Outline

• Message-oriented communication
  – Persistence and synchronicity
  – Message-oriented transient communication
    • Berkeley socket
    • MPI
  – Message-oriented persistent communication

• Stream-oriented communication
  – Data stream
  – Quality of services
  – Stream synchronization
Example: Communication System

Example: e-mail system
• Persistent communication
  – Definition:
  – Examples: email, pony express

Mail stored and sorted, to be sent out depending on destination and when pony and rider available.
• Transient communication
  – Example: transport-level communication services offer transient communication
  – Example: Typical network router
• Asynchronous communication

• Synchronous communication

• Six combinations of persistence and synchronicity
a) Persistent asynchronous communication (e.g., email)
b) Persistent synchronous communication
c) Transient asynchronous communication (e.g., UDP)
d) Receipt-based transient synchronous communication
e) Delivery-based transient synchronous communication at message delivery (e.g., asynchronous RCP)
f) Response-based transient synchronous communication (RPC)
Summary of Synchronicity

• Persistent communication
  – Messages are stored by communication middleware for as long as needed to ensure delivery of message
  – Example: email

• Transient communication
  – Messages are stored by communication middleware only for as long as the sending and receiving application are executing
  – Example: TCP/UDP

• Asynchronous communication
  – Sender continues immediately after message sent

• Synchronous communication
  – Sender blocks until the request is known to be accepted
Persistence and Synchronicity: Comments

• Transient synchronous comm: response-based, delivery-based and reply-based

• Transient asynchronous comm: message-passing systems

• Persistent comm: developing of middleware for large-scale interconnected networks; failure masking and recovery
Many distributed systems built on top of simple message-oriented model
  - Example: Berkeley sockets
  - Socket?

Server

Client

Synchronization point

Communication

CS550: Advanced Operating Systems
# Berkeley Socket Primitives

<table>
<thead>
<tr>
<th>Primitive</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Socket</td>
<td>Create a new communication endpoint</td>
</tr>
<tr>
<td>Bind</td>
<td>Attach a local address to a socket</td>
</tr>
<tr>
<td>Listen</td>
<td>Announce willingness to accept connections</td>
</tr>
<tr>
<td>Accept</td>
<td>Block caller until a connection request arrives</td>
</tr>
<tr>
<td>Connect</td>
<td>Actively attempt to establish a connection</td>
</tr>
<tr>
<td>Send</td>
<td>Send some data over the connection</td>
</tr>
<tr>
<td>Receive</td>
<td>Receive some data over the connection</td>
</tr>
<tr>
<td>Close</td>
<td>Release the connection</td>
</tr>
</tbody>
</table>
• Sockets designed for network communication (e.g., TCP/IP)
  – Support simple send/receive primitives
  – Use general-purpose protocol stacks such as TCP/IP

• 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)

• Message-passing interface (MPI)
  – Hardware independent
  – Designed for parallel applications (uses transient communication)

• Key idea: communication between groups of processes
  – Each endpoint is a \((\text{groupId}, \text{processId})\) pair

• Support most of the forms of transient communication (c)-(f)
## MPI Primitives

<table>
<thead>
<tr>
<th>Primitive</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MPI_bsend</td>
<td>Append outgoing message to a local send buffer</td>
</tr>
<tr>
<td>MPI_send</td>
<td>Send a message and wait until copied to local or remote buffer</td>
</tr>
<tr>
<td>MPI_ssend</td>
<td>Send a message and wait until receipt starts</td>
</tr>
<tr>
<td>MPI_sendrecv</td>
<td>Send a message and wait for reply</td>
</tr>
<tr>
<td>MPI_isend</td>
<td>Pass reference to outgoing message, and continue</td>
</tr>
<tr>
<td>MPI_isend</td>
<td>Pass reference to outgoing message, and wait until receipt starts</td>
</tr>
<tr>
<td>MPI_recv</td>
<td>Receive a message; block if there are none</td>
</tr>
<tr>
<td>MPI Irerecv</td>
<td>Check if there is an incoming message, but do not block</td>
</tr>
</tbody>
</table>

- **MPI reference:** [http://www.mcs.anl.gov/mpi/](http://www.mcs.anl.gov/mpi/)
Message-orientated Persistent Communication

- Message queuing systems or Message-Oriented Middleware (MOM)
  - 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
  - When/if the message will be read?
• General architecture of MOM
The general organization of a message-queuing system with routers
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

• Stream oriented comm is required!
A data stream is a sequence of data units.

Discrete or continuous:
- Discrete stream
- Continuous stream

For continuous stream, three transmission modes:
- Asynchronous transmission mode
  - No timing requirements
- Synchronous transmission mode
  - Maximum end-to-end delay
- Isochronous transmission mode
  - Both minimum and maximum end-to-end delay
Set Up Stream

(a) Sending process

(b) Camera

Stream

Network

Stream

Network

OS

Program

Receiving process

Display

OS

Camera

Display

OS
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
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
### Specify QoS: Flow Specification

<table>
<thead>
<tr>
<th>Characteristics of the Input</th>
<th>Service Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>maximum data unit size (bytes)</td>
<td>Loss sensitivity (bytes)</td>
</tr>
<tr>
<td>Token bucket rate (bytes/sec)</td>
<td>Loss interval (μsec)</td>
</tr>
<tr>
<td>Toke bucket size (bytes)</td>
<td>Burst loss sensitivity (data units)</td>
</tr>
<tr>
<td>Maximum transmission rate (bytes/sec)</td>
<td>Minimum delay noticed (μsec)</td>
</tr>
<tr>
<td></td>
<td>Maximum delay variation (μsec)</td>
</tr>
<tr>
<td></td>
<td>Quality of guarantee</td>
</tr>
</tbody>
</table>
QoS: Set Up Stream

• Lack of a model
  – Specify QoS parameter
  – Generically describe resources in any communication system
  – Translate QoS parameters to resource usage

• Expressing and establishing QoS is often difficult

• Incompatible approaches exist
Stream Synchronization

• Deal with maintaining temporal relations between streams

• Example:
  – A slide show on the Web that has been enhanced with audio
  – A movie play

• Two issues:
  – Synchronization mechanism
  – The distribution of synchronization mechanisms
Synchronization Mechanisms

Procedure that reads two audio data units for each video data unit

Incoming stream

Network

Receiver's machine

Application

OS

Multimedia control is part of middleware

Middleware layer

Incoming stream

Network

Application tells middleware what to do with incoming streams

CS550: Advanced Operating Systems
• Whether synchronization should take place at the sending or receiving side?

• What is the local synchronization specification?
Summary

• **Message-oriented communication**
  – Persistence and synchronicity
  – Message-oriented transient communication
    • Berkeley socket
    • MPI
  – Message-oriented persistent communication

• **Stream-oriented communication**
  – Data stream
  – Quality of services
  – Stream synchronization

• **Readings:**
  – Chpt 4 of AST