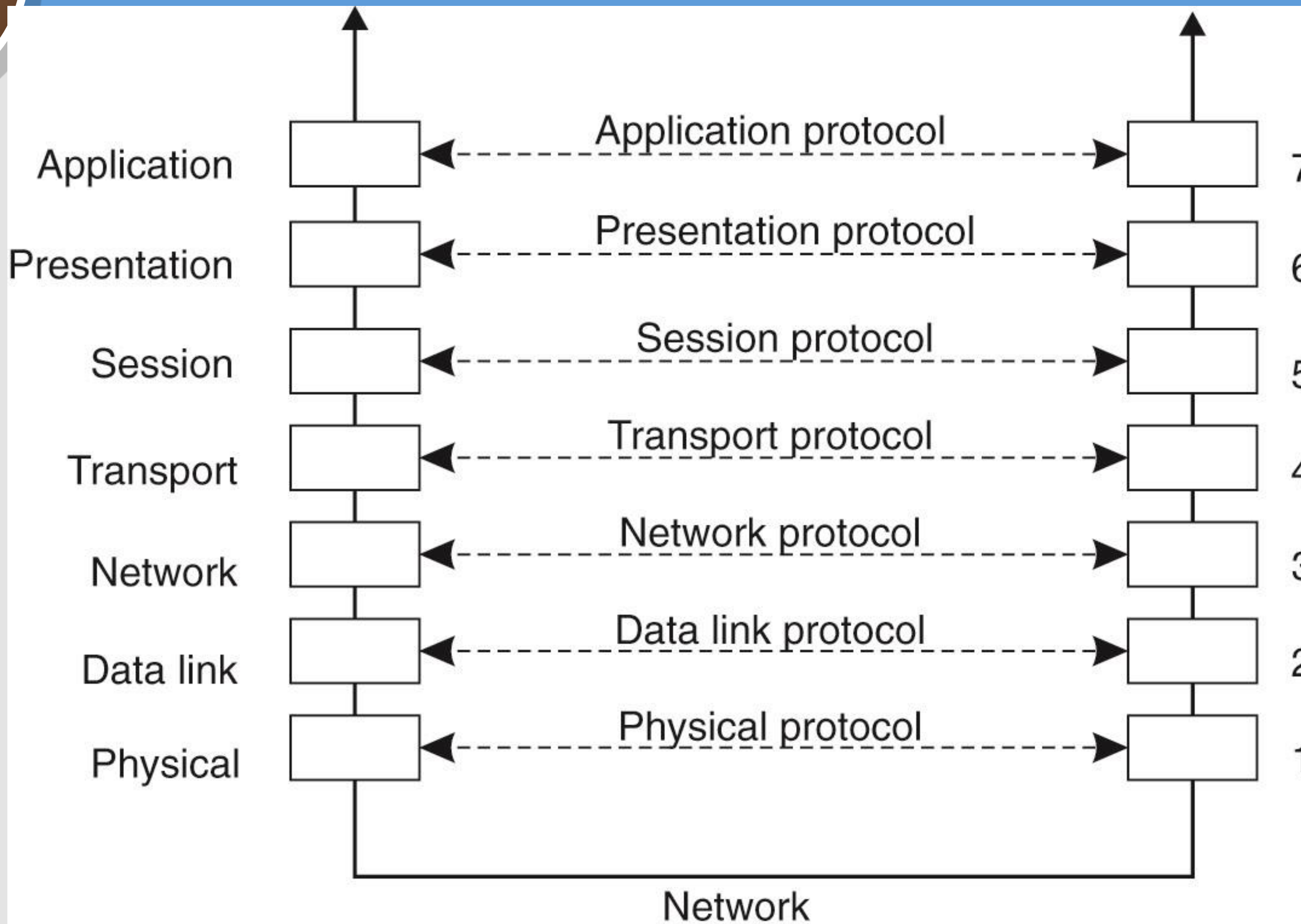
Communication

Munawar, PhD

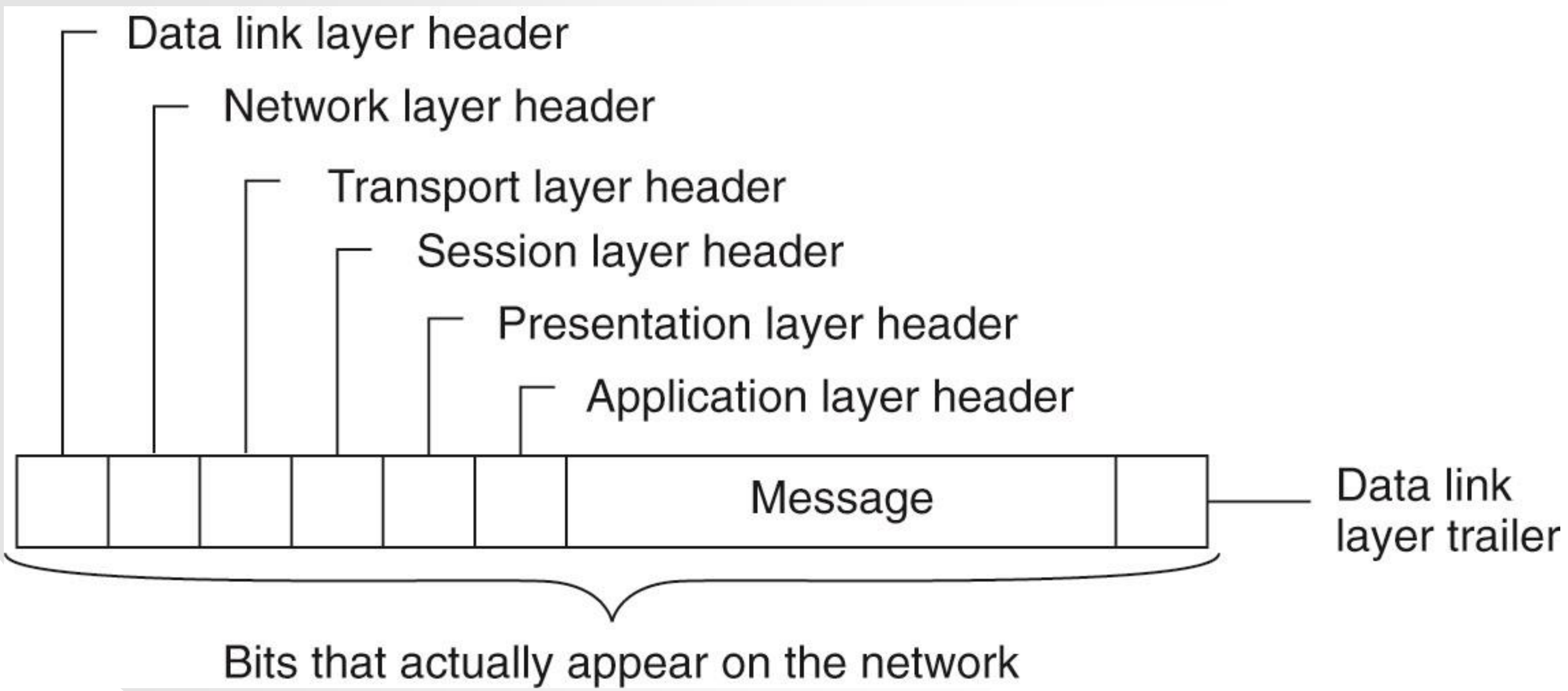
Layered Protocols (1)



- Figure 4-1. Layers, interfaces, and protocols in the OSI model.

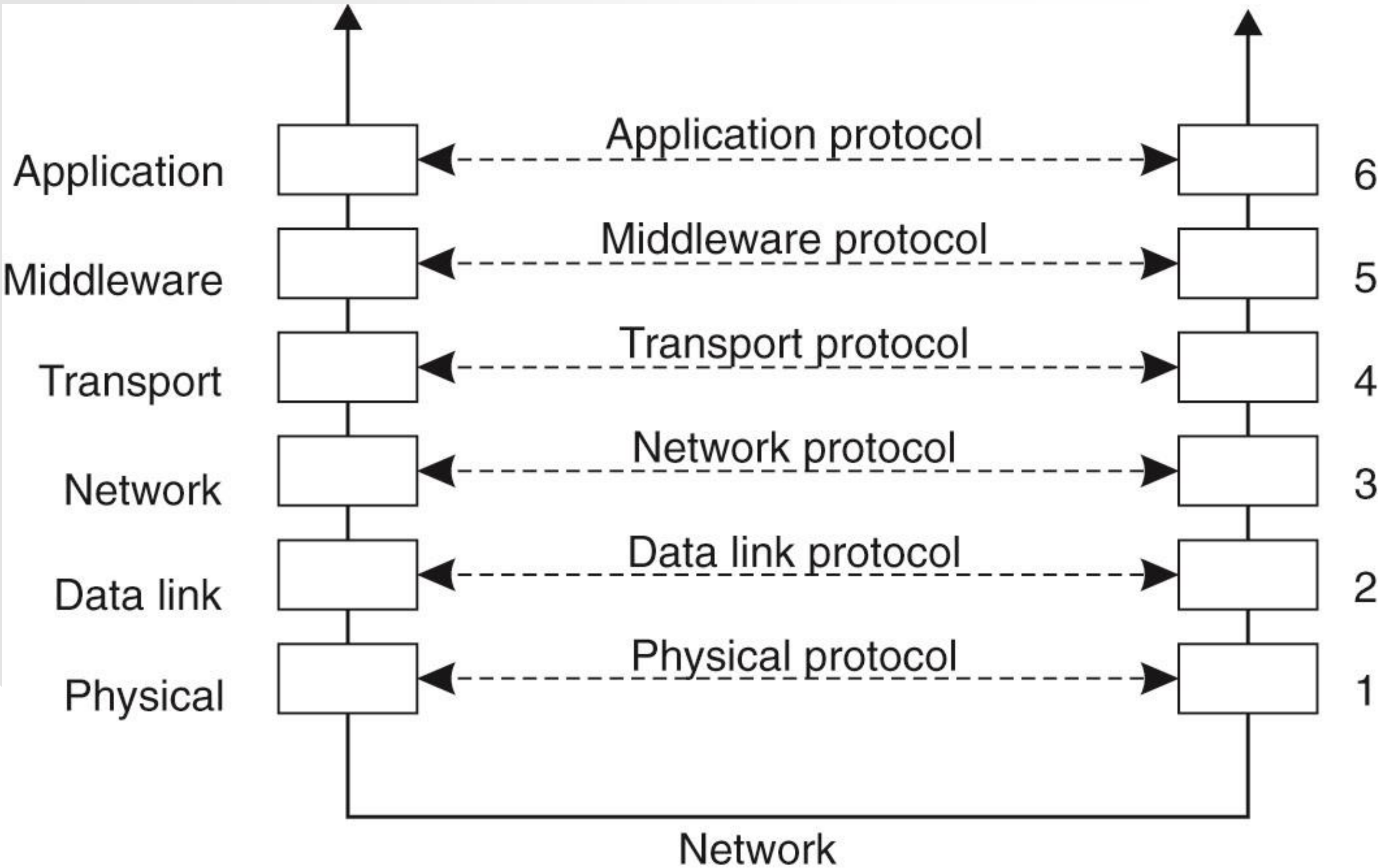
Layered Protocols (2)

- Figure 4-2. A typical message as it appears on the network.



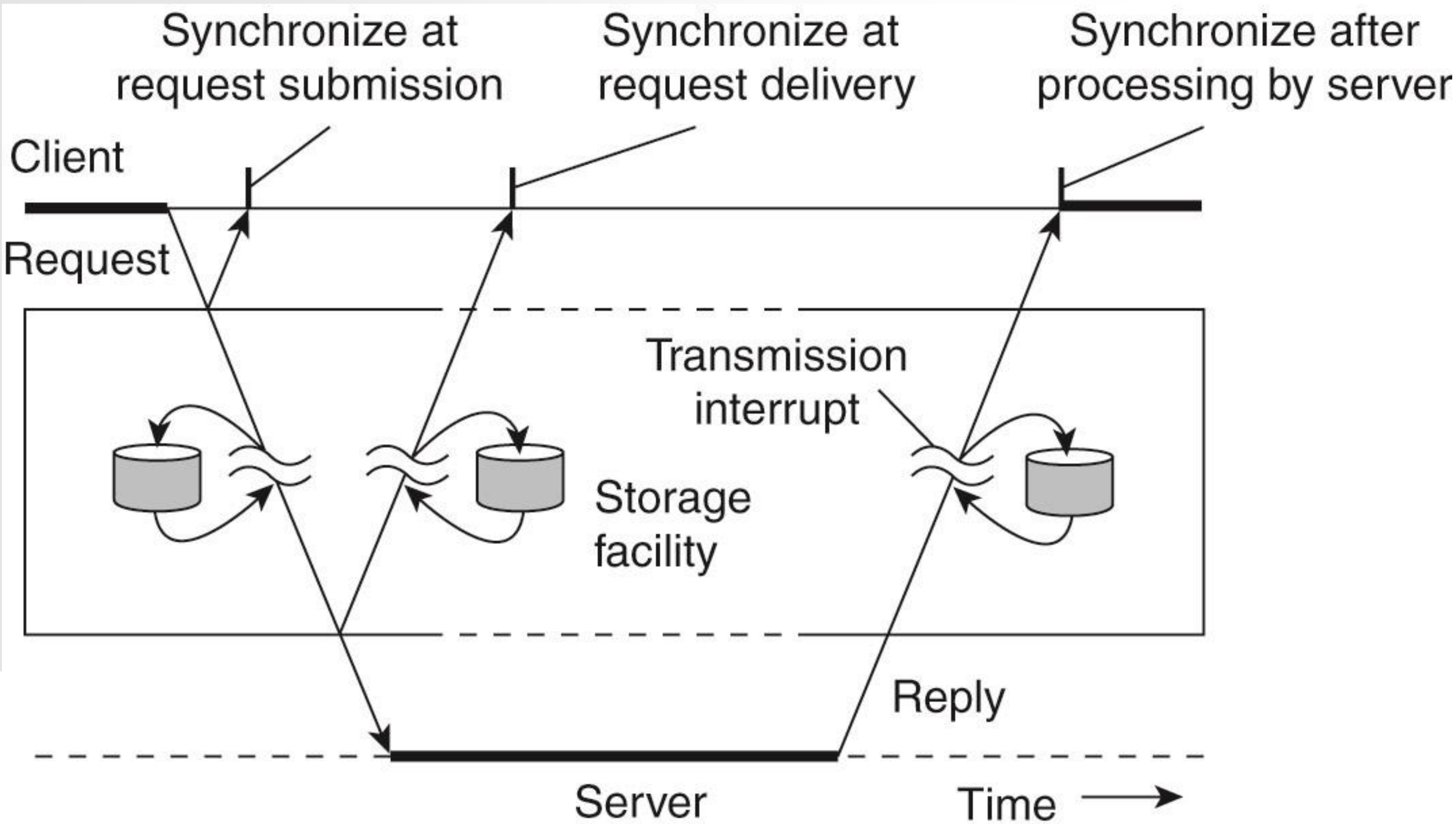
Middleware Protocols

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

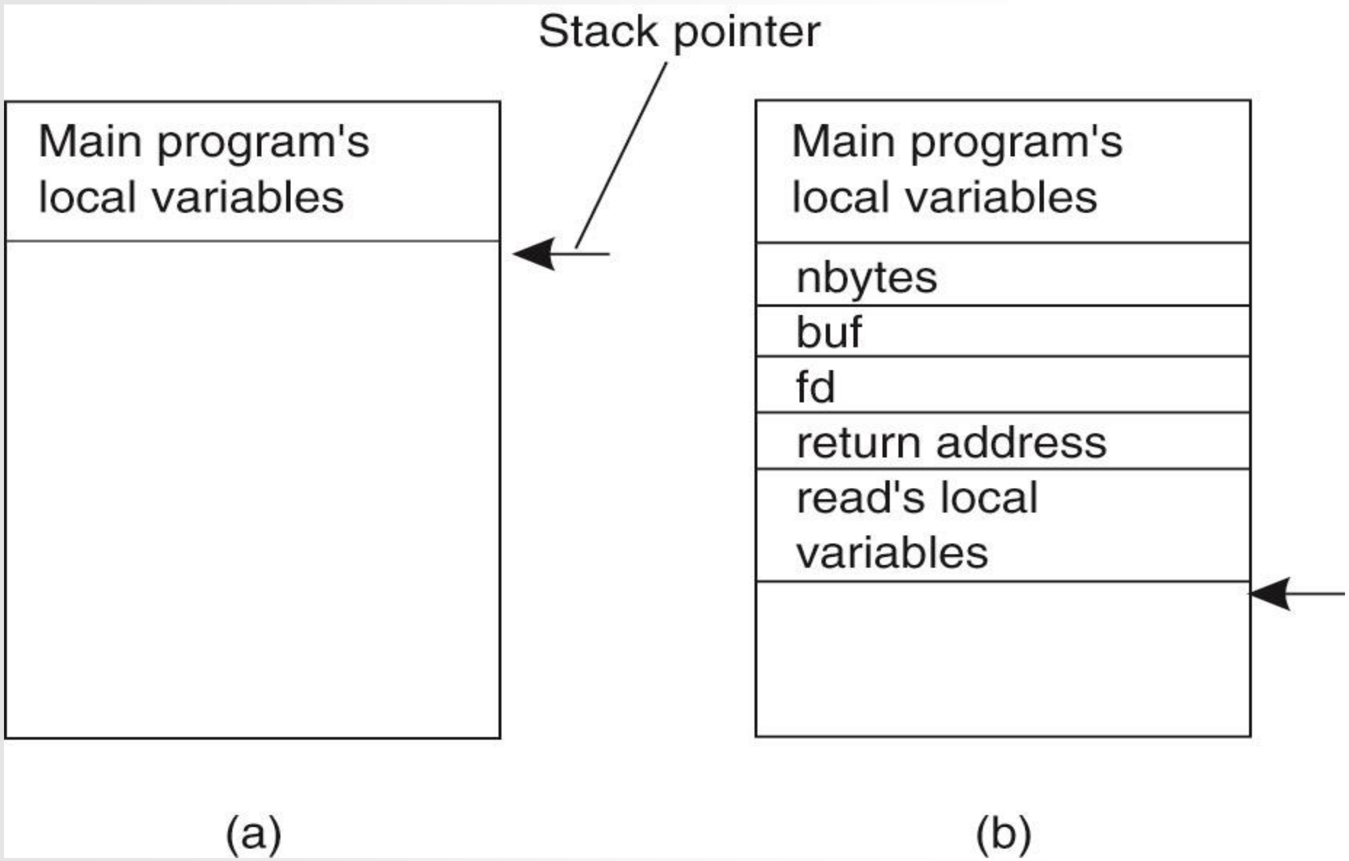


Types of Communication

- Figure 4-4. Viewing middleware as an intermediate (distributed) service in application-level communication.

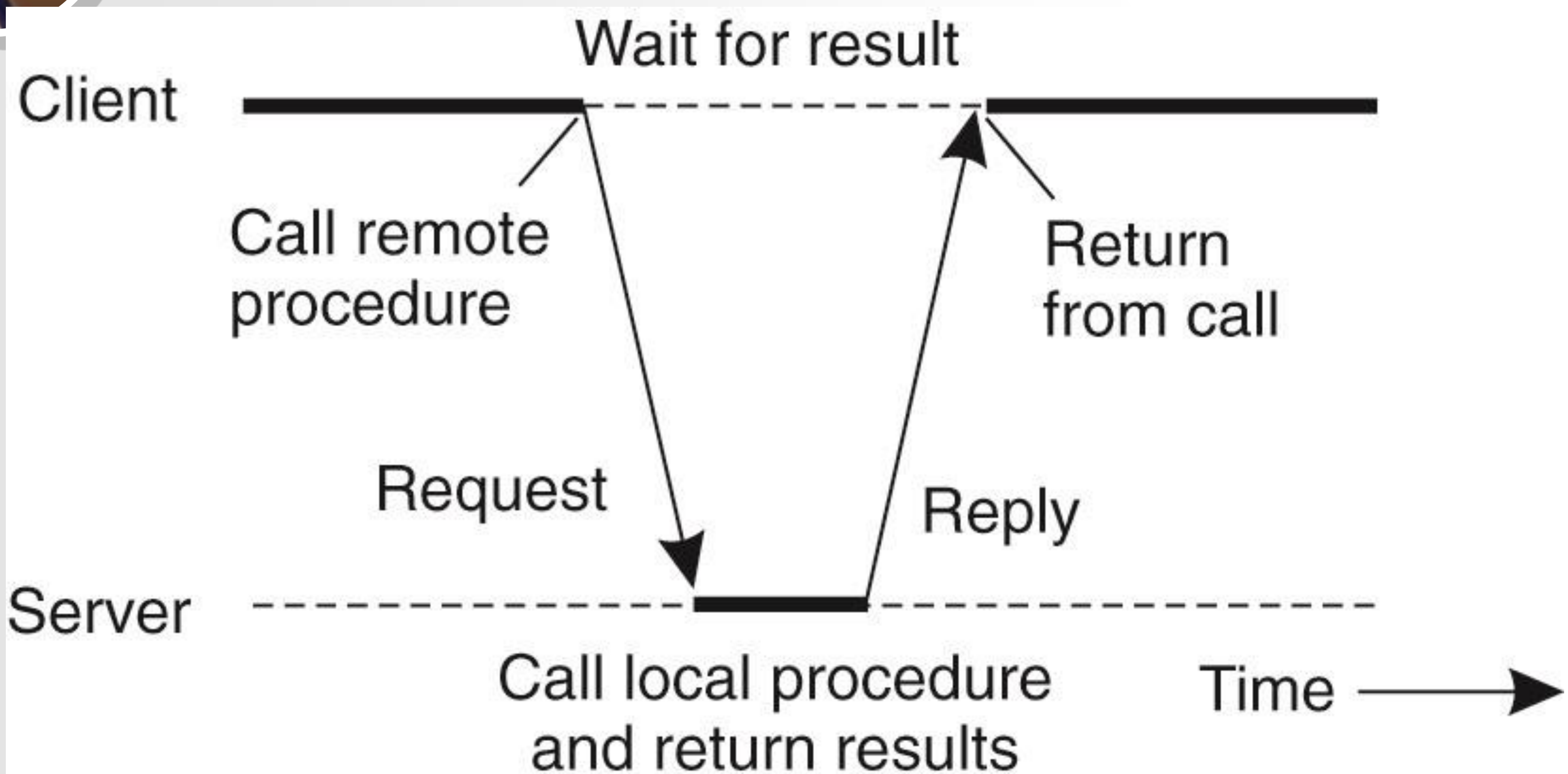


Conventional Procedure Call



- Figure 4-5. (a) Parameter passing in a local procedure call: the stack before the call to read. (b) The stack while the called procedure is active.

Client and Server Stubs



- Figure 4-6. Principle of RPC between a client and server program.

Remote Procedure Calls (1)

- A remote procedure call occurs in the following
- steps:

1. The client procedure calls the client stub in the normal way.
2. The client stub builds a message and calls the local operating system.
3. The client's OS sends the message to the remote OS.
4. The remote OS gives the message to the server stub.
5. The server stub unpacks the parameters and calls the server.

Continued …

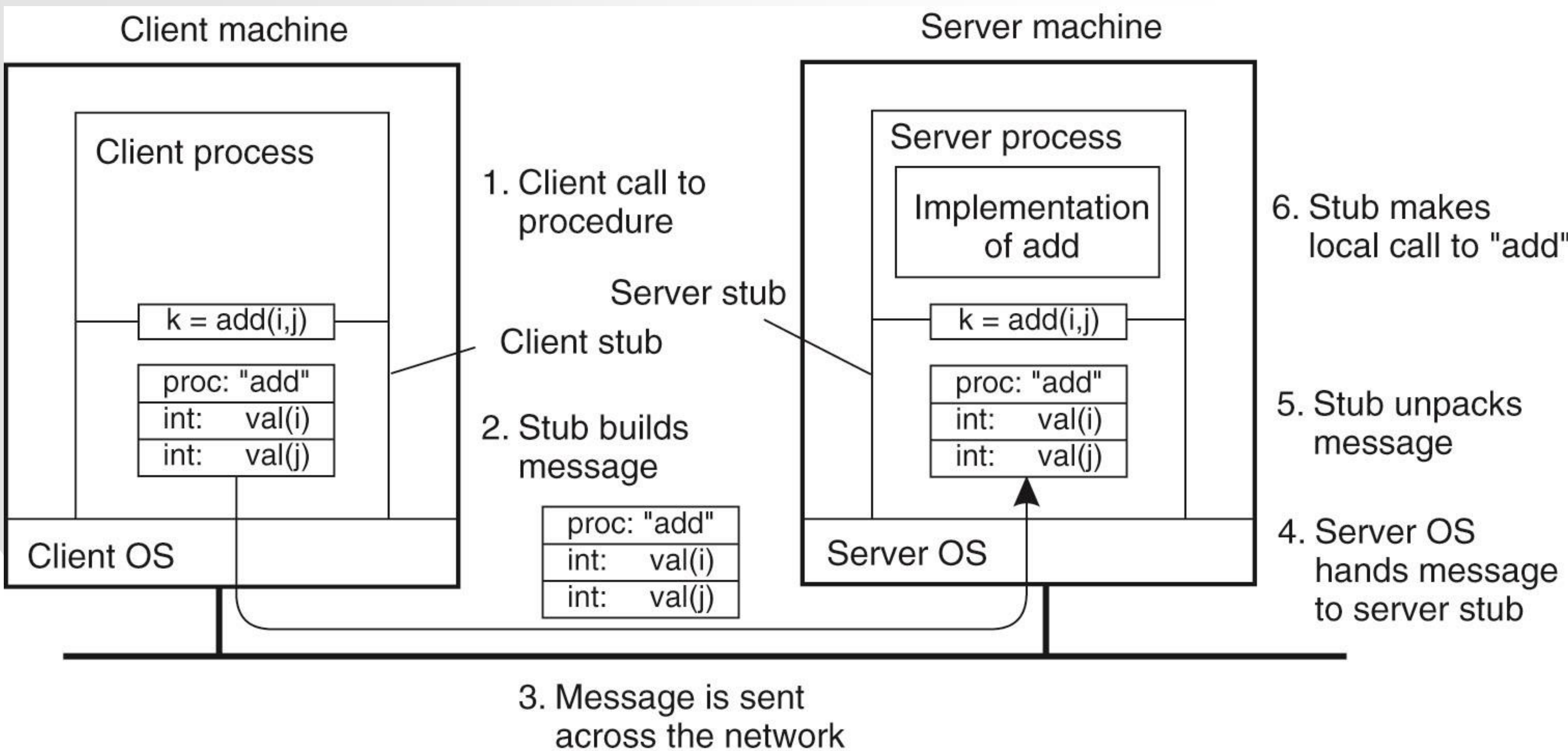
Remote Procedure Calls (2)

- A remote procedure call occurs in the following
- steps (continued):

6. The server does the work and returns the result to the stub.
7. The server stub packs it in a message and calls its local OS.
8. The server’s OS sends the message to the client’s OS.
9. The client’s OS gives the message to the client stub.
10. The stub unpacks the result and returns to the client.

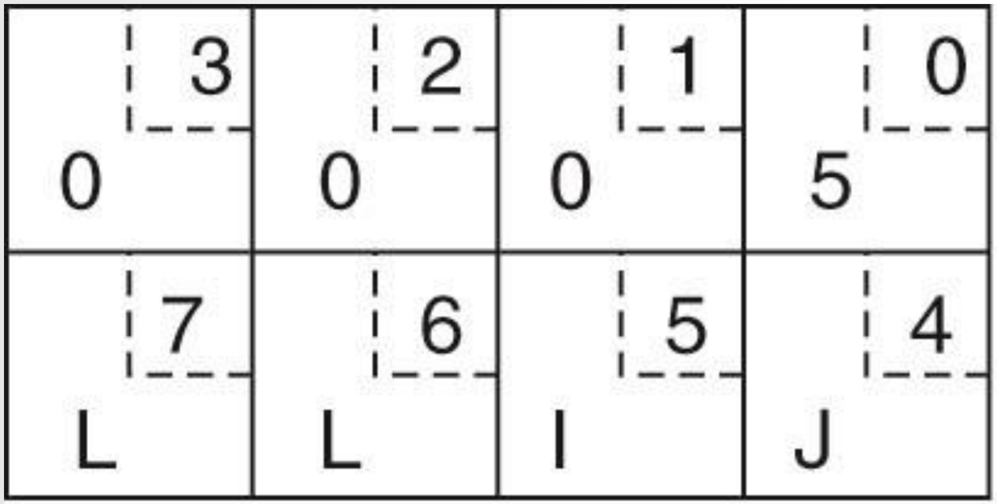
Passing Value Parameters (1)

- Figure 4-7. The steps involved in a doing a remote computation through RPC.



Passing Value Parameters (2)

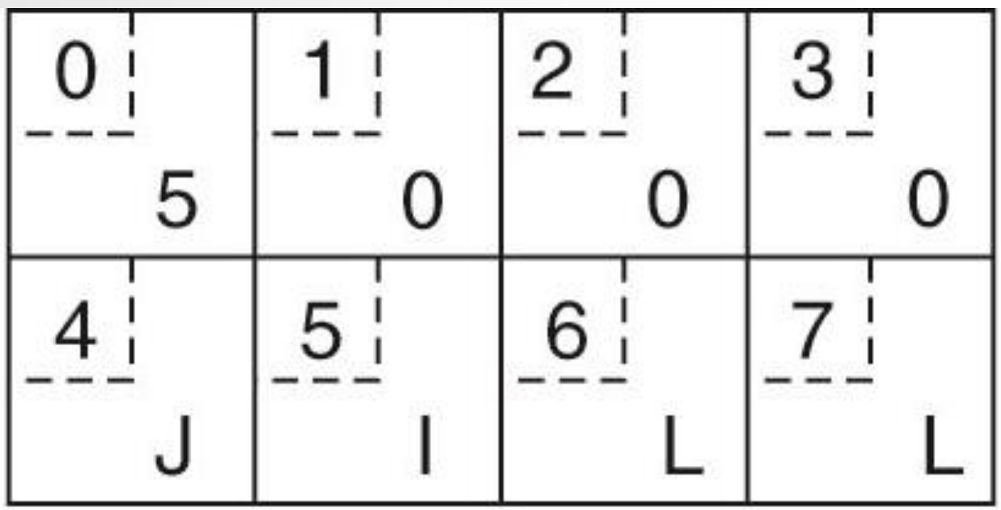
- Figure 4-8. (a) The original message on the Pentium.



(a)

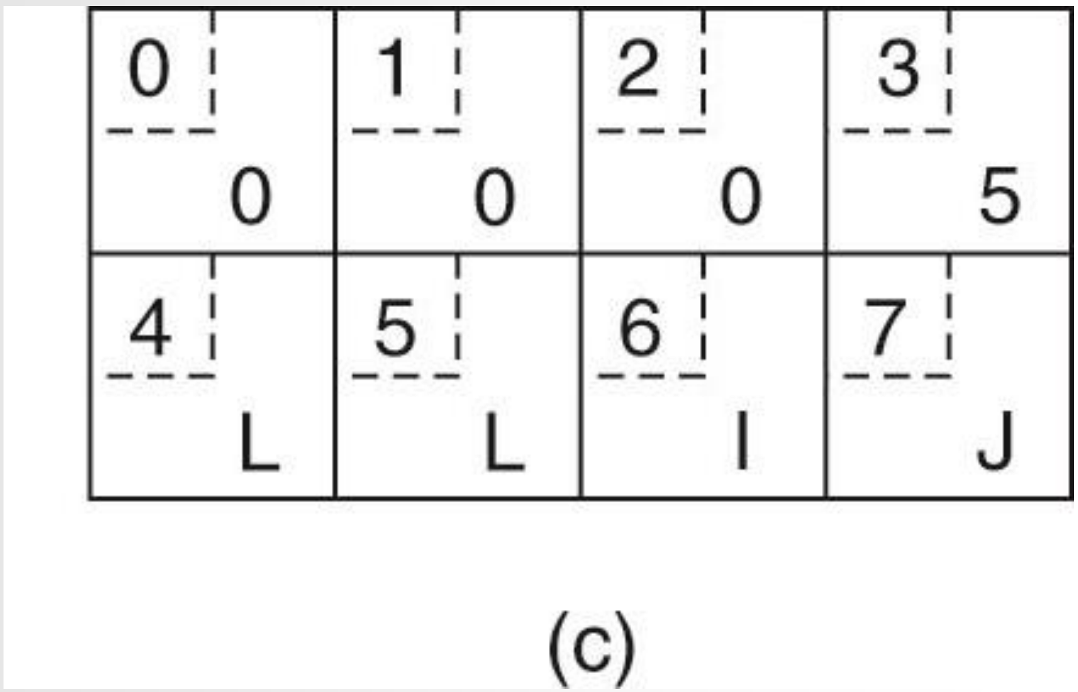
Passing Value Parameters (3)

- Figure 4-8. (a) The original message on the Pentium.



(b)

Passing Value Parameters (4)



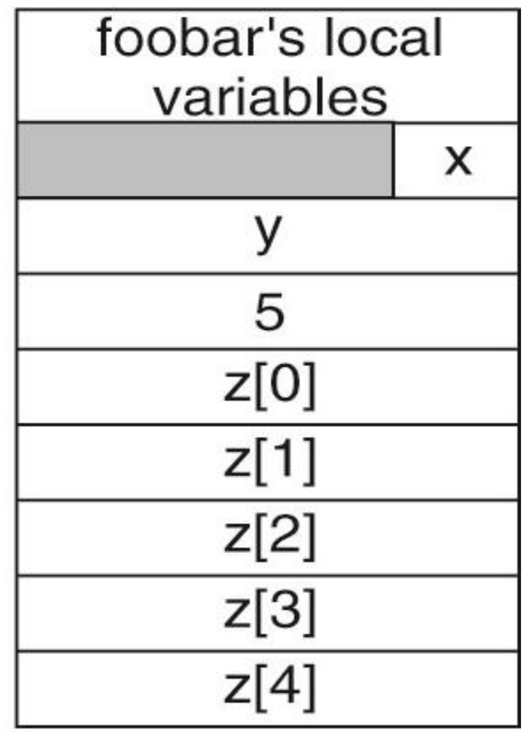
- Figure 4-8. (c) The message after being inverted. The little numbers in boxes indicate the address of each byte.

Parameter Specification and Stub Generation

- Figure 4-9. (a) A procedure. (b) The corresponding message.

```
foobar( char x; float y; int z[5] )
{
   ....
}
```

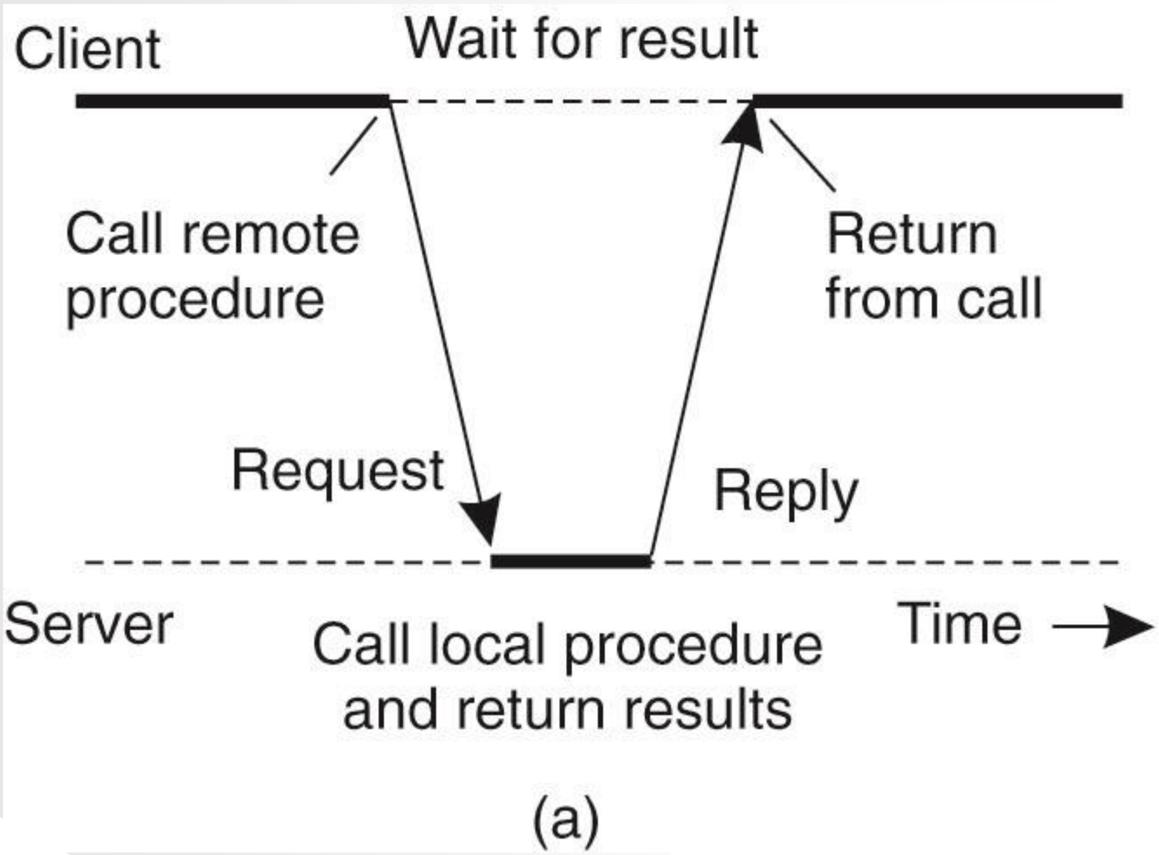
(a)



(b)

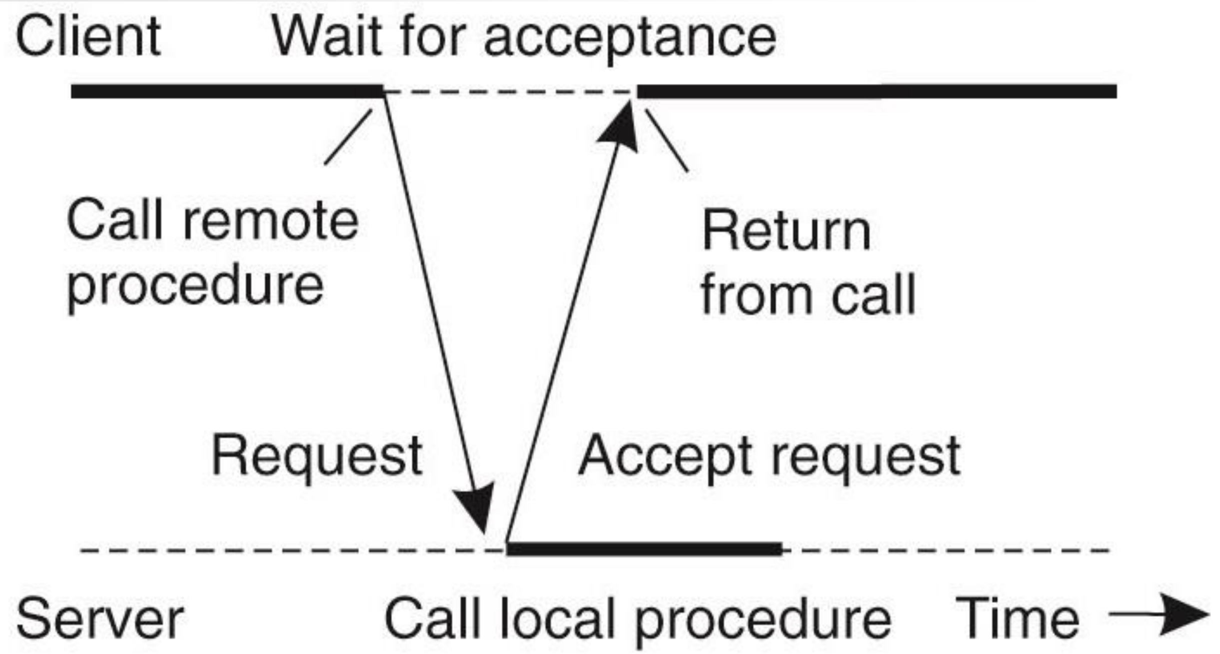
Asynchronous RPC (1)

- Figure 4-10. (a) The interaction between client and server in a traditional RPC.



Asynchronous RPC (2)

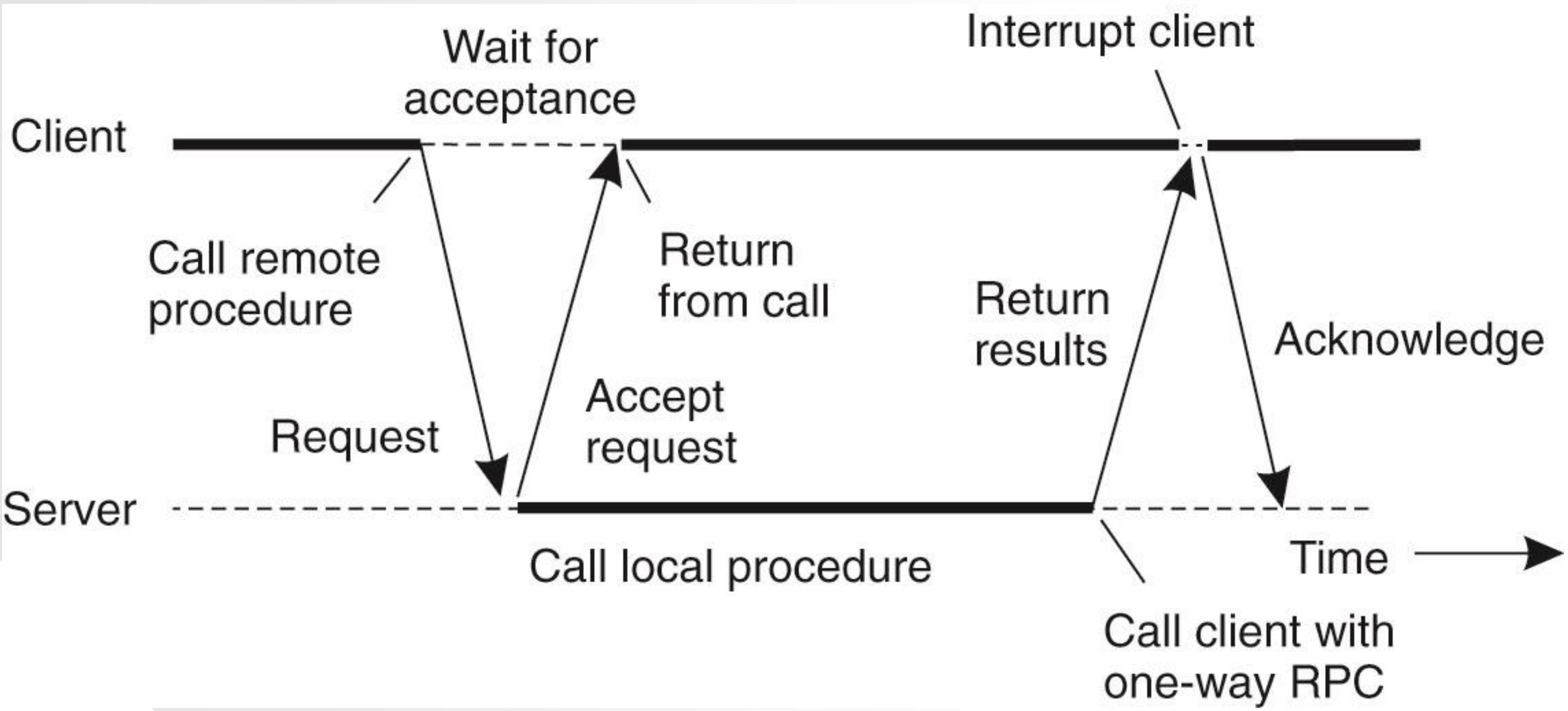
- Figure 4-10. (b) The interaction using asynchronous RPC.



(b)

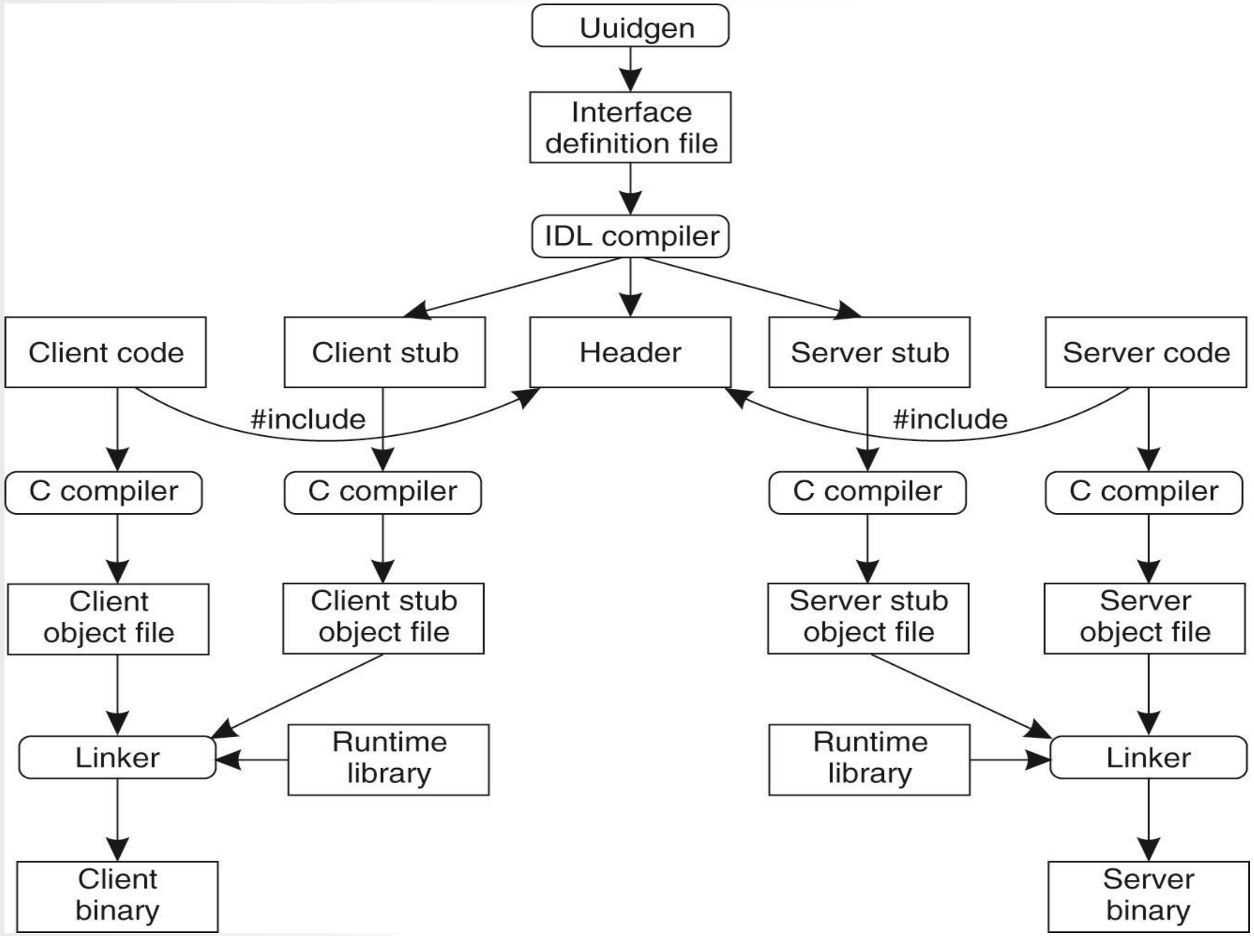
Asynchronous RPC (3)

- Figure 4-11. A client and server interacting through two asynchronous RPCs.



Writing a Client and a Server (1)

- Figure 4-12. The steps in writing a client and a server in DCE RPC.



Writing a Client and a Server (2)

Three files output by the IDL compiler:

- A header file (e.g., interface.h, in C terms).
- The client stub.
- The server stub.



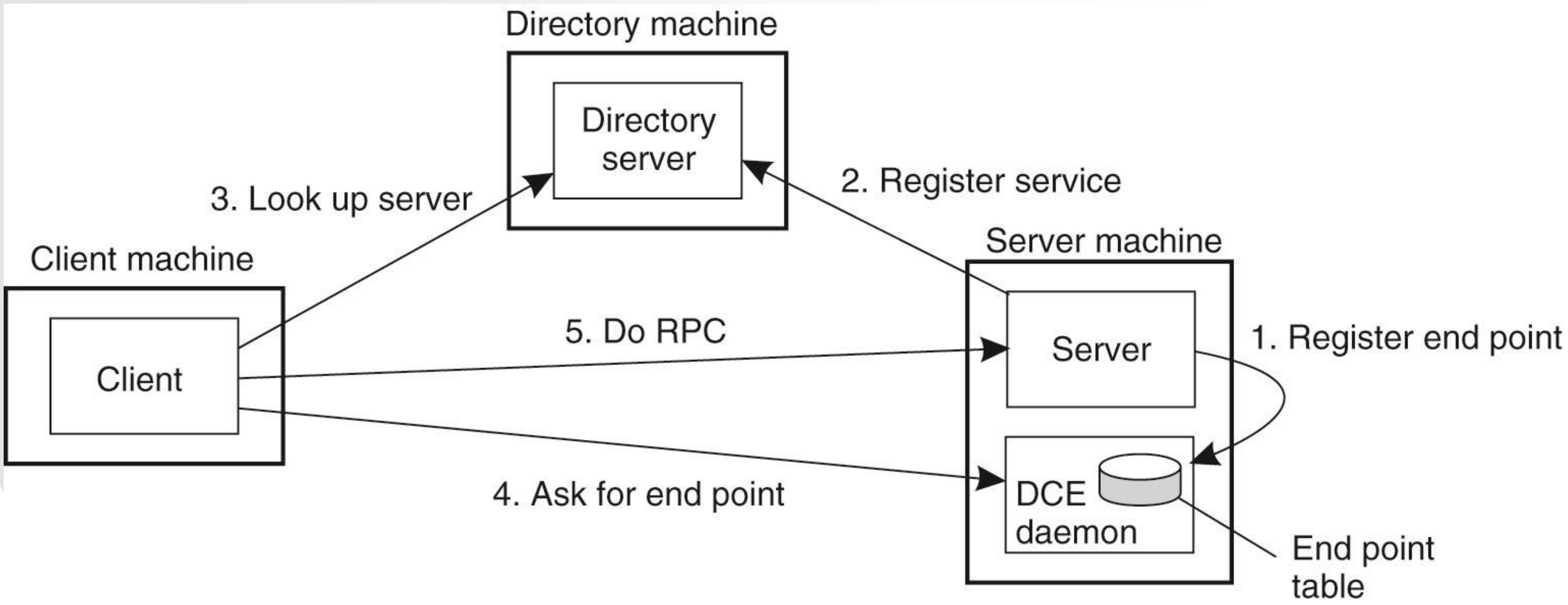
Binding a Client to a Server (1)

- Registration of a server makes it possible for a client to locate the server and bind to it.

- Server location is done in two steps:
  1. Locate the server's machine.
  2. Locate the server on that machine.

Binding a Client to a Server (2)

- Figure 4-13. Client-to-server binding in DCE.

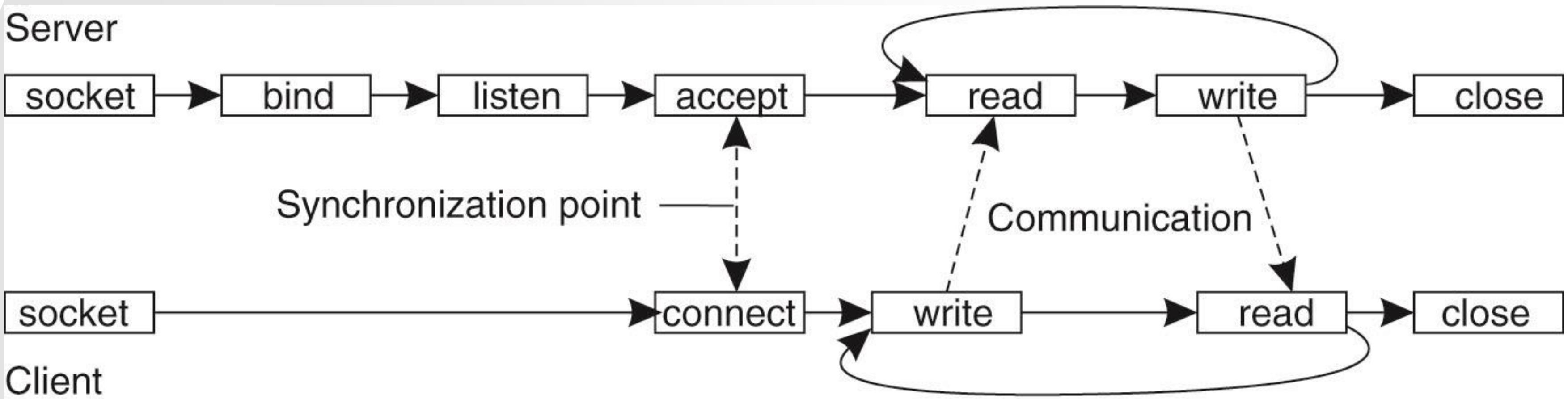


Berkeley Sockets

- Figure 4-14. The socket primitives for TCP/IP.

Primitive	Meaning
Socket	Create a new communication end point
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

The Message-Passing Interface (1)



- Figure 4-15. Connection-oriented communication pattern using sockets.

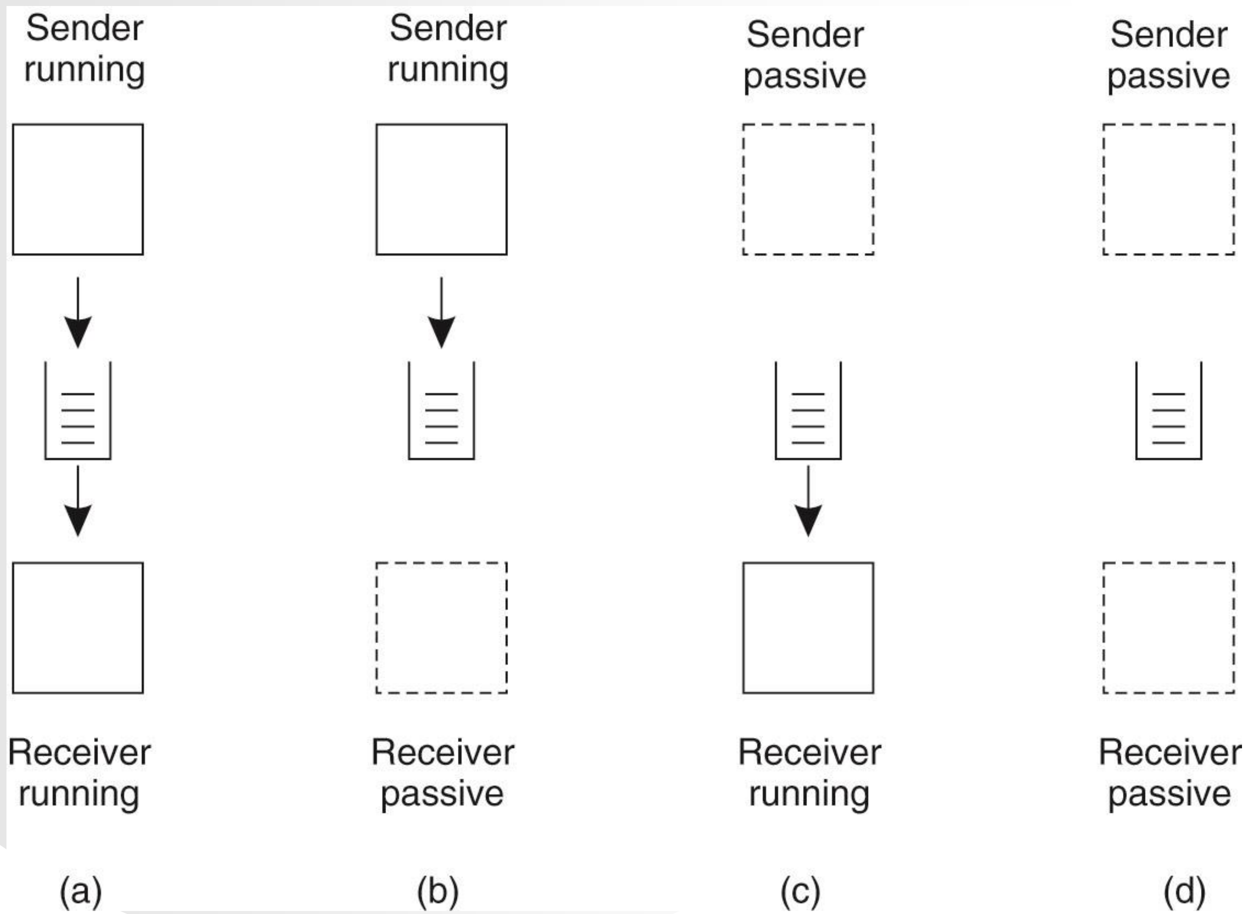
The Message-Passing Interface (2)

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 is none
MPI_irecv	Check if there is an incoming message, but do not block

- Figure 4-16. Some of the most intuitive message-passing primitives of MPI.

Message-Queuing Model (1)

- Figure 4-17. Four combinations for loosely-coupled communications using queues.



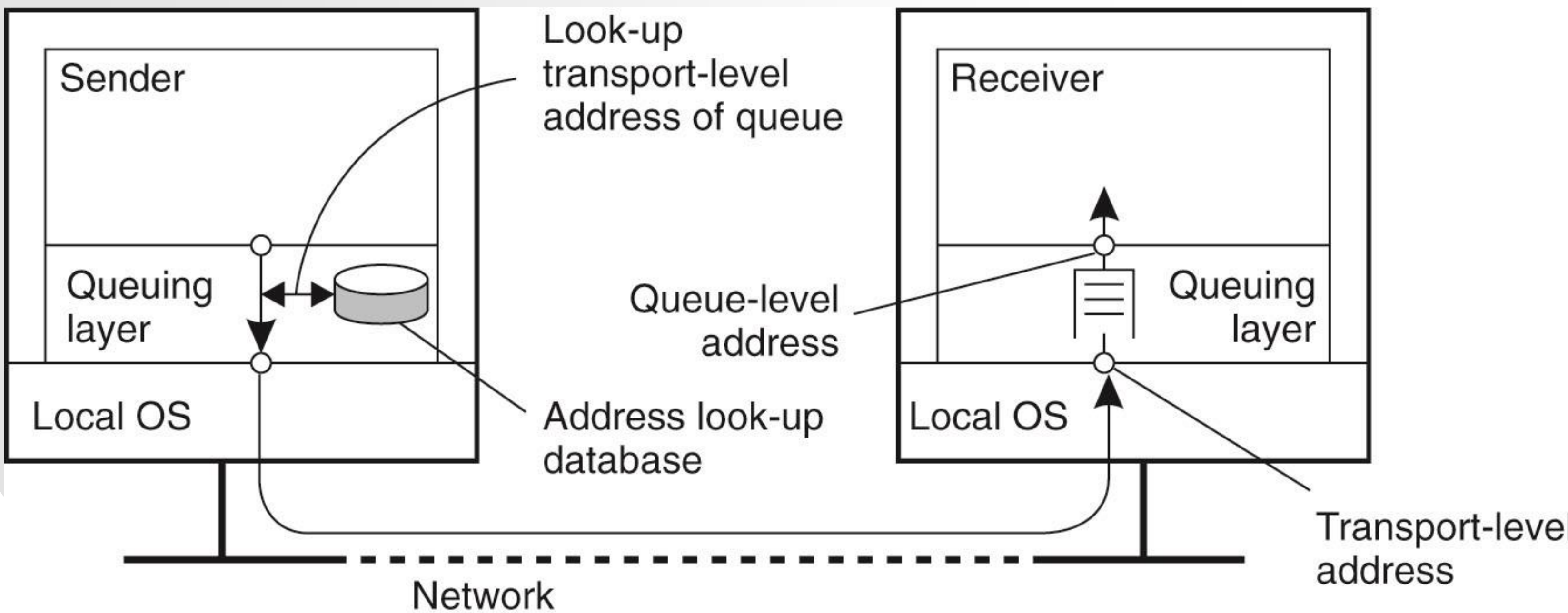
Message-Queuing Model (2)

- Figure 4-18. 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

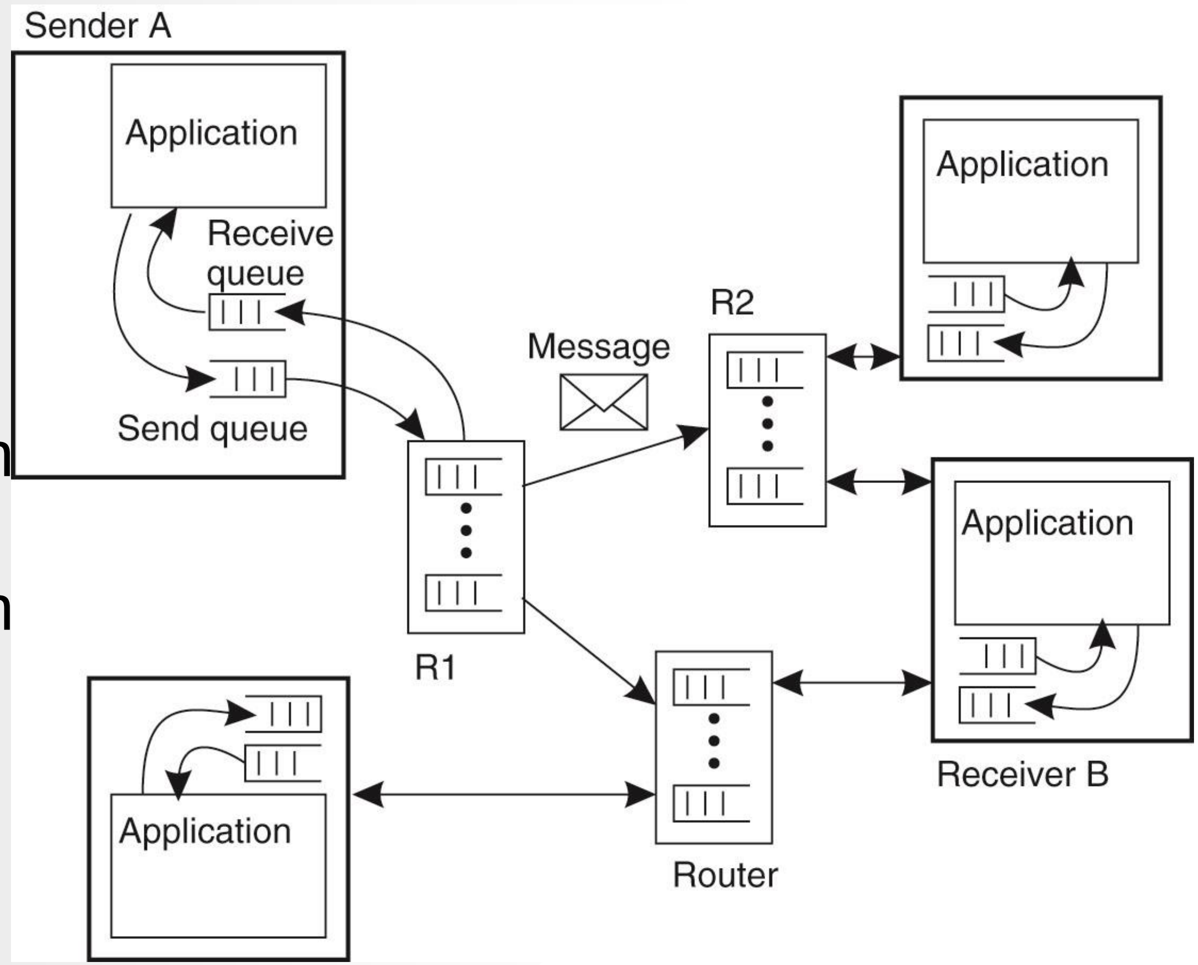
General Architecture of a Message-Queuing System (1)

- Figure 4-19. The relationship between queue-level addressing and network-level addressing.



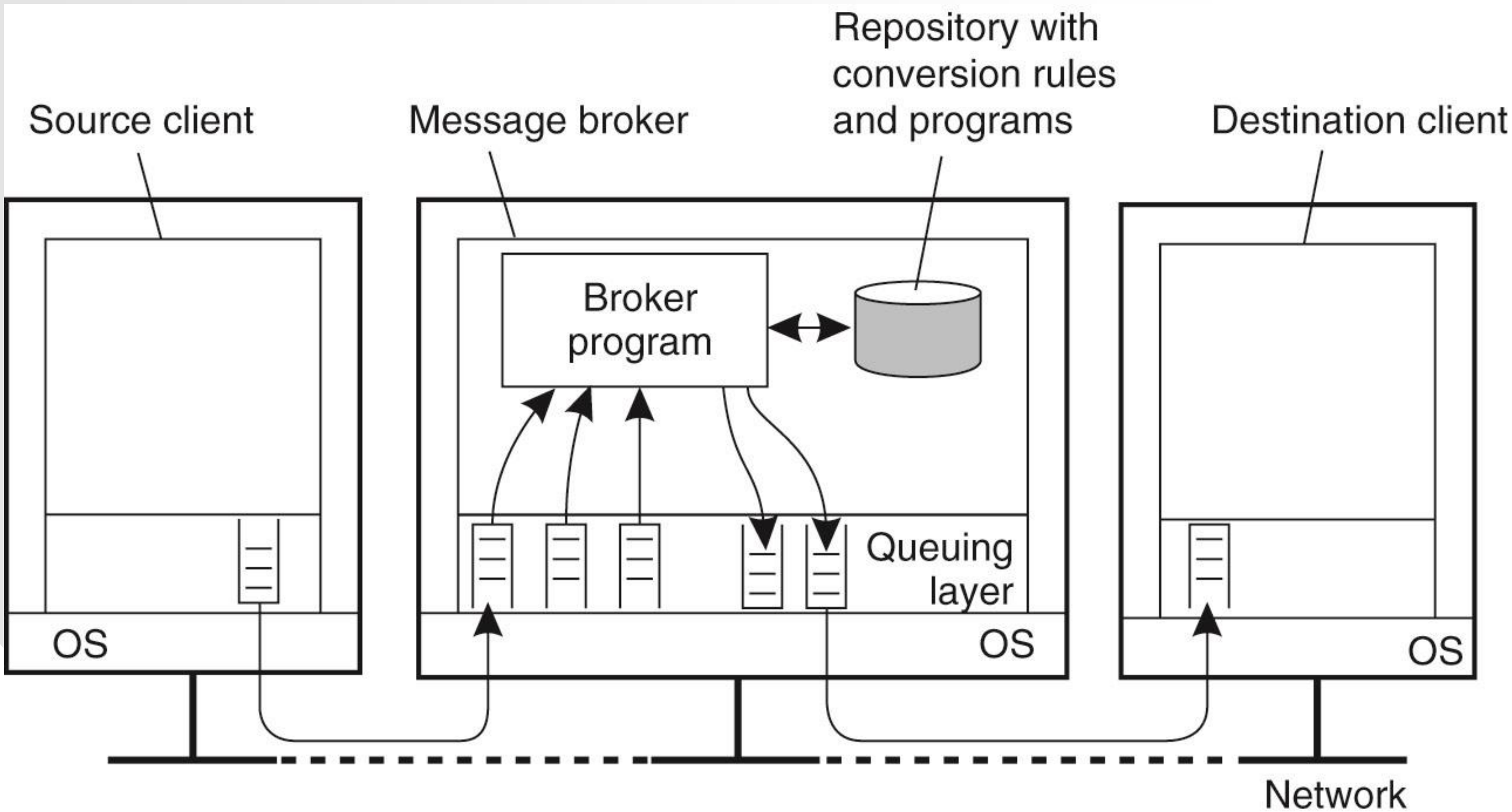
General Architecture of a Message-Queuing System (2)

- Figure 4-20. The
- general organization of a message-queuing system with routers.



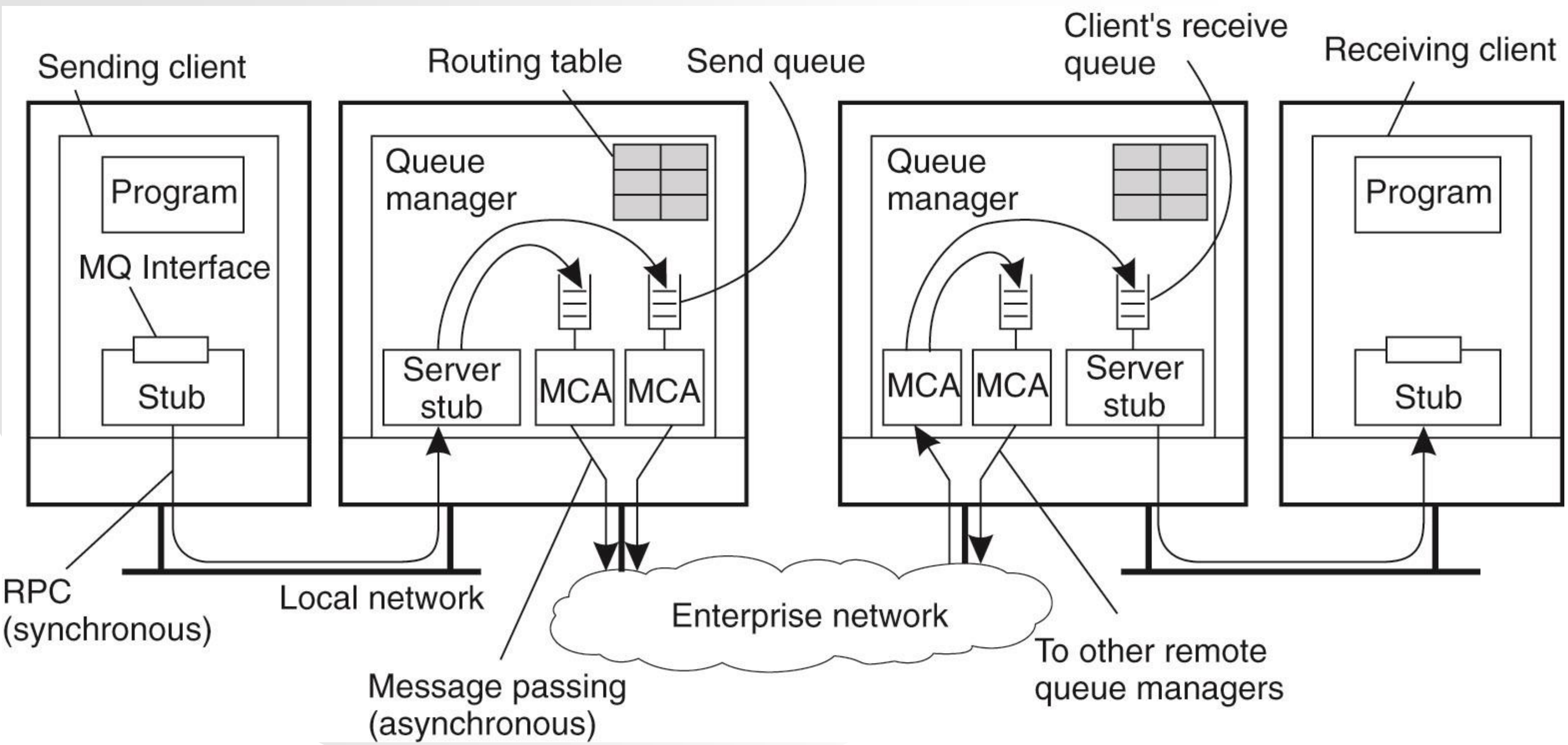
Message Brokers

- Figure 4-21. The general organization of a message broker in a message-queuing system.



IBM's WebSphere Message-Queuing System

- Figure 4-22. General organization of IBM's message-queuing system.



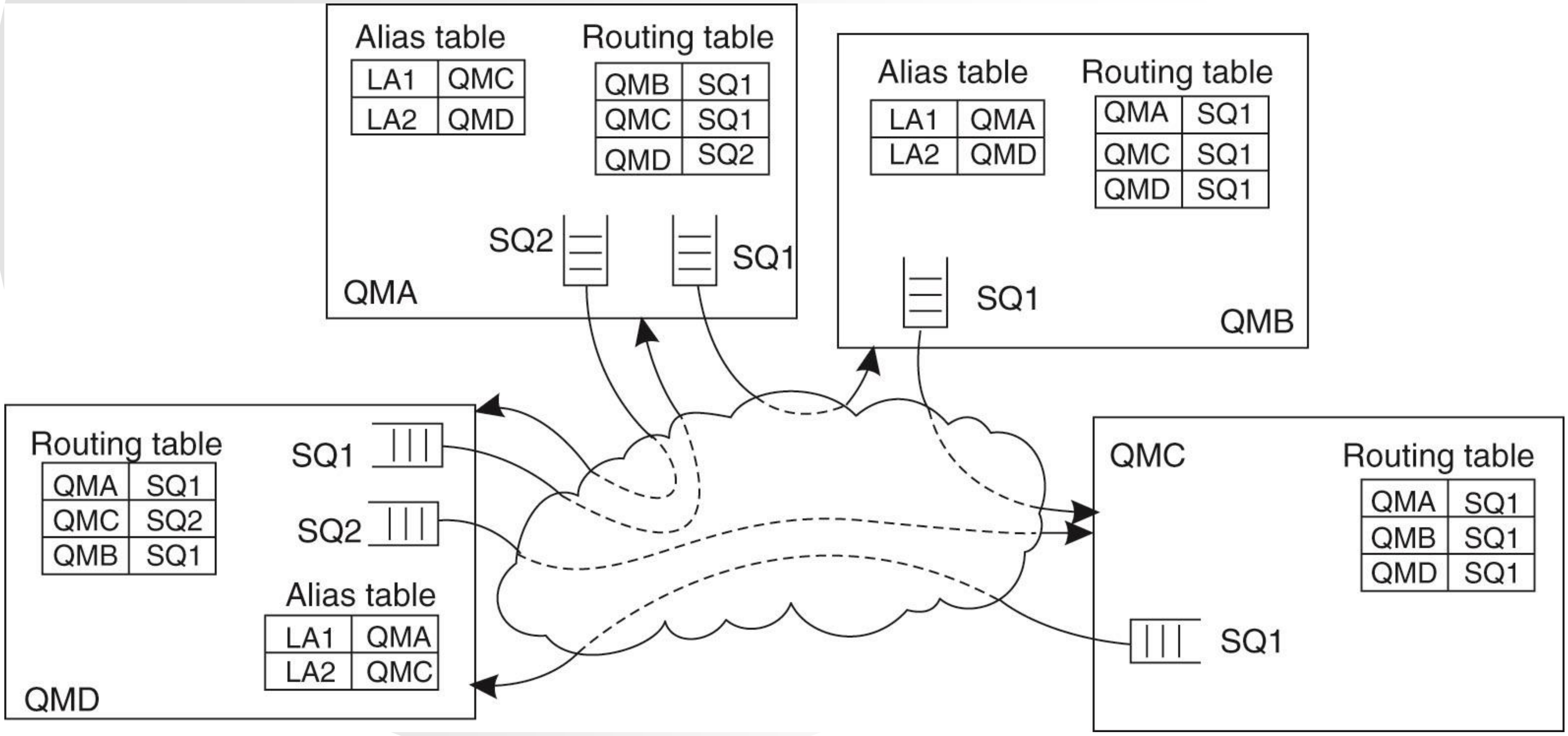
Channels

Attribute	Description
Transport type	Determines the transport protocol to be used
FIFO delivery	Indicates that messages are to be delivered in the order they are sent
Message length	Maximum length of a single message
Setup retry count	Specifies maximum number of retries to start up the remote MCA
Delivery retries	Maximum times MCA will try to put received message into queue

- Figure 4-23. Some attributes associated with message channel agents.

Message Transfer (1)

- Figure 4-24. The general organization of an MQ queuing network using routing tables and aliases.



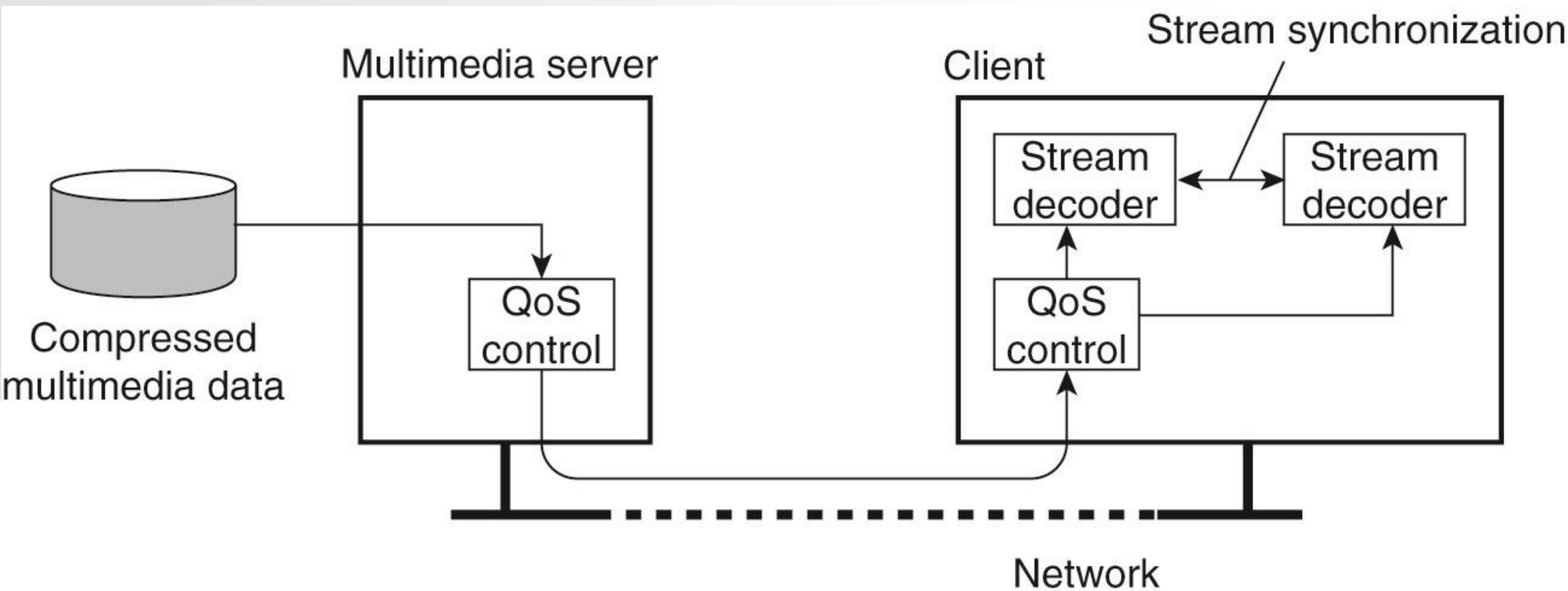
Message Transfer (2)

- Figure 4-25. Primitives available in the message-queuing interface.

Primitive	Description
MQopen	Open a (possibly remote) queue
MQclose	Close a queue
MQput	Put a message into an opened queue
MQget	Get a message from a (local) queue

Data Stream

- Figure 4-26. A general architecture for streaming stored multimedia data over a network.

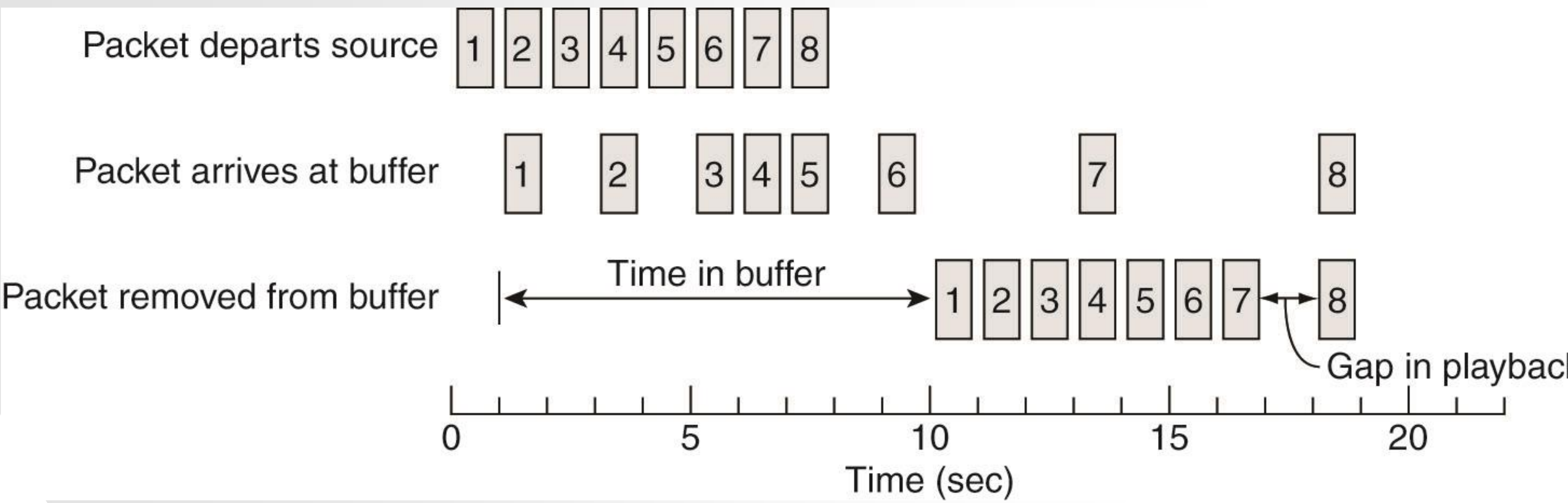



Streams and Quality of Service

- Properties for Quality of Service:
  - The required bit rate at which data should be transported.
  - The maximum delay until a session has been set up
  - The maximum end-to-end delay .
  - The maximum delay variance, or jitter.
  - The maximum round-trip delay.

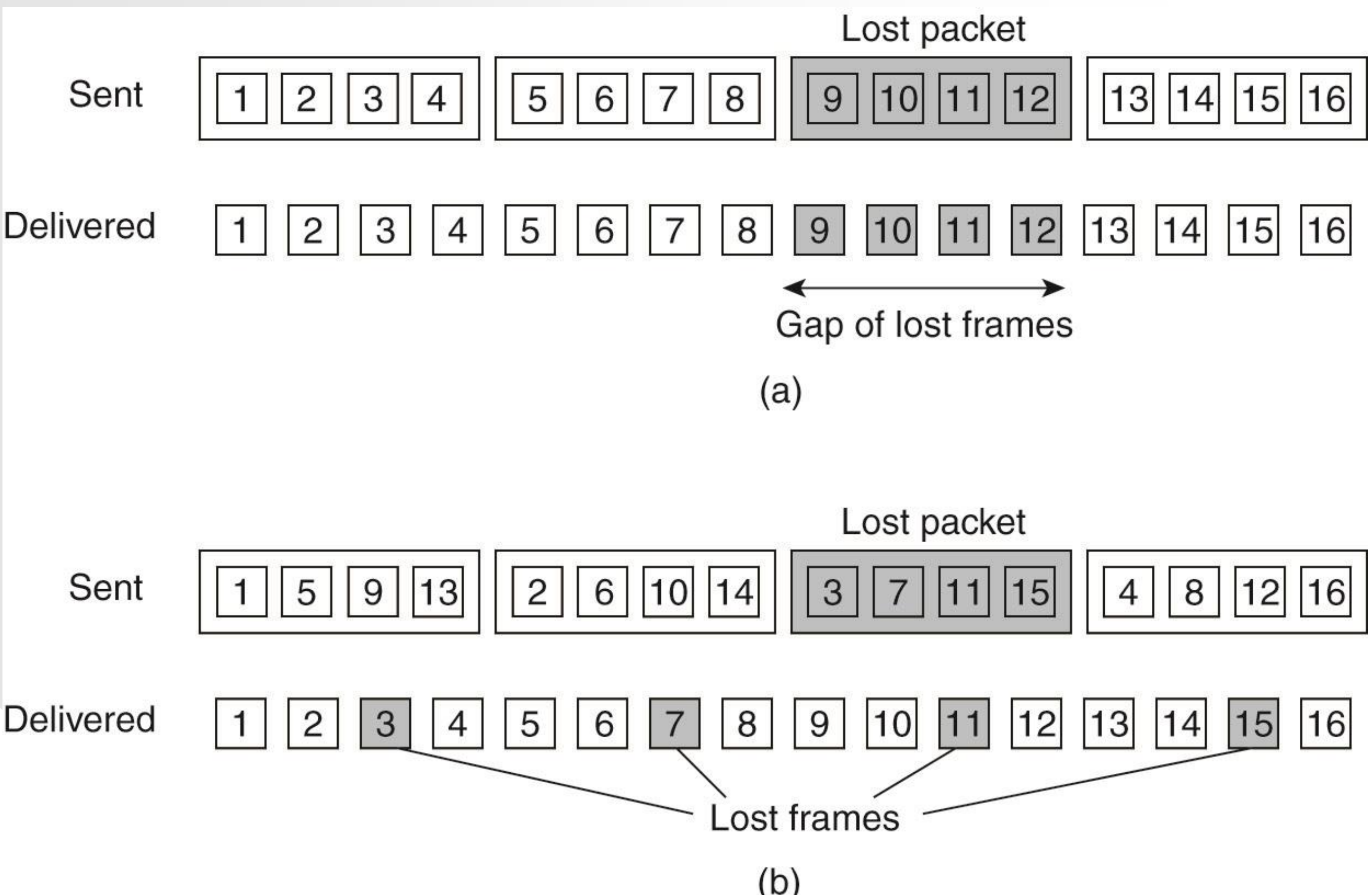
Enforcing QoS (1)

- Figure 4-27. Using a buffer to reduce jitter.



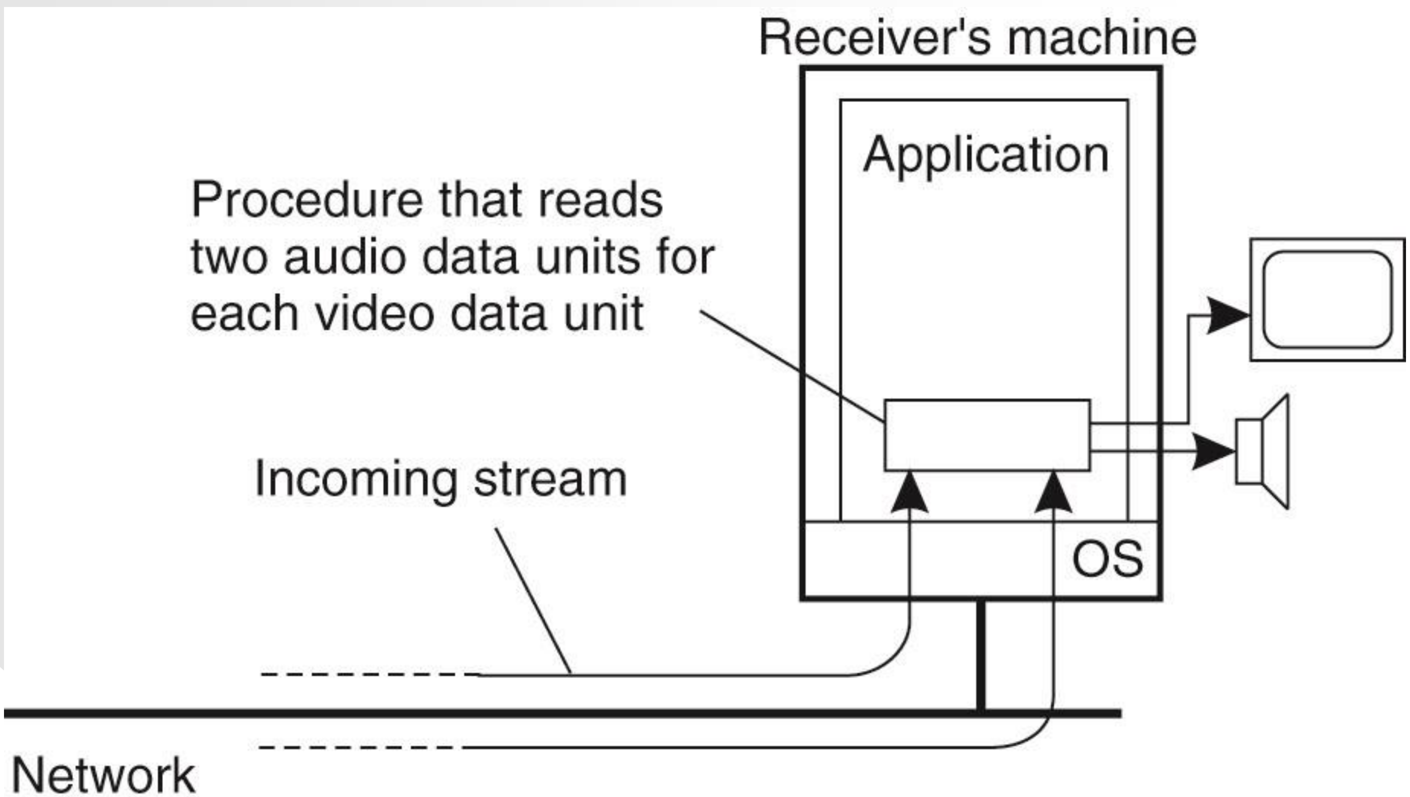
Enforcing QoS (2)

- Figure 4-28. The effect of packet loss in (a) non interleaved transmission and (b) interleaved transmission.



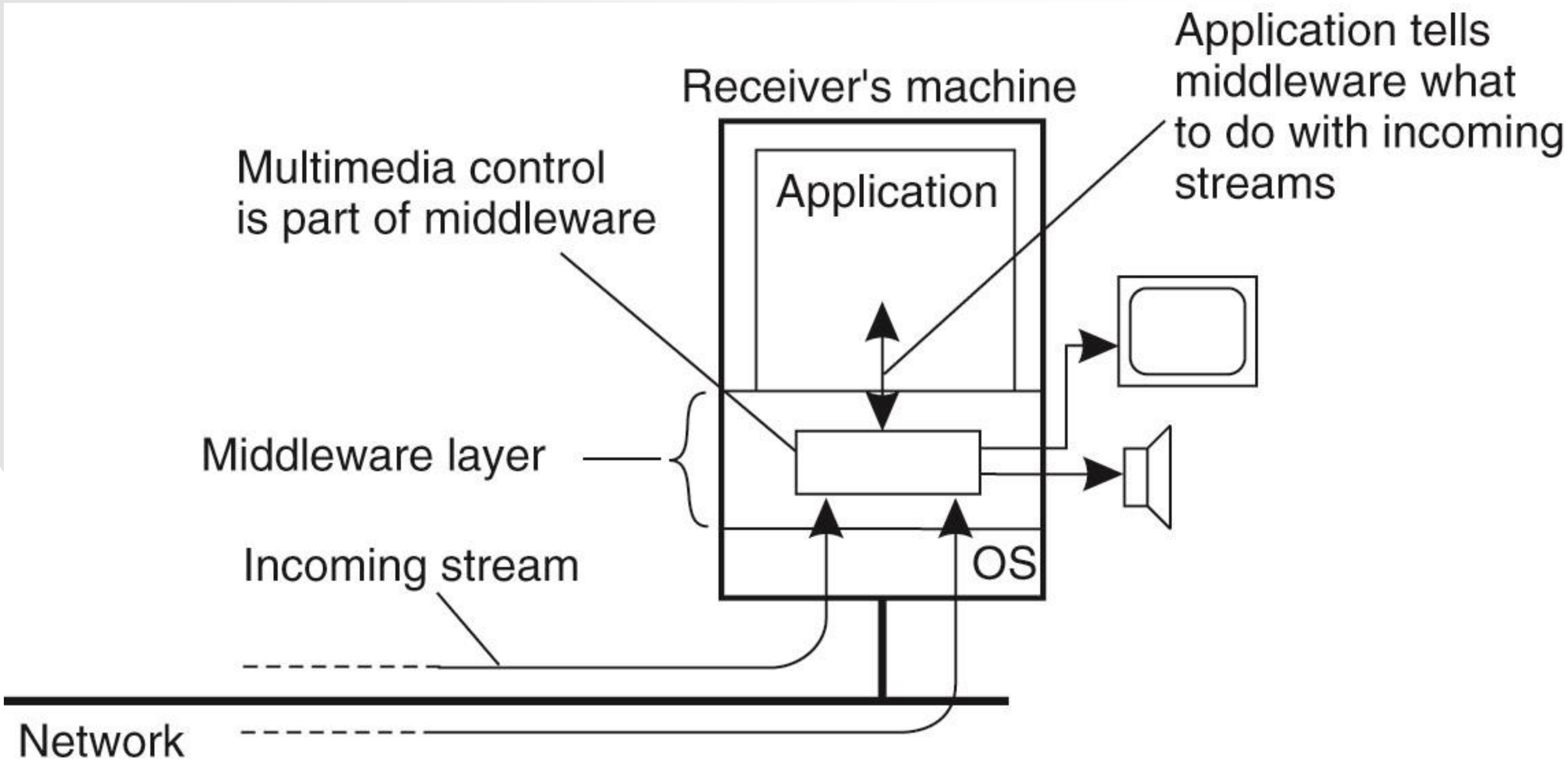
Synchronization Mechanisms (1)

- Figure 4-29. The principle of explicit synchronization on the level data units.



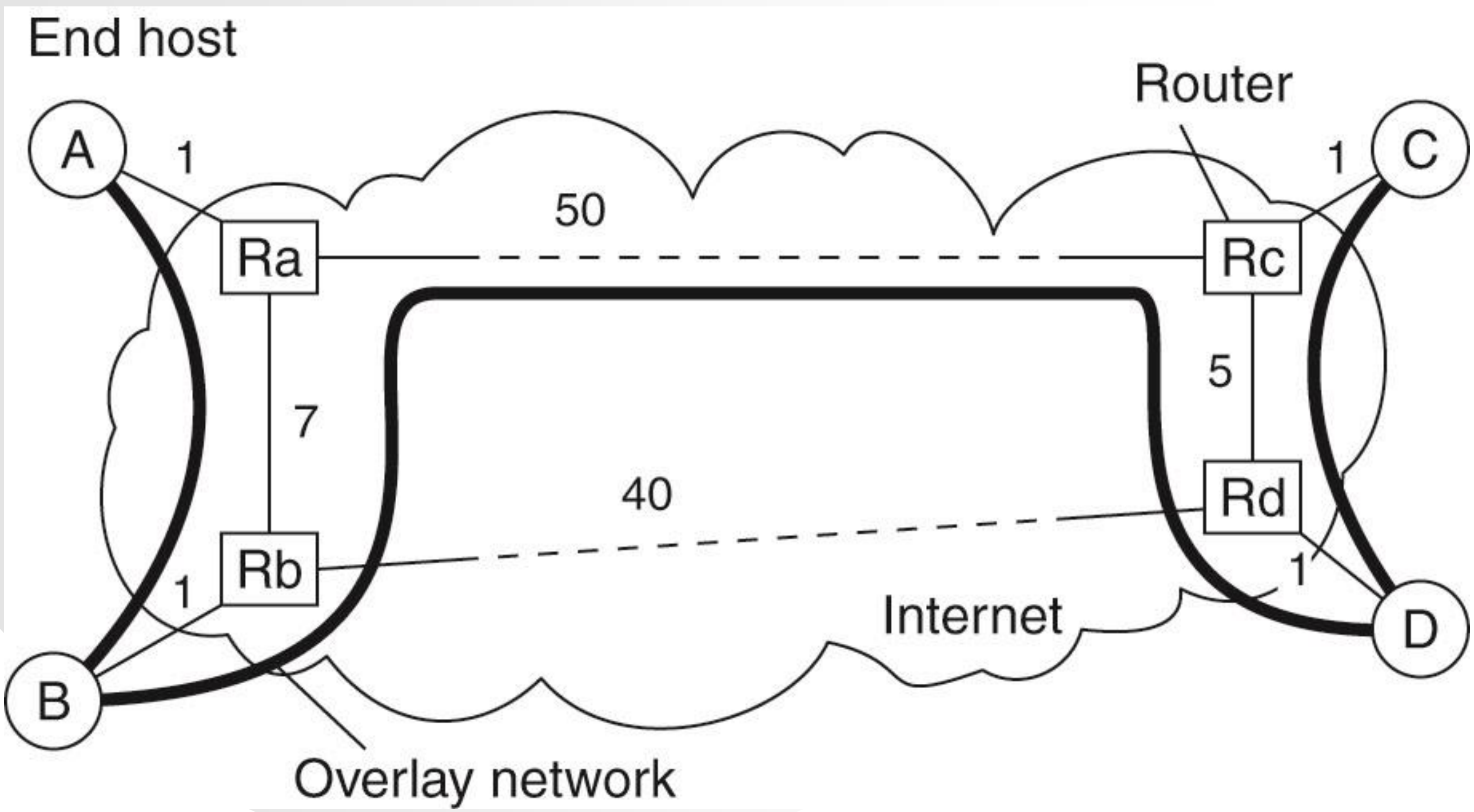
Synchronization Mechanisms (2)

- Figure 4-30. The principle of synchronization as supported by high-level interfaces.



Overlay Construction

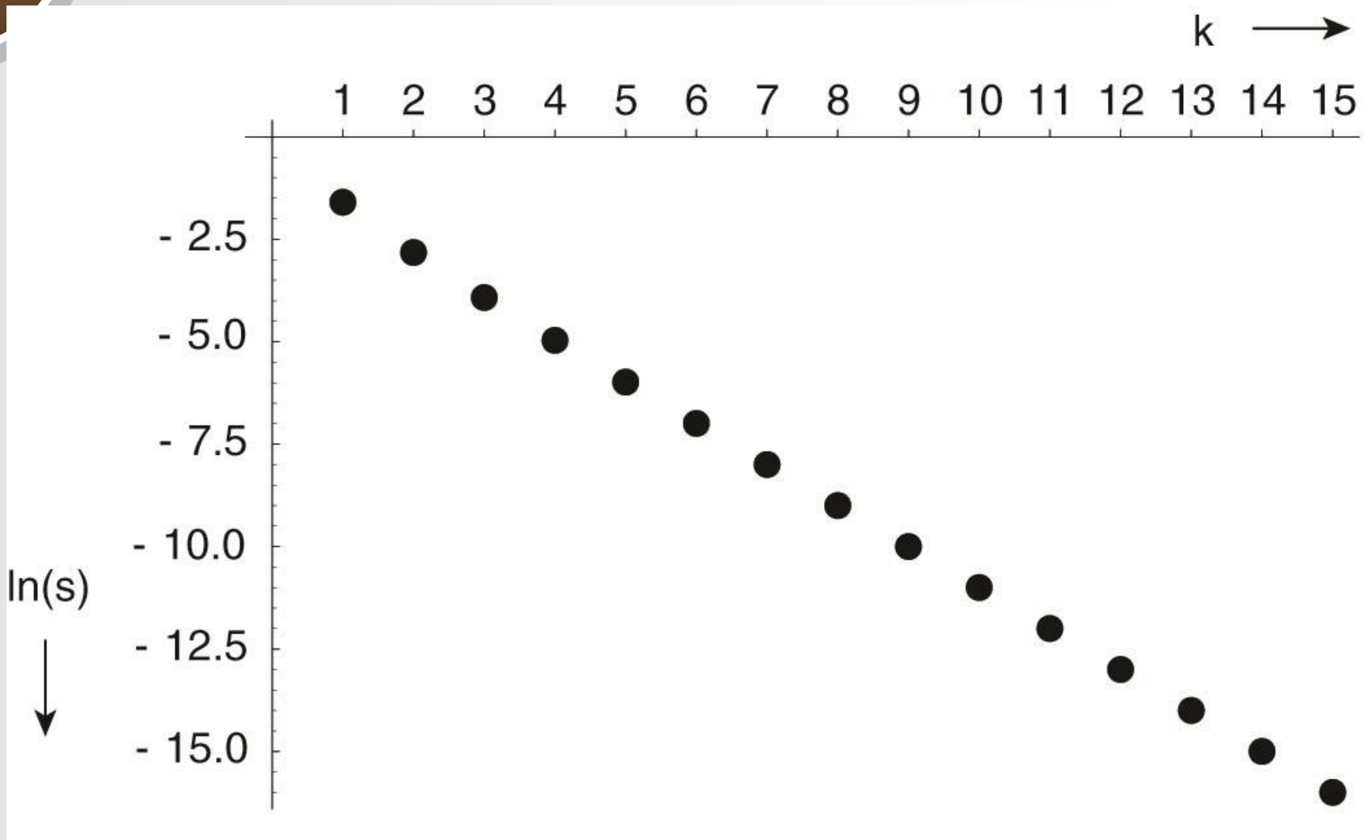
- Figure 4-31. The relation between links in an overlay and actual network-level routes.



Information Dissemination Models (1)

- Anti-entropy propagation model
  - Node P picks another node Q at random
  - Subsequently exchanges updates with Q
- Approaches to exchanging updates
  - P only pushes its own updates to Q
  - P only pulls in new updates from Q
  - P and Q send updates to each other

Information Dissemination Models (2)



❖ Figure 4-32. The relation between the fraction s of update-ignorant nodes and the parameter k in pure gossiping. The graph displays ln(s) as a function of k.

Thank You !