

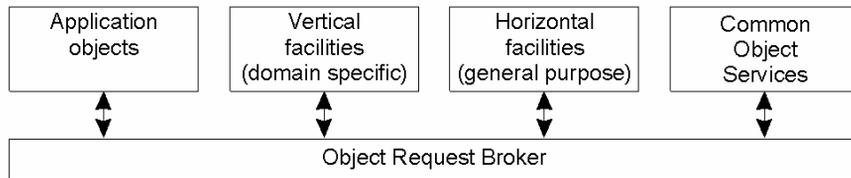
Today: Distributed Middleware

- Middleware concepts
- Case study: CORBA

Middleware

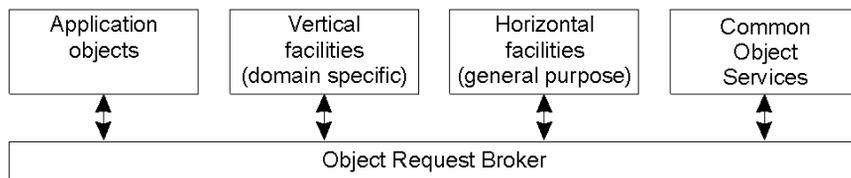
- Software layer between application and the OS
 - Provides useful services to the application
 - Abstracts out common functionality required by distributed applications
 - Applications use the middleware API to invoke services
- Examples:
 - CORBA
 - DCOM

Overview of CORBA



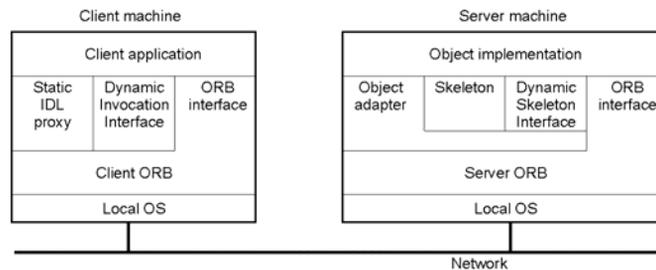
- Common Object Request Broker Architecture
 - Specification of a distributed middleware
 - Specs drawn up by Object Management Group (OMG)
 - <http://www.omg.org>
- Goal: Interoperability with distributed applications on various platforms

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

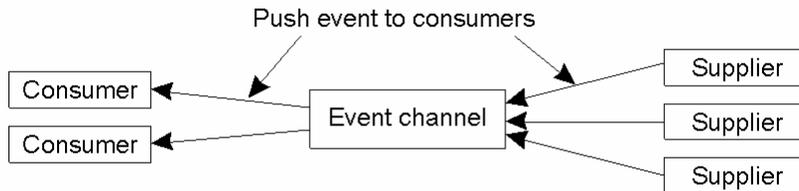
Service	Description
Collection	Facilities for grouping objects into lists, queue, sets, etc.
Query	Facilities for querying collections of objects in a declarative manner
Concurrency	Facilities to allow concurrent access to shared objects
Transaction	Flat and nested transactions on method calls over multiple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

Object Invocation Models

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

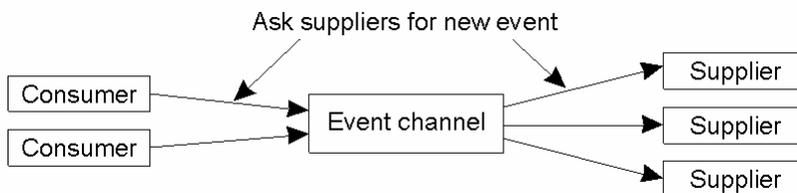
- Invocation models supported in CORBA.
 - Original model was RMI/RPC-like
 - Current CORBA versions support additional semantics

Event and Notification Services (1)



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

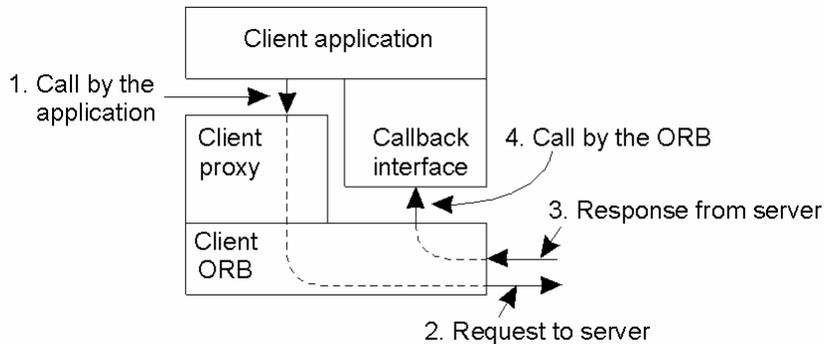
Event and Notification Services (2)



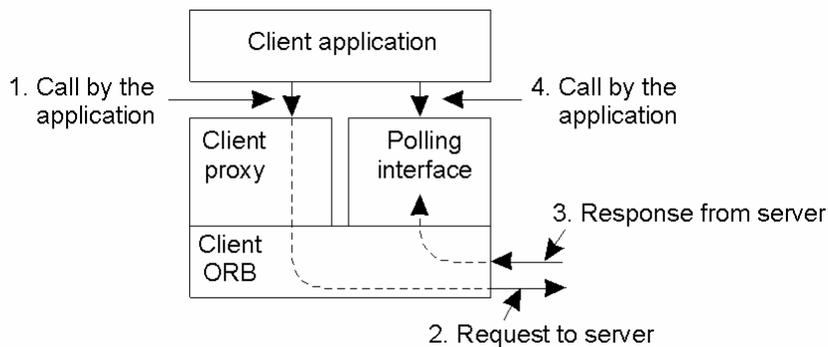
- The pull-style model for event delivery in CORBA.

Messaging: Async. Method Invocation

- CORBA's callback model for asynchronous method invocation.

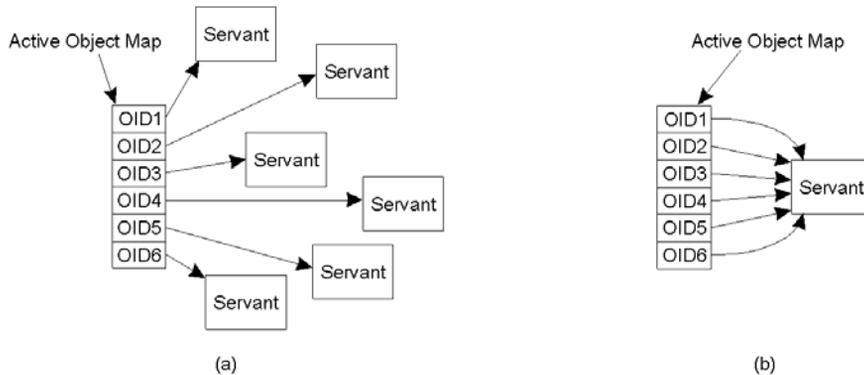


Messaging (2)



- CORBA'S polling model for asynchronous method invocation.

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.

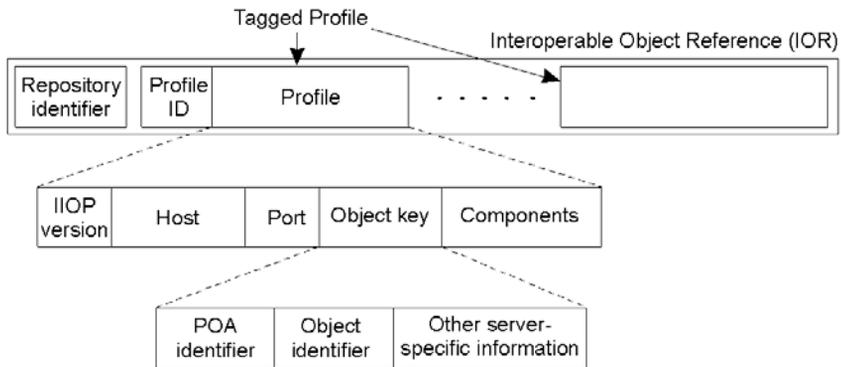
Portable Object Adaptor (2)

```
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.

