Definition of a Distributed System (1)

A distributed system is:

A collection of independent computers that appears to its users as a single coherent system.

Definition of a Distributed System (2)

A distributed system organized as middleware. Note that the middleware layer extends over multiple machines.

Transparency in a Distributed System

<table>
<thead>
<tr>
<th>Transparency</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Access</td>
<td>Hide differences in dispacement and how a resource is accessed</td>
</tr>
<tr>
<td>Location</td>
<td>Hide where a resource is located</td>
</tr>
<tr>
<td>Migration</td>
<td>Hide that a resource may move to another location</td>
</tr>
<tr>
<td>Relocation</td>
<td>Hide that a resource may be moved to another location at will</td>
</tr>
<tr>
<td>Replication</td>
<td>Hide that a resource may be shared by several competitive users</td>
</tr>
<tr>
<td>Concurrency</td>
<td>Hide that a resource may be shared by several competitive users</td>
</tr>
<tr>
<td>Failure</td>
<td>Hide the failure and discovery of a resource</td>
</tr>
<tr>
<td>Persistence</td>
<td>Hide whether a following resource is in memory or on disk</td>
</tr>
</tbody>
</table>

Different forms of transparency in a distributed system.

Scalability Problems

<table>
<thead>
<tr>
<th>Concept</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Centralized services</td>
<td>A single server for all users</td>
</tr>
<tr>
<td>Centralized data</td>
<td>A single online telephone book</td>
</tr>
<tr>
<td>Centralized algorithms</td>
<td>Routing based on complete information</td>
</tr>
</tbody>
</table>

Examples of scalability limitations.
Scaling Techniques (1)

The difference between letting:

b) a server or
c) a client check forms as they are being filled

Positioning Middleware

General structure of a distributed system as middleware.

Comparison between Systems

A comparison between multiprocessor operating systems, multicomputer operating systems, network operating systems, and middleware based distributed systems.

Middleware and Openness

In an open middleware-based distributed system, the protocols used by each middleware layer should be the same, as well as the interfaces they offer to applications.

Clients and Servers

General interaction between a client and a server.
An Example Client and Server (1)

The `header.h` file used by the client and server.

An Example Client and Server (2)

A sample server.

An Example Client and Server (3)

A client using the server to copy a file.

Processing Level

The general organization of an Internet search engine into three different layers.

Multitiered Architectures (1)

Alternative client-server organizations (a) – (e).

Multitiered Architectures (2)

An example of a server acting as a client.
Modern Architectures

An example of horizontal distribution of a Web service.

Communication Methods

Chapter 2

Layered Protocols (1)

Layers, interfaces, and protocols in the OSI model.

Layered Protocols (2)

A typical message as it appears on the network.

Data Link Layer

Discussion between a receiver and a sender in the data link layer.

Client-Server TCP

(a) Normal operation of TCP.
(b) Transactional TCP.
Middleware Protocols

Conventional Procedure Call

Client and Server Stubs

Steps of a Remote Procedure Call
1. Client procedure calls client stub in normal way
2. Client stub builds message, calls local OS
3. Client's OS sends message to remote OS
4. Remote OS gives message to server stub
5. Server stub unpacks parameters, calls server
6. Server does work, returns result to the stub
7. Server stub packs it in message, calls local OS
8. Server's OS sends message to client's OS
9. Client's OS gives message to client stub
10. Stub unpacks result, returns to client

Passing Value Parameters (1)

Passing Value Parameters (2)

Steps involved in doing remote computation through RPC.

a) Original message on the Pentium
b) The message after receipt on the SPARC
c) The message after being inverted. The little numbers in boxes indicate the address of each byte
Parameter Specification and Stub Generation

a) A procedure

```c
foobar [char x, float y, int z[]]
{
    ...
}
```

b) The corresponding message.

Doors

The principle of using doors as IPC mechanism.

Asynchronous RPC (1)

a) The interconnection between client and server in a traditional RPC

b) The interaction using asynchronous RPC

Asynchronous RPC (2)

A client and server interacting through two asynchronous RPCs

Writing a Client and a Server

The steps in writing a client and a server in DCE RPC

Binding a Client to a Server

Client-to-server binding in DCE
Distributed Objects

Common organization of a remote object with client-side proxy.

Parameter Passing

The situation when passing an object by reference or by value.

Persistence and Synchronicity in Communication (1)

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

Persistence and Synchronicity in Communication (2)

Persistent communication of letters back in the days of the Pony Express.
Persistence and Synchronicity in Communication (3)

a) Persistent asynchronous communication
b) Persistent synchronous communication

Persistence and Synchronicity in Communication (4)

a) Transient asynchronous communication
b) Receipt-based transient synchronous communication

Persistence and Synchronicity in Communication (5)

a) Delivery-based transient synchronous communication at message delivery
b) Response-based transient synchronous communication

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 connection</td>
</tr>
<tr>
<td>Accept</td>
<td>Block caller until connection request arrives</td>
</tr>
<tr>
<td>Connect</td>
<td>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>Close the connection</td>
</tr>
</tbody>
</table>

Socket primitives for TCP/IP.

Berkeley Sockets (2)

Connection-oriented communication pattern using sockets.

The Message-Passing Interface (MPI)

<table>
<thead>
<tr>
<th>Primitive</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MPI-send</td>
<td>Send a message and wait until copied to local or remote buffer</td>
</tr>
<tr>
<td>MPI_sendv</td>
<td>Send a message and wait until receipt state</td>
</tr>
<tr>
<td>MPI_numeric</td>
<td>Send a message and wait for reply</td>
</tr>
<tr>
<td>MPI_send</td>
<td>Pass reference to outgoing message, and continue</td>
</tr>
<tr>
<td>MPI_sendv</td>
<td>Pass reference to outgoing message, and wait until receipt state</td>
</tr>
<tr>
<td>MPI_recvc</td>
<td>Receive a message, block if there are none</td>
</tr>
<tr>
<td>MPI_test</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-Queuing Model (1)

Four combinations for loosely-coupled communications using queues.

General Architecture of a Message-Queuing System (1)

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

Message Brokers

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

Message-Queuing Model (2)

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

General Architecture of a Message-Queuing System (2)

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

Example: IBM MQSeries

General organization of IBM’s MQSeries message-queuing system.
### Channels

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transport type</td>
<td>Determine 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>Selection count</td>
<td>Specifies the maximum number of retries to start up the remote MCA</td>
</tr>
<tr>
<td>Failover address</td>
<td>Maximum number of MCA entries to use failed messages into a failover queue</td>
</tr>
</tbody>
</table>

Some attributes associated with message channel agents.

### Message Transfer (1)

The general organization of an MQSeries queuing network using routing tables and aliases.

### 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