

CSC 546: Client/Server Fundamentals

Fall 2000

Communicating objects

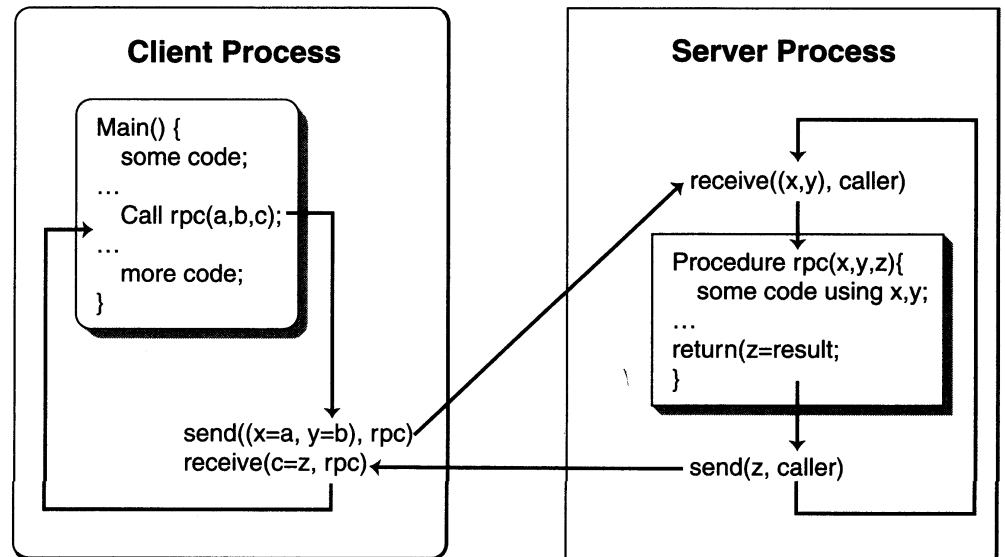
- remote procedure calls
- object-oriented programming
- remote method invocation
- CORBA vs. OLE/COM

Remote procedure calls

recall: remote procedure calls provide an intuitive model for client/server computing

1. client makes remote procedure call
2. calling process is suspended, parameters are sent to the server, and procedure is executed there
3. when procedure completes, results are sent back to the client
4. client resumes processing

Note: if global variables are allowed, then the RPC call must send process context to the server



RPC stubs & harnesses

goal: make the distributed nature of RPCs transparent to the client

- want an RPC to look like a local procedure call

on the client machine:

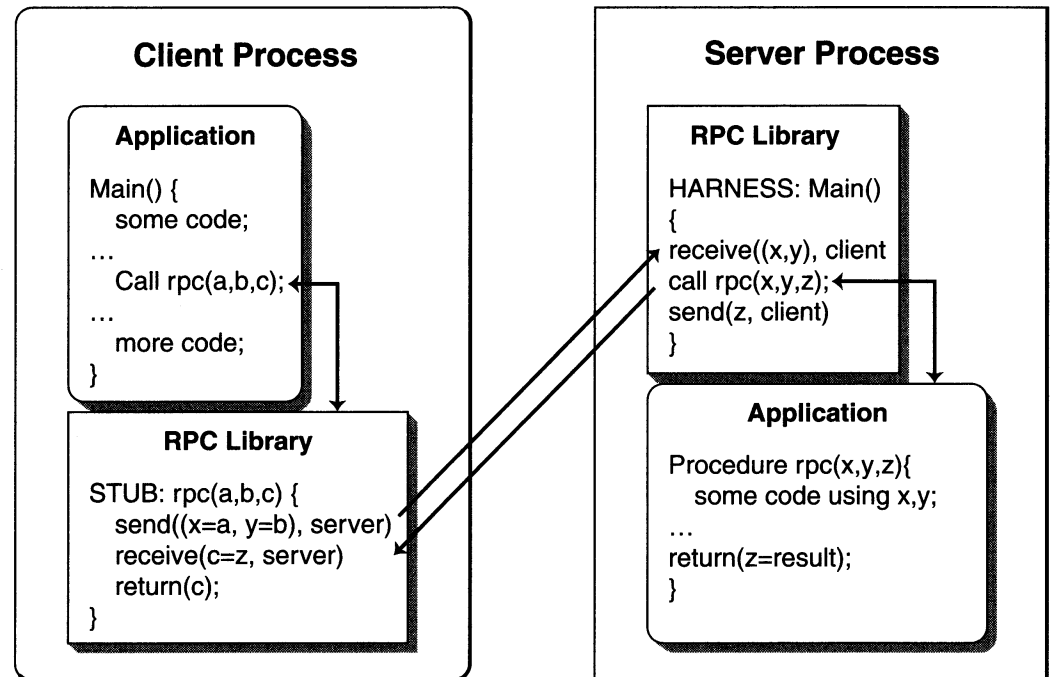
maintain a library of procedure stubs

the RPC invokes the stub, which acts as intermediary for communication with the server

on the server machine:

maintain a library of harnesses

the harness receives the communication from the client, calls the appropriate procedure, and returns the result



Creating stubs & harnesses

start with an Interface Definition Language (IDL) file

- describes the interface for the remote procedure + unique ID number
- interface definitions are similar to C/C++ code
- not fully standardized, proprietary variations exist

```
[ uuid (906B0CE0-C70B-1067-B317-00DD010662DA), version(1.0), pointer_default(unique)]  
  
interface hello  
{  
    void HelloProc([in, string] unsigned char * pszString);  
}
```

an IDL compiler generates the stubs & harnesses from the IDL file
e.g., Microsoft IDL (MIDL) compiler,

Communicating objects

software development has moved toward object-oriented programming

- model real-world objects as software objects (ADTs / classes)
- objects encapsulate data + operations (methods)
- a program is a collection of interacting objects, communicate via method calls

- inheritance allows for new classes to be built on top of existing classes
- inheritance defines an IS_A relationship, can write generic code for an entire hierarchy of classes

the OO equivalent of an RPC is an RMI (remote method invocation)

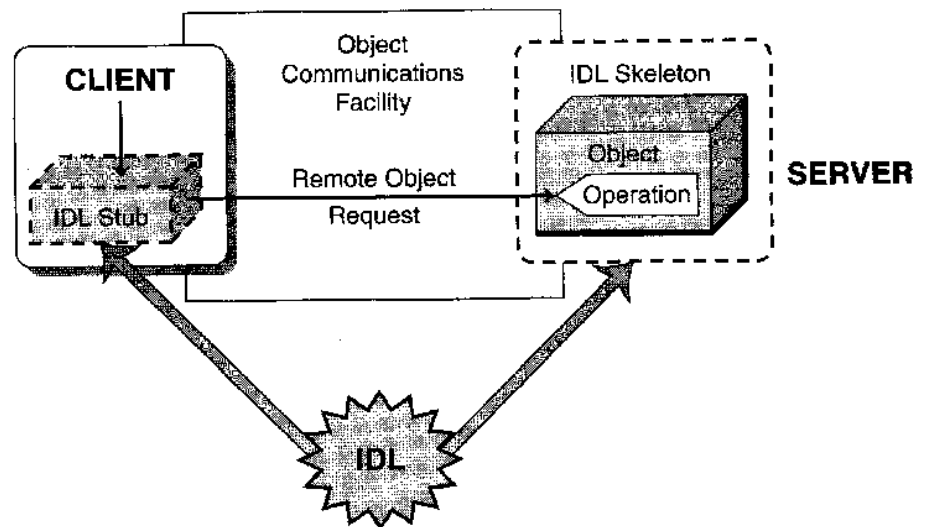
1. client makes remote method invocation
2. calling process is suspended, parameters are sent to the server, and the method is applied to the appropriate object there
3. when procedure completes, results are sent back to the client
4. client resumes processing

RMI stubs & skeletons

as with RPCs, use an IDL to define an interface (signature) for the remote object

```
interface BankAccount : Bank_Object
{
    exception Insufficient_Funds { float balance; }
    float method_withdrawal(in float amount);
    float method_deposit(in float amount);
    float method_transfer(in long from_account, in long to_account,
        in float amount, out float to_account_balance) raises Insufficient_Funds;
    float method_balance();
};
```

as with RPCs, an IDL compiler generates intermediary files for the client & server



Object references

local method invocation: object maintains a pointer to its method

- when apply method to an object, dereference the pointer & execute

remote method invocation: client uses an object reference (OID)

- obtained by the client when the stub & skeleton are bound together
- OID is a typed data structures, serves as a local proxy for the remote object
- can be substantial in size → RMIs can be costly

locating the appropriate object on the server:

- a broker service looks up the OID by name (analogy: white pages)
- a trader service looks up the OID by its interface (analogy: yellow pages)

- most object communication facilities provide broker service

CORBA & COM/OLE

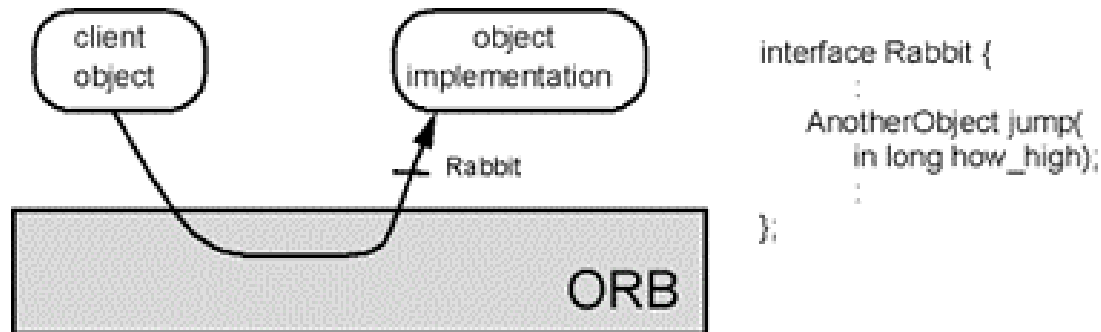
2 main communication standards for distributed objects

- CORBA (Communicating Object Request Broker Architecture)
defined by the Object Management Group (OMG), a non-profit consortium
gives a specification for a distributed environment with object communication
NOTE: CORBA is a specification, not an implementation
 - see IBM's DSOM, HP's DOMF, Sun's DOE, ...
- OLE/COM (Object Linking & Embedding/Component Object Model)
object communications package developed by Microsoft

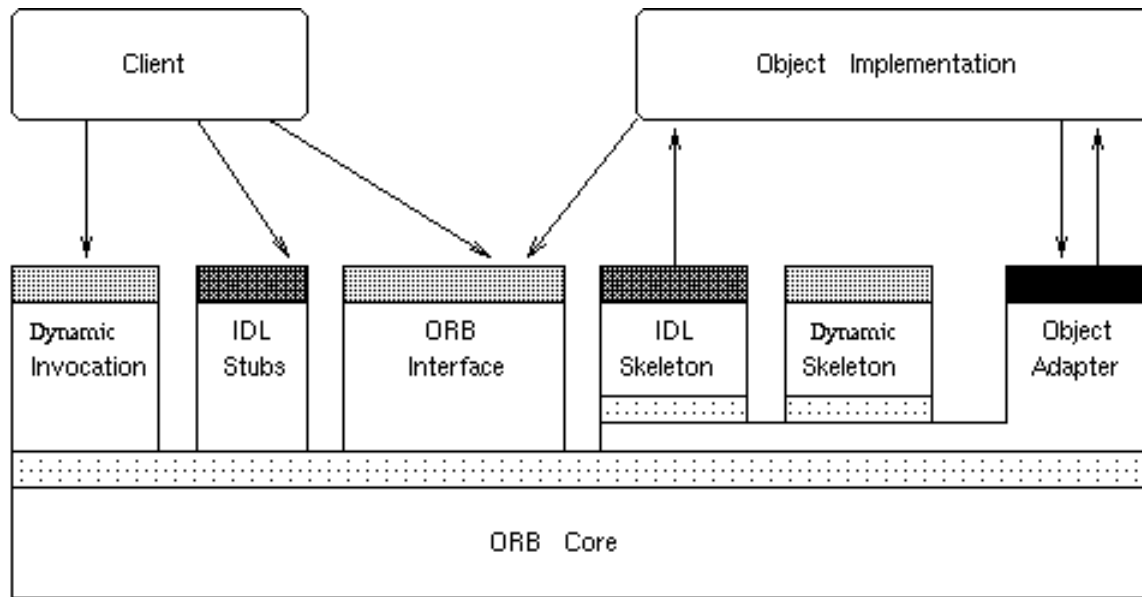
CORBA





CORBA defines an architecture for distributed objects

- the central component is the Object Request Broker (ORB)
- the ORB must
 - (1) locate the remote object
 - (2) communicate the request (method invocation)
 - (3) wait for the results
 - (4) communicate the results back to the client
- the ORB implements programming language independence for the request
the client and server object do not have to be in the same language



ORB architecture



-  interface identical for all ORB implementations
-  there may be multiple object adapters
-  stubs and skeletons for each object type
-  ORB dependent interface

client can communicate through

- IDL Stub (as described earlier)
- dynamic invocation interface (allows client to search for object given specific attributes)
- the ORB interface to common functions

Object Adapters generate & interpret

- OIDs
- method invocation
- security of transaction
- object activation/deactivation
- mapping references to objects
- registration of implementations

CORBA compliance

there are many different ORB products available

to be CORBA-compliant, must support

- static and dynamic binding
- IDL language bindings for C, C++, Smalltalk
- Interface Repository in which to register object services (for DI)
- Basic Object Adapter interface

- common services are optional, will vary according to vendor

OLE/COM

OLE provides an application integration framework for Windows

- OLE defines the Component Object Model, a language-independent standard for object implementations
- objects are encapsulated into components (which hide internal details)
- a component server provides access to components via interfaces
 - interface manipulation is more advanced than in CORBA
 - e.g., client can query object about supported interfaces
 - supports limited inheritance of interfaces
- OLE maintains a system registry of components
 - when the client makes a request, the COM Service Control Manager
 1. accesses the registry to find the component
 2. loads the component into memory
 3. asks the component to create an instance of the desired object
 4. returns an interface pointer to the client
 - the client then appears to invoke the method directly on that interface

OLE services

OLE provides an extensive library of interfaces

- containing Windows APIs and predefined functions
- similar in functionality to the CORBA ORB & Object Adapters
- much richer set of operations than CORBA
 - e.g., unlike CORBA, OLE allows the user to add new services
- to be fair: CORBA is a generic specification, vendors can add extras

Next week...

SQL

- relational database model
- Structured Query Language (SQL)
- object database model
- SQL APIs

Read Chapters 11 & 12

As always, be prepared for a short quiz