Distributed Object-Based Systems
The WWW Architecture
Web Services
Handout 11 Part(a)

EECS 591
Farnam Jahanian
University of Michigan

Reading List
Remote Object Invocation -- Tanenbaum Chapter 2.3
CORBA -- Tanenbaum Chapter 9.1
DCOM -- Tanenbaum 9.2 (optional)
The Web architecture – Tanenbaum 11.1 (pp. 647-662)
The History of Universe Part I

- TREND #1: Shift from process-oriented programming to object-oriented programming in the mid to late 80’s
  - TREND #2: client-server computing
    - introduction of RPC
    - DCE (early middleware and run-time environment) proposed to standardize distributed computing on client-server model → DCE failed but C/S model and object-orientation lived on
  - TREND #3: TREND 1 and 2 came together.
    - UNIX community (and the rest of the world except MSFT) community proposed CORBA
    - Microsoft proposed DCOM and ActiveX
    - Both approaches continue to have followers
- TREND #4: WWW --- the mother of all client-server applications
  - WWW became the most popular document service known to mankind in the 90’s
  - Client-server model lives on … client-side and server-side scripts
  - CGI scripts enable execution of programs on servers (taking user data as input – HTML form) → program may fetch a document, or manipulate data from a DB to generate results, and send the document to the client on-the-fly to generate a document.

- TREND #5: Advances in OO programming languages and run-time environments
  - SUN introduced JAVA (MSFT later promoted C# after Java took off)
- TREND #6: toward network computing
  - server doesn’t have to do everything → first client-side scripts, then applets
  - it is possible to pass a pre-compiled program to the client to be executed in the browser address space (applets)
  - Servlet: is a pre-compiled program executing in the address space of the (web) server. (note: CGI scripts are executed as a separate process.)
- TREND #7: The WEB grows up → web services
  - XML introduced to define new documents and structure documents
  - A Web Service is just an object accessible over a network
  - Two competing models:
    - .NET
The World Wide Web

Overall organization of the Web.

WWW Architectural Overview (1)

The principle of using server-side CGI programs.
Distributed Objects

- **IDEA:**
  - Treat everything as objects
  - Services and resources are made available as (remote) objects to clients
  - An powerful paradigm for distributed computing
  - Attractive because easy to hide distribution behind object’s interface
  - Two widely-deployed middleware for distributed object-based systems:
    - CORBA (industry-standard)
    - DCOM (Microsoft)
Logical View of Distributed Objects

- Objects are physically distributed across the network
- Services and resources are accessed by invoking methods on remote objects

Remote Object Invocation

- Objects:
  - **State** is the data encapsulated by the object
  - **Methods** are the operations allowed on the data through an interface
  - Separation between interface and object implementation is very desirable for distributed systems → put the interface on one machine (client) and the object itself on another machine (server)
  - When a client binds to an object, a **proxy** stub (implementation of object’s interface) is loaded into the client’s address space.
  - Incoming requests are handled by a server stub (called a **skeleton**) which invokes the proper method.
  - Proxy and skeleton are responsible for marshaling and unmarshaling, and client-server communication.
Remote Object Invocation

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

Binding to an Object

- Distributed objects provide system-wide object references (contrast to RPC)
- Binding results in a proxy being placed the process’ address space
- Implicit binding: directly invoke methods using a ref to an object
- Explicit binding: client first calls a special function to bind before invoking a method
Binding a Client to an Object

(a) Example with implicit binding using only global references

```
Distr_object* obj_ref; //Declare a systemwide object reference
obj_ref = ...; // Initialize the reference to a distributed object
obj_ref-> do_something(); // Implicitly bind and invoke a method
```

(b) Example with explicit binding using global and local references

```
Distr_object objPref; //Declare a systemwide object reference
Local_object* obj_ptr; //Declare a pointer to local objects
obj_ref = ...; //Initialize the reference to a distributed object
obj_ptr = bind(obj_ref); //Explicitly bind and obtain a pointer to the local proxy
obj_ptr -> do_something(); //Invoke a method on the local proxy
```

a) (a) Example with implicit binding using only global references
b) (b) Example with explicit binding using global and local references

Parameter Passing

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

- Common Object Request Broker Architecture from OMG (a consortium of 800 companies!)
- An open standard for application interoperability based on the concept of distributed objects.
- Supports access to resources and services across heterogeneous languages, platforms and networks.

Overview of CORBA

- Four architectural elements of CORBA:
  - ORB: forms the core of CORBA, responsible for handling communication between clients and objects while hiding distribution and heterogeneity
  - Horizontal facilities: general-purpose high-level services at the application level, e.g., user interface, task management, system management, information management
  - Vertical facilities: high-level services targeted to specific application domains such as e-commerce, banking, manufacturing, …
  - Common object services: basic application-independent services similar to common services provided by OS (later)
Remote Object Model

- Objects are services are specified in CORBA Interface Definition Language – syntax for expressing methods and parameters
- Statically-defined interface (compiled) vs. Dynamic invocation interface (run-time)

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>Flat and nested transactions on method calls 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) pair with objects</td>
</tr>
<tr>
<td>Trading</td>
<td>Facilities to publish and find the services on object has 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 with specified error margins</td>
</tr>
</tbody>
</table>

Overview of CORBA services.
All services specified in CORBA IDL.
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.

Event and Notification Services (1)

A service that signals the occurrence of an event

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 (1)

CORBA's callback model for asynchronous method invocation.
Messaging (2)

CORBA's polling model for asynchronous method invocation.

ORB Interoperability

<table>
<thead>
<tr>
<th>Message type</th>
<th>Originator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Request</td>
<td>Client</td>
<td>Contains an invocation request</td>
</tr>
<tr>
<td>Reply</td>
<td>Server</td>
<td>Contains the response to an invocation</td>
</tr>
<tr>
<td>LocateRequest</td>
<td>Client</td>
<td>Contains a request on the exact location of an object</td>
</tr>
<tr>
<td>LocateReply</td>
<td>Server</td>
<td>Contains location information on an object</td>
</tr>
<tr>
<td>CancelRequest</td>
<td>Client</td>
<td>Indicates client no longer expects a reply</td>
</tr>
<tr>
<td>CloseConnection</td>
<td>Both</td>
<td>Indication that connection will be closed</td>
</tr>
<tr>
<td>MessageError</td>
<td>Both</td>
<td>Contains information on an error</td>
</tr>
<tr>
<td>Fragment</td>
<td>Both</td>
<td>Part (fragment) of a larger message</td>
</tr>
</tbody>
</table>

GIOP (General Inter-ORB Protocol): a standard for inter ORB communication
Realization on TCP is called Internet Inter-ORB Protocol (IIOP)
Portable Object Adaptor

(a)

(b)

Mapping of CORBA object identifiers to servants.

a) The POA supports multiple servants.
b) The POA supports a single servant.

Object References (1)

The organization of an IOR with specific information for IIOP.
Object References (2)

Indirect binding in CORBA.

Object Groups

A possible organization of an IOGR for an object group having a primary and backups.
An example architecture of a fault-tolerant CORBA system.

Random Notes

- Two types of processes: clients & servers
- Persistent msg. queue implemented as a messaging service
- Object adapter (wrappers) is a mechanism a specific invocation methods for a group of objects
- Portable object adaptors & servants
- Agent system
- Two types of naming service
- No generic caching or replication service – pass to application
- CORBA v.3 explicitly support object groups (replicated copies)
### Summary (section 9.4)

<table>
<thead>
<tr>
<th>Issue</th>
<th>CORBA</th>
<th>DCOM</th>
<th>Globe</th>
</tr>
</thead>
<tbody>
<tr>
<td>Design goals</td>
<td>Interoperability</td>
<td>Functionality</td>
<td>Scalability</td>
</tr>
<tr>
<td>Object model</td>
<td>Remote objects</td>
<td>Remote objects</td>
<td>Distributed objects</td>
</tr>
<tr>
<td>Services</td>
<td>Many of its own</td>
<td>From environment</td>
<td>Few</td>
</tr>
<tr>
<td>Interfaces</td>
<td>IDL based</td>
<td>Binary</td>
<td>Binary</td>
</tr>
<tr>
<td>Sync. communication</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Async. communication</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Callbacks</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Events</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Messaging</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Object server</td>
<td>Flexible (POA)</td>
<td>Hard-coded</td>
<td>Object dependent</td>
</tr>
<tr>
<td>Directory service</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Trading service</td>
<td>yes</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

Continued …

Comparison of CORBA, DCOM, and Globe.

### Summary

<table>
<thead>
<tr>
<th>Issue</th>
<th>CORBA</th>
<th>DCOM</th>
<th>Globe</th>
</tr>
</thead>
<tbody>
<tr>
<td>Naming service</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Location service</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Object reference</td>
<td>Object's location</td>
<td>Interface pointer</td>
<td>True identifier</td>
</tr>
<tr>
<td>Synchronization</td>
<td>Transactions</td>
<td>Transactions</td>
<td>Only intra-object</td>
</tr>
<tr>
<td>Replication support</td>
<td>Separate server</td>
<td>None</td>
<td>Separate subobject</td>
</tr>
<tr>
<td>Transactions</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Fault tolerance</td>
<td>By replication</td>
<td>By transactions</td>
<td>By replication</td>
</tr>
<tr>
<td>Recovery support</td>
<td>Yes</td>
<td>By transactions</td>
<td>No</td>
</tr>
<tr>
<td>Security</td>
<td>Various mechanisms</td>
<td>Various mechanisms</td>
<td>More work needed</td>
</tr>
</tbody>
</table>

Comparison of CORBA, DCOM, and Globe.