Electronic Payment Systems (1)

- Payment systems based on direct payment between customer and merchant.
  a) Paying in cash.
  b) Using a check.
  c) Using a credit card.

E-cash

- The principle of anonymous electronic cash using blind signatures.
Today: Distributed Middleware

• Middleware concepts

• Case study: CORBA

Middleware

• Software layer between application and the OS
  – Provides useful services to the application
  – Abstracts out common functionality required by distributed applications
  –Applications use the middleware API to invoke services

• Examples:
  – CORBA
  – DCOM
Overview of CORBA

- **Common Object Request Broker Architecture**
  - Specification of a distributed middleware
  - Specs drawn up by Object Management Group (OMG)
  - [http://www.omg.org](http://www.omg.org)
- **Goal**: Interoperability with distributed applications on various platforms

CORBA Overview

- **Object request broker (ORB)**
  - Core of the middleware platform
  - Handles communication between objects and clients
  - Handles distribution and heterogeneity issues
  - May be implemented as libraries
- **Facilities**: composition of CORBA services
Object Model

- Objects & services specified using an Interface Definition language (IDL)
  - Used to specify interface of objects and/or services
- ORB: run-time system that handles object-client communication
- Dynamic invocation interface: allows object invocation at run-time
  - Generic invoke operation: takes object reference as input
  - Interface repository stores all interface definitions

CORBA Services

- Collection service: group objects into lists, queues,..
- Query service: use query language to query for service(s)
- Concurrency control service: locking services
- Event service: interrupt upon a specific event
- Many more…
Corba Services

<table>
<thead>
<tr>
<th>Service</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Collection</td>
<td>Facilities for grouping objects into lists, queue, sets, etc.</td>
</tr>
<tr>
<td>Query</td>
<td>Facilities for querying collections of objects in a declarative manner</td>
</tr>
<tr>
<td>Concurrency</td>
<td>Facilities to allow concurrent access to shared objects</td>
</tr>
<tr>
<td>Transaction</td>
<td>Facilities for asynchronous communication over multiple objects</td>
</tr>
<tr>
<td>Event</td>
<td>Facilities for asynchronous communication through events</td>
</tr>
<tr>
<td>Notification</td>
<td>Advanced facilities for event-based asynchronous communication</td>
</tr>
<tr>
<td>Externalization</td>
<td>Facilities for marshaling and unmarshaling of objects</td>
</tr>
<tr>
<td>Life cycle</td>
<td>Facilities for creation, deletion, copying, and moving of objects</td>
</tr>
<tr>
<td>Licensing</td>
<td>Facilities for attaching a license to an object</td>
</tr>
<tr>
<td>Naming</td>
<td>Facilities for systemwide name of objects</td>
</tr>
<tr>
<td>Property</td>
<td>Facilities for associating (attribute, value) pairs with objects</td>
</tr>
<tr>
<td>Trading</td>
<td>Facilities to publish and find the services on object to offer</td>
</tr>
<tr>
<td>Persistence</td>
<td>Facilities for persistently storing objects</td>
</tr>
<tr>
<td>Relationship</td>
<td>Facilities for expressing relationships between objects</td>
</tr>
<tr>
<td>Security</td>
<td>Mechanisms for secure channels, authorization, and auditing</td>
</tr>
<tr>
<td>Time</td>
<td>Provides the current time within specified error margins</td>
</tr>
</tbody>
</table>

Object Invocation Models

<table>
<thead>
<tr>
<th>Request type</th>
<th>Failure semantics</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Synchronous</td>
<td>At-most-once</td>
<td>Caller blocks until a response is returned or an exception is raised</td>
</tr>
<tr>
<td>One-way</td>
<td>Best effort delivery</td>
<td>Caller continues immediately without waiting for any response from the server</td>
</tr>
<tr>
<td>Deferred synchronous</td>
<td>At-most-once</td>
<td>Caller continues immediately and can later block until response is delivered</td>
</tr>
</tbody>
</table>

- Invocation models supported in CORBA.
  - Original model was RMI/RPC-like
  - Current CORBA versions support additional semantics
Event and Notification Services (1)

- The logical organization of suppliers and consumers of events, following the push-style model.

Event and Notification Services (2)

- The pull-style model for event delivery in CORBA.
Messaging: Async. Method Invocation

- CORBA's callback model for asynchronous method invocation.

**Diagram:**

1. Call by the application
2. Request to server
3. Response from server
4. Call by the ORB

Messaging (2)

- CORBA's polling model for asynchronous method invocation.

**Diagram:**

1. Call by the application
2. Request to server
3. Response from server
4. Call by the application
Portable Object Adaptor (1)

- POA: Wrappers for server-side code (makes code look like objects)
  a) The POA supports multiple servants.
  b) The POA supports a single servant.

Portable Object Adaptor (2)

```
My_servant *my_object;    // Declare a reference to a C++ object
CORBA::Objectid_var oid;  // Declare a CORBA identifier
my_object = new MyServant;  // Create a new C++ object
oid = poa ->activate_object (my_object);  // Register C++ object as CORBA OBJECT
```

- Changing a C++ object into a CORBA object.
Naming: Object References

- Interoperable object reference: language-independent techniques for referring to objects

An Example Architecture

- An example architecture of a fault-tolerant CORBA system.