Overview

- CORBA
  - Common Object Request Broker Architecture
  - Industry Standard, available since 1990
  - Still in development
  - Object Management Group (OMG)
  - Designed by committee: features and facilities galore

Distributed Objects

- Easy to hide distribution aspects behind an interface
- An object can be anything
- Powerful paradigm for building systems

Structure of this chapter

- Architecture overview
- Communication
- Processes
- Naming
- Synchronization
- Caching and Replication
- Fault Tolerance
- Security

Overview of CORBA

The global architecture of CORBA.
Overview of CORBA

- Object Request Broker
  - core of any CORBA distribution
  - enabling communication
  - hiding distribution and heterogeneity
  - implemented as libraries
- CORBA facilities
  - horizontal facilities
  - vertical facilities

Horizontal Facilities

- General-purpose, high-level services
- Independent of application domains
- Examples
  - for user interfaces
  - information management
  - system management,
  - task management.

Vertical Facilities

- High-level facilities
- Targeted to a specific application domain
- Examples
  - electronic commerce
  - banking
  - manufacturing
  - etc.

Object Model

- Remote-object model is used
- Implementation of object resides in address space of server
- Objects and services are specified exclusively in the CORBA IDL
- No semantic description possible

Interfaces

- Interface: collection of methods
- Objects specify which interfaces they implement
- No binary interfaces - only IDL
- IDL mapping rules necessary.
- They exist for C, C++, Java, Smalltalk, Ada, and COBOL

The general organization of a CORBA system.
Client side

- ORB offers
  - basic communication
  - manipulating object references
  - unmarshaling and comparison operations
  - locating services (e.g., naming)
- Static invocation: IDL proxy
- Dynamic invocation (genericinvoke())
- DII - Dynamic Invocation Interface

Server Side

- Object Implementation
- ORB offers complimentary services
- Object adaptor
  - forwards incoming requests to the right object
- Static invocation: IDL skeleton
- Dynamic invocation interface

Dynamic Invocation

- Interface repository stores all interface definitions, incl. dynamic ones
- Repository ID assigned to every interface (not necessarily unique)
- Implementation repository highly platform dependent

CORBA Services

- General-purpose services
- Independent of applications
- Resemblance to typical OS services
- Partially overlapping
- Life cycle service: factory objects

Corba Services

<table>
<thead>
<tr>
<th>Service</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Security</td>
<td>Functions for encrypting data to the user, server, and so on</td>
</tr>
<tr>
<td>Identity</td>
<td>Functions for establishing identity and verifying signatures</td>
</tr>
<tr>
<td>Transaction</td>
<td>Functions for establishing transactions or aborting transactions</td>
</tr>
<tr>
<td>Naming</td>
<td>Functions for organizing names or objects</td>
</tr>
<tr>
<td>Reporting</td>
<td>Functions for reporting events or services</td>
</tr>
<tr>
<td>Naming</td>
<td>Functions for organizing names or objects</td>
</tr>
<tr>
<td>Encapsulation</td>
<td>Functions for encrypting data to the user, server, and so on</td>
</tr>
</tbody>
</table>

Communication

- Originally: client-server
- ...too simple...
- therefore, several communication facilities/object invocation models were added.

Overview of CORBA services.
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>All-responses</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>All-responses</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

- Method invocation models not deemed sufficient.
- Event service: Events are...
  - produced by supplier,
  - delivered through an event channel, and
  - consumed by consumer.
- Push vs. pull model.

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

- Calculate \( k = i + j \)
- \( \text{int add}(\text{in int } i, \text{ in int } j, \text{ out int } k) \)
- Callback model
- Polling model
  - \( \text{void sendcb}\_\text{add}(\text{in int } i, \text{ in int } j) \)

Callback model

\( \text{int add}(\text{in int } i, \text{ in int } j, \text{ out int } k) \)

- Called by client:
  - \( \text{void sendcb}\_\text{add}(\text{in int } i, \text{ in int } j) \)
- Called by client's ORB
  - \( \text{void replycb}\_\text{add}(\text{in int } \text{ret_val, in int } k) \)
Messaging (1)

CORBA's callback model for asynchronous method invocation.

Polling model

```c
int add(int i, int j, out int k);
```

- Called by client:
  ```c
  void sendpoll_add(int i, int j);
  ```
- Also called by client:
  ```c
  void replypoll_add(out int ret_val, out int k);
  ```

Messaging (2)

CORBA's polling model for asynchronous method invocation.

Interoperability

- Initially: lack of communication specification
- GIOP - General Inter-ORB Protocol
- IIOP - Internet Inter-ORB Protocol
  surprise... it's TCP!

GIOP message types.

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>Results</td>
<td>Server</td>
<td>Contains the response to an invocation</td>
</tr>
<tr>
<td>LocateRequest</td>
<td>Client</td>
<td>Contains a request to locate 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>ClientConnectionFault</td>
<td>Client</td>
<td>Indicates that connection will be closed</td>
</tr>
<tr>
<td>MessageError</td>
<td>Client</td>
<td>Contains information on an error</td>
</tr>
<tr>
<td>Fragment</td>
<td>Client</td>
<td>Part (fragment) of a per message</td>
</tr>
</tbody>
</table>

Processes

- Two process types: clients and servers
- CORBA design goal:
  Make the client as simple as possible
- Focus on
  - clients
  - portable object adaptors
  - agents
Clients

- IDL only generates stubs
- Choice of ORB limits system behavior
- Ugly hacks:
  - instruct developer to use specific IDL compiler
  - provide client proxy with implementation
  - Restrict system to specific ORB(s)
- There is a need for a mechanism to use IDL generated proxies in combination with existing client-side ORB.

Interceptors

- Mechanism...
- ...by which an invocation can be intercepted on its way from client to server...
- ...and adapted as necessary.
- Request and response can be adapted
- Request-level interceptor
- Message-level interceptor

Portable Object Adaptors (POA)

- **Mechanism** that implements a specific activation **policy** for a group of objects.
- Adaptors are also called wrappers
- POA: makes server-side code appear to clients as CORBA objects
- Server-side code is ORB independent
- **Servant**: method implementation that client can invoke (prog.language dependent.)

POA (cont.)

- Each POA offers:
  - `ObjectId activate_object(in Servant p_servant);`
  - Active Object Map
- Assume: `my_servant` is subclass of `ServantBase`.

Mapping of CORBA object identifiers to servants.

b) The POA supports multiple servants.

c) The POA supports a single servant.
Portable Object Adaptor (2)

Changing a C++ object into a CORBA object.

Agents

- Agent systems: platform that allows the creation, execution, transfer, and termination of agents.
- Places, systems, regions
- Each region has a finder

Naming

- Object references
- Location of objects based on associated properties
- Process-dependent (local) representation vs. process-independent representation
- Interoperable Object Reference (IOR): Language- and ORB-independent representation of object references.

Object References (1)

The organization of an IOR with specific information for IIOP.

Object References (2)

Indirect binding in CORBA.
CORBA Naming Service

- Name in CORBA: \( \langle \text{id}, \text{kind} \rangle \)
- Naming context - similar to directory concept
- Naming graphs
- No root context - instead initial naming context.

Synchronization

- Provided through two CORBA services:
  - Concurrency Control Service and
  - Transaction Service
- Distributed and nested transaction, using two-phase locking
- Recoverable objects: abort possible (transaction can be rolled back)
- Transactional objects (transaction cannot be rolled back)

Caching and Replication

- CORBA offers no generic support for caching or replication.
- Burden on application developer
- Workaround usually involves interceptors
- Example system: CASCADE
  - DCS - Domain Caching Server
  - Client-side consistency models

Fault Tolerance

- Objects are replicated into object groups, identical copies of the same object
- Replication strategies:
  - E.g., primary-backup, active replication, quorum-based replication, etc.
- Interoperable Object Group Reference (IOGR)

Caching and Replication

The (simplified) organization of a DCS.

Object Groups

A possible organization of an IOGR for an object group having a primary and backups.
Security

- Added late to CORBA (version 2.4)
- Which services (i.e., mechanisms) and which policies?
- Services need to be added at different points in time and space:
  - When: binding time, invocation time, ...
  - Where: application level, inside the ORB, during message transfer, ...

Security (cont.)

- Secure object invocation: usually transparent to client application
- Client can specify security policy requirements to be taken into account:
  - Policy objects
    - client-specific policy objects
    - object-specific policy objects
  - Implementation through... interceptors

Security (1)

The general organization for secure object invocation in CORBA.

Security (2)

The role of security interceptors in CORBA.