Distributed Systems
Communication (II)

[2] Communication (II)
1. Layered Protocols
2. Remote Procedure Call
3. Remote Object Invocation
4. Message-oriented Communication
5. Stream-oriented Communication

[3] Persistence and Synchronicity in Communication (1)
Assumption – communication system organized as follows:

- applications are executed on hosts,
- each host connected to one communication server,
- buffers may be placed either on hosts or in the communication servers of the underlying network,
- example: an e-mail system.

persistent vs transient communication,

asynchronous communication – sender continues immediately after it has submitted its message for transmission,

synchronous communication – the sender blocked until its message is stored in a local buffer at the receiving host or actually delivered to the receiver.

[4] Persistence and Synchronicity in Communication (2)
Client/server computing generally based on a model of synchronous communication:

- client and server to be active at the time of communication,
- client issues request and blocks until reply received,
server essentially waits only for incoming requests and subsequently processes them.

Drawbacks of synchronous communication:

- client cannot do any other work while waiting for reply,
- failures to be dealt with immediately (the client is waiting),
- in many cases the model simply not appropriate (mail, news).

[5] Persistence and Synchronicity in Communication (3)

General organization of a communication system in which hosts are connected through a network.

- queued messages sent among processes,
- sender not stopped in waiting for immediate reply,
- fault tolerance often ensured by middleware.

[6] Persistence and Synchronicity in Communication (4)

Persistent vs. transient communication

Persistent communication
A message is stored at a communication server as long as it takes to deliver it at the receiver.

Transient communication
A message is discarded by a communication server as soon as it cannot be delivered at the next server or at the receiver.

[7] Persistence and Synchronicity in Communication (5)
Persistent communication of letters back in the days of the Pony Express.

[8] **Persistence and Synchronicity in Communication (6)**

Different forms of communication:

a. persistent asynchronous,

b. persistent synchronous,

c. transient asynchronous,

d. receipt-based transient synchronous,

e. delivery-based transient synchronous,

f. response-based transient synchronous,

[9] **Message-Oriented Transient Communication**

- socket interface introduced in Berkeley UNIX,
another transport layer interface: XTI, X/Open Transport Interface, formerly called the Transport Layer Interface (TLI), developed by AT&T

**socket**
Communication endpoint to which an application write data that are to be sent over the underlying network and from which incoming data can be read.

[10] **Berkeley Sockets (1)**

<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>

Socket primitives for TCP/IP.


Connection-oriented communication pattern using sockets.

[12] **The Message-Passing Interface (MPI) (1)**
**MPI**
Group of message-oriented primitives that would allow developers to easily write highly efficient applications.

Sockets insufficient because:
- at the wrong level of abstraction supporting only send and receive primitives,
- designed to communicate using general-purpose protocol stacks such as TCP/IP, not suitable in high-speed interconnection networks, such as those used in COWs and MPPs (with different forms of buffering and synchronization).


MPI assumptions:
- communication within a known group of processes,
- each group with assigned id,
- each process within a group also with assigned id,
- all serious failures (process crashes, network partitions) assumed as fatal and without any recovery,
- a (groupID, processID) pair used to identify source and destination of the message,
- only receipt-based transient synchronous communication (d) not supported, other supported.

[14] **The Message-Passing Interface (3)**

<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_issend</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_irecv</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 is none</td>
</tr>
<tr>
<td>MPI_irecv</td>
<td>Check if there is an incoming message, but do not block</td>
</tr>
</tbody>
</table>
Some of the most intuitive message-passing primitives of MPI.


Message-queueing systems = Message-Oriented Middleware (MOM)

The essence of MOM systems:

- offer the intermediate-term storage capacity for messages,
- target to support message transfers that are allowed to take minutes instead of seconds or milliseconds,
- no guarantees about when or even if the message will be actually read,
- the sender and receiver can execute completely independently.

[16] **Message-Queuing Model**

<table>
<thead>
<tr>
<th>Primitive</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Put</td>
<td>Append a message to a specified queue</td>
</tr>
<tr>
<td>Get</td>
<td>Block until the specified queue is nonempty, and remove the first message</td>
</tr>
<tr>
<td>Poll</td>
<td>Check a specified queue for messages, and remove the first. Never block</td>
</tr>
<tr>
<td>Notify</td>
<td>Install a handler to be called when a message is put into the specified queue</td>
</tr>
</tbody>
</table>

Basic interface to a queue in a message-queueing system.

Most queuing systems also allow a process to install handlers as callback functions.

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

*source queue, destination queue, a database of queue names to network locations mapping.*


The general organization of a message-queuing system with routers:

– may grow into overlay network,

– may need dynamic routing schemes.

Queue managers:

– normally interact directly with applications,

– some operate as routers or relays.

[19] **Message Brokers**
The general organization of a message broker in a message-queuing system.

**Message broker**
Acts as an application-level gateway in a message-queuing system. Its main purpose is to convert incoming messages to a format that can be understood by the destination application. It may provide routing capabilities.


- with message brokers it may be necessary to accept a certain loss of information during transformation,
- at the heart of a message broker lies a database of conversion rules,
- general message-queuing systems are not aimed at supporting only end users,
- they are set up to enable persistent communication,
- range of applications:
  - e-mail, workflow, groupware, batch processing,
  - integration of a collection of databases or database applications.

[21] **Example: IBM MQSeries**
General organization of IBM’s MQSeries message-queuing system.

[22] **Channels**

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

Some attributes associated with message channel agents.

[23] **Message Transfer (1)**
The general organization of an MQSeries queuing network using routing tables and aliases. By using logical names, in combination with name resolution to local queues, it is possible to put a message in a remote queue.

[24] **Message Transfer (2)**

<table>
<thead>
<tr>
<th>Primitive</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MQopen</td>
<td>Open a (possibly remote) queue</td>
</tr>
<tr>
<td>MQclose</td>
<td>Close a queue</td>
</tr>
<tr>
<td>MQput</td>
<td>Put a message into an opened queue</td>
</tr>
<tr>
<td>MQget</td>
<td>Get a message from a (local) queue</td>
</tr>
</tbody>
</table>

Primitives available in an IBM MQSeries MQI.

[25] **Stream-Oriented Communication**

- forms of communication in which timing plays a crucial role,
- example:
  - an audio stream built up as a sequence of 16-bit samples each representing the amplitude of the sound wave as it is done through PCM (Pulse Code Modulation),
  - audio stream represents CD quality, i.e. 44100Hz,
  - samples to be played at intervals of exactly 1/44100,
- which facilities a distributed system should offer to exchange time-dependent information such as audio and video streams?
  - support for the exchange of time-dependent information = support for **continuous media**,
  - **continuous** (representation) media vs. **discrete** (representation) media.

[26] **Support for Continuous Media**

In continuous media:
– temporal relationships between data items fundamental to correctly interpreting the data,

– timing is crucial.

**Asynchronous transmission mode**
Data items in a stream are transmitted one after the other, but there are no further timing constraints on when transmission of items should take place.

**Synchronous transmission mode**
Maximum end-to-end delay defined for each unit in a data stream.

**Isochronous transmission mode**
It is necessary that data units are transferred on time. Data transfer is subject to bounded (delay) jitter.

[27] **Data Stream (1)**

\[\text{Diagram}\]

a. Setting up a stream between two processes across a network,

b. Setting up a stream directly between two devices.

– stream sequence of data units, may be considered as a virtual connection between a source and a sink,

– simple stream vs. complex stream (consisting of several related sub-streams).
[28] **Data Stream (2)**

An example of multicasting a stream to several receivers.

- problem with receivers having different requirements with respect to the quality of the stream,
- **filters** to adjust the quality of an incoming stream, differently for outgoing streams.

[29] **Specifying QoS (1)**

<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>Token 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>

A flow specification.

Time-dependent requirements among other **Quality of Service (QoS)** requirements.

[30] **Specifying QoS (2)**
The principle of a token bucket algorithm.

- tokens generated at a constant rate,
- tokens buffered in a bucket which has limited capacity.

[31] **Setting Up a Stream**

The basic organization of RSVP (Resource reSerVation Protocol), transport-level protocol for resource reservation in a distributed system.

[32] **Synchronization Mechanisms (1)**
The principle of explicit synchronization on the level data units. Given a complex stream, how to keep the different substreams in synch?

[33] **Synchronization Mechanisms (2)**

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

Multiplex of all substreams into a single stream and demultiplexing at the receiver. Synchronization is handled at multiplexing/demultiplexing point (MPEG).