

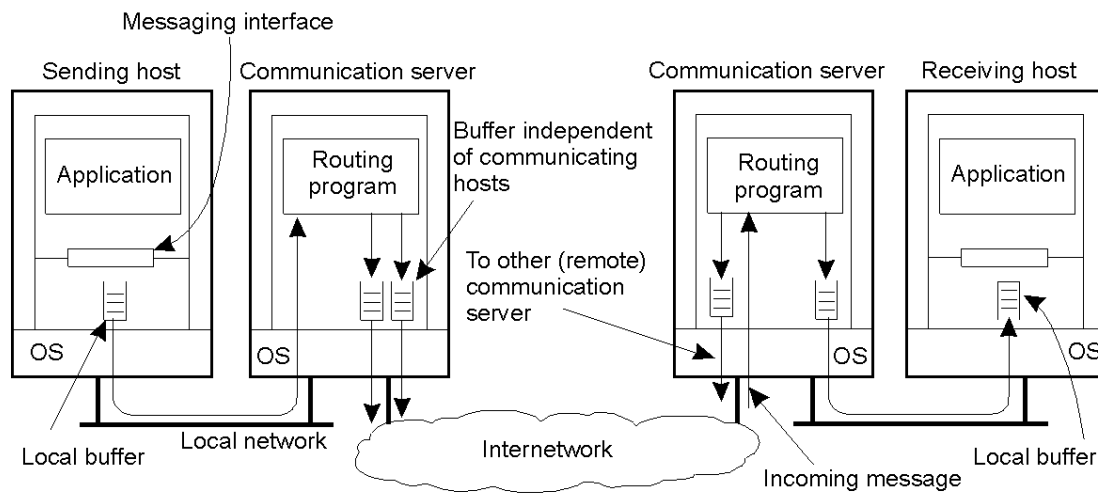
Last Class: RPCs and RMI

- Case Study: Sun RPC
- Lightweight RPCs
- Remote Method Invocation (RMI)
 - Design issues

Today: Communication Issues

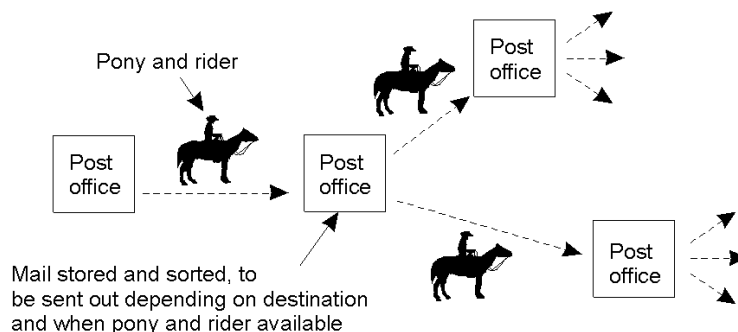
- Message-oriented communication
 - Persistence and synchronicity
- Stream-oriented communication

Persistence and Synchronicity in Communication



Persistence

- Persistent communication
 - Messages are stored until (next) receiver is ready
 - Examples: email, pony express



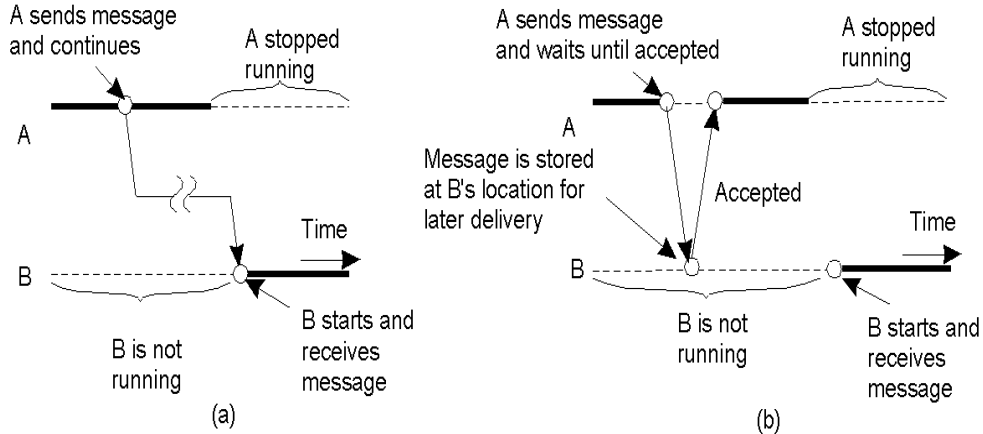
Transient Communication

- Transient communication
 - Message is stored only so long as sending/receiving application are executing
 - Discard message if it can't be delivered to next server/receiver
 - Example: transport-level communication services offer transient communication
 - Example: Typical network router – discard message if it can't be delivered next router or destination

Synchronicity

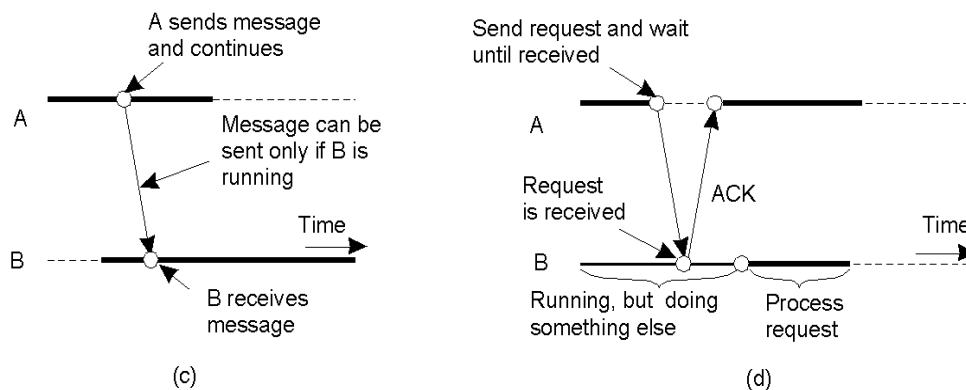
- Asynchronous communication
 - Sender continues immediately after it has submitted the message
 - Need a local buffer at the sending host
- Synchronous communication
 - Sender blocks until message is stored in a local buffer at the receiving host or actually delivered to sending
 - Variant: block until receiver processes the message
- Six combinations of persistence and synchronicity

Persistence and Synchronicity Combinations



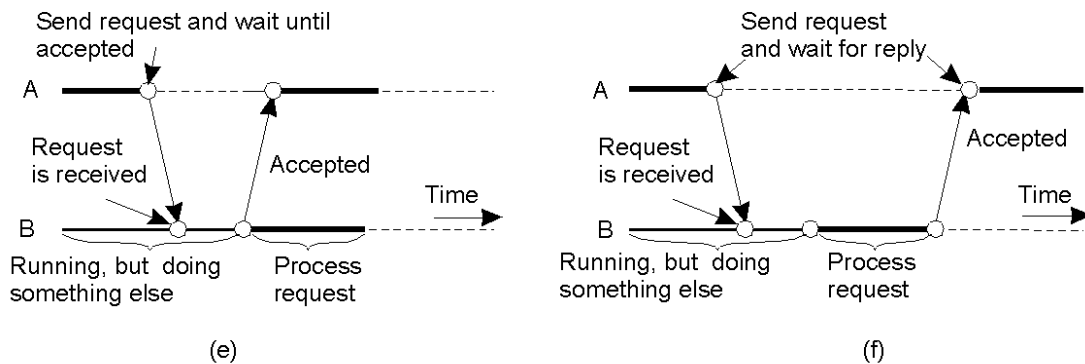
- a) Persistent asynchronous communication (e.g., email)
- b) Persistent synchronous communication

Persistence and Synchronicity Combinations



- c) Transient asynchronous communication (e.g., UDP)
- d) Receipt-based transient synchronous communication

Persistence and Synchronicity Combinations

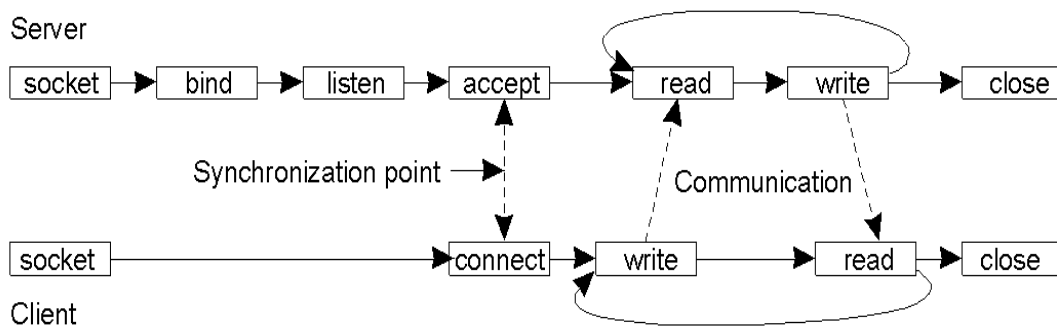


- e) Delivery-based transient synchronous communication at message delivery (e.g., asynchronous RCP)
- f) Response-based transient synchronous communication (RPC)



Message-oriented Transient Communication

- Many distributed systems built on top of simple message-oriented model
 - Example: Berkeley sockets



Berkeley Socket Primitives

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-Passing Interface (MPI)

- Sockets designed for network communication (e.g., TCP/IP)
 - Support simple send/receive primitives
- Abstraction not suitable for other protocols in clusters of workstations or massively parallel systems
 - Need an interface with more advanced primitives
- Large number of incompatible proprietary libraries and protocols
 - Need for a standard interface
- Message-passing interface (MPI)
 - Hardware independent
 - Designed for parallel applications (uses transient communication)
- Key idea: communication between groups of processes
 - Each endpoint is a (*groupID*, *processID*) pair

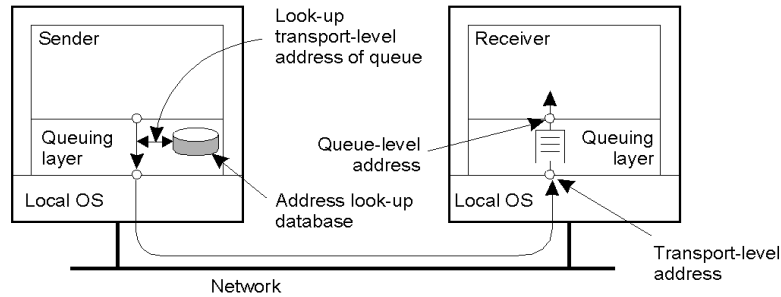
MPI Primitives

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 systems
 - Support asynchronous persistent communication
 - Intermediate storage for message while sender/receiver are inactive
 - Example application: email
- Communicate by inserting messages in queues
- Sender is only guaranteed that message will be eventually inserted in recipient's queue
 - No guarantees on when or if the message will be read
 - “Loosely coupled communication”

Message-Queuing Model

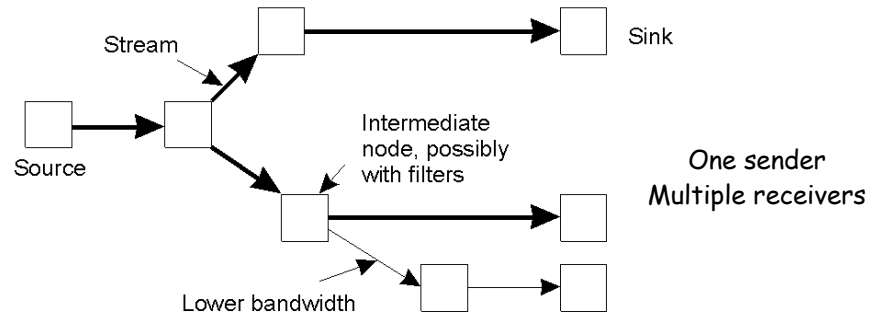
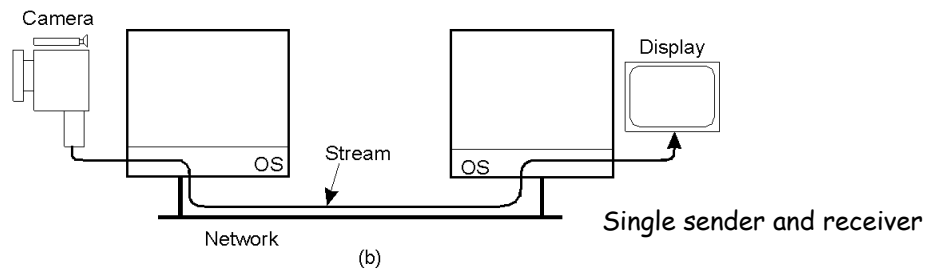


Primitive	Meaning
Put	Append a message to a specified queue
Get	Block until the specified queue is nonempty, and remove the first message
Poll	Check a specified queue for messages, and remove the first. Never block.
Notify	Install a handler to be called when a message is put into the specified queue.

Stream Oriented Communication

- Message-oriented communication: request-response
 - When communication occurs and speed do not affect correctness
- Timing is crucial in certain forms of communication
 - Examples: audio and video (“continuous media”)
 - 30 frames/s video => receive and display a frame every 33ms
- Characteristics
 - Isochronous communication
 - Data transfers have a maximum bound on end-end delay and jitter
 - Push mode: no explicit requests for individual data units beyond the first “play” request

Examples

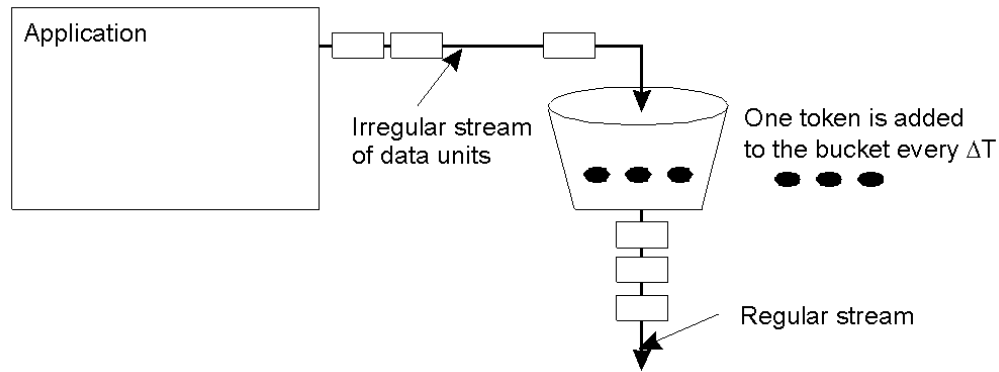


Quality of Service (QoS)

- Time-dependent and other requirements are specified as *quality of service (QoS)*
 - Requirements/desired guarantees from the underlying systems
 - Application specifies workload and requests a certain service quality
 - Contract between the application and the system

Characteristics of the Input	Service Required
<ul style="list-style-type: none"> • maximum data unit size (bytes) • Token bucket rate (bytes/sec) • Token bucket size (bytes) • Maximum transmission rate (bytes/sec) 	<ul style="list-style-type: none"> • Loss sensitivity (bytes) • Loss interval (μsec) • Burst loss sensitivity (data units) • Minimum delay noticed (μsec) • Maximum delay variation (μsec) • Quality of guarantee

Specifying QoS: Token bucket



- The principle of a token bucket algorithm
 - Parameters (rate r , burst b)
 - Rate is the average rate, burst is the maximum number of packets that can arrive simultaneously

Setting Up a Stream: RSVP

