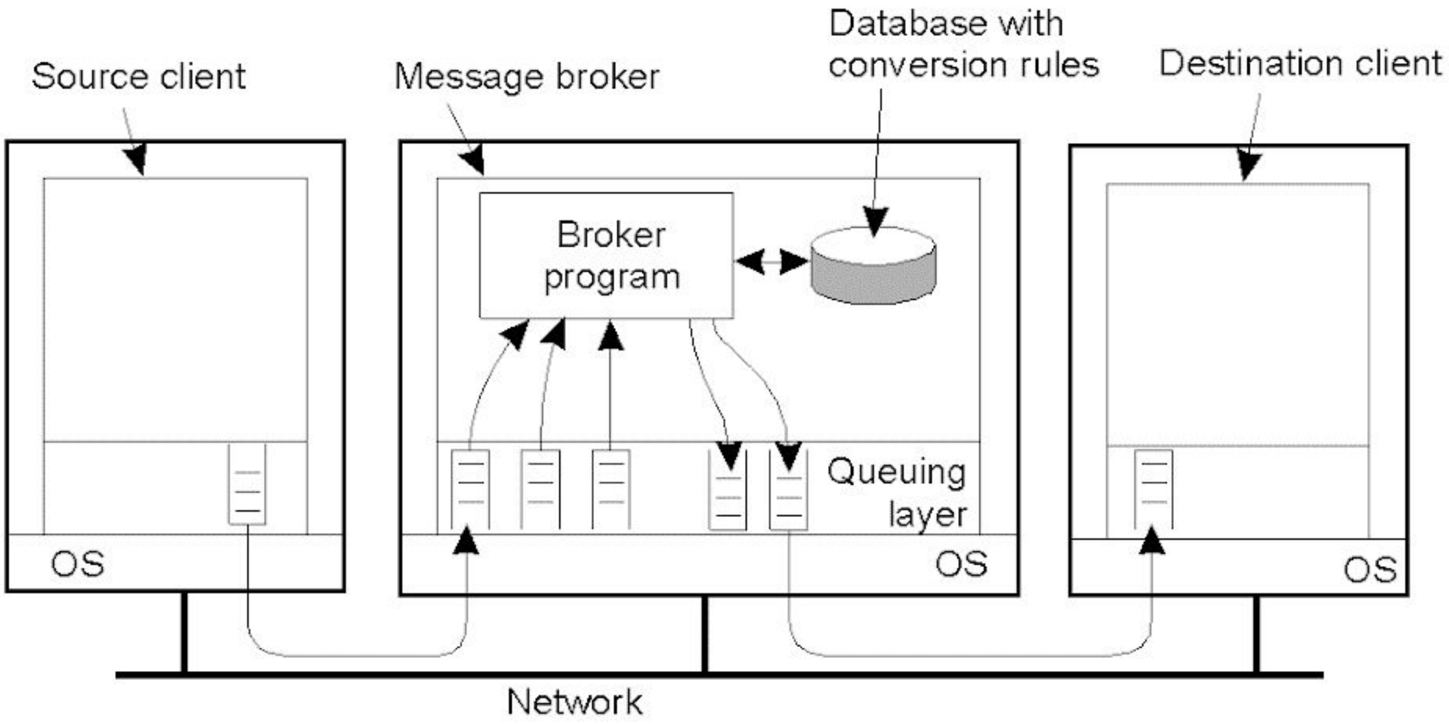
Queue managers form an overlay network, acting as routers

Message-Orient Persistent Communication(5)

- General–purpose of a Message-Queuing System
  - Enable persistent communication between processes
  - Handling access to database
  - In wide range of application, include:
    - Email
    - Groupware
    - Batch processing

Message-Oriented Persistent Communication(6)

- Message Broker



The general organization of a message broker in a message-queuing system

Summary & Conclusion

- **Summary**
  - Two different communication concept ‘Transient vs. Persistent’
    - Persistent messages are stored as long as necessary
    - Transient messages are discarded when they cannot be delivered
  - Message-Oriented Transient Comm.
    - Berkeley socket and MPI
  - Message-Oriented Persistent Comm.
    - Message-Queuing Model and Message Broker

- **Conclusion**
  - Message-Oriented communication solve the blocking problems that may occur in general communication between Server/Client
  - Message-Queuing systems can users(including applications) to do Persistent communication

4.4 Stream-Oriented Communication

- RPC, RMI, message-oriented communication are based on the exchange of discrete messages
  - Timing might affect performance, but not correctness
- In stream-oriented communication the message content must be delivered at a certain rate, as well as correctly.
  - e.g., music or video

Representation

- Different representations for different types of data
  - ASCII or Unicode
  - JPEG or GIF
  - PCM (Pulse Code Modulation)
- **Continuous representation media**: temporal relations between data are significant
- **Discrete representation media:** not so much (text, still pictures, etc.)

Data Streams

- Data stream = sequence of data items
- Can apply to discrete, as well as continuous media
  - e.g. UNIX pipes or TCP/IP connections which are both byte oriented (discrete) streams
- Audio and video require continuous data streams between file and device.

Data Streams

- **Asynchronous transmission mode:** the order is important, and data is transmitted one after the other.
- **Synchronous transmission** mode transmits each data unit with a guaranteed upper limit to the delay for each unit.
- **Isochronous transmission** mode have a maximum and minimum delay.
  - Not too slow, but not too fast either

Streams

- Simple streams have a single data sequence
- Complex streams have several substreams, which must be synchronized with each other; for example a movie with
  - One video stream
  - Two audio streams (for stereo)
  - One stream with subtitles

Questions

- With neat diagrams, explain the synchronicity and persistence in communication (6 types of timing diagrams)
- Significance of queuing system
- **Architecture and working of MOM**
  - Slides 25-30 and study the description of architecture from the book