Outline

• Distributed Programming
• RMI
• CORBA
• Web Services
• Test Yourself
Remote Procedure Calls

- Distributed Computing is also known as a **Remote Procedure Call (RPC)**
- An RPC is an inter-process communication mechanism that allows a program to execute a procedure in another address space
  - This could be another computer or another program running within the same computer
- There are many different RPC frameworks
  - Remote Method Invocation (RMI)
  - Common Object Request Broker Architecture (CORBA)
  - Web Services
  - XML-RPC
  - JSON-RPC
  - Microsoft .NET Remoting
  - Microsoft RPC
  - Many others
Message Passing

- When a procedure is invoked in another address space (which typically would be on another computer), data needs to be passed to it
- This is accomplished through **message passing**
- Message passing is as rudimentary as just passing data in an agreed-upon protocol
  - Each RPC framework has its own message passing protocol, and unfortunately, they are not typically interchangeable
- If the message passing protocol is generic enough, the programs communicating with each other do not have to be written in the same language
  - This usually requires variable types to be standardized
RPC Runtime Lifecycle

- Client code calls a procedure in the client stub just as it would call a local procedure
- The client stub packs the parameters into a message
  - Packing the parameters is called marshalling
- The client stub sends the message to the server
- The server stub receives the message
- The server stub unpacks the parameters from the message
  - Unpacking the parameters is called unmarshalling
- The server stub calls the server’s procedure as it would call a local procedure
- The response from the RPC is similar
RPC Synchronization

- RPCs can be either *blocking* or *non-blocking*
  - A **blocking** RPC will wait for a response from the server before continuing
  - A **non-blocking** RPC will continue executing while waiting for a response from the server
    - This can be implemented either with multi-threading or parallel programming
Outline

• Distributed Programming
• RMI
• CORBA
• Web Services
• Test Yourself
RMI Overview

- RMI is a Java-implementation of RPC referred to as a distributed object application
- An RMI server typically creates some remote objects, makes references to those objects accessible, and waits for clients to invoke methods on those objects
- An RMI client obtains a remote reference to one or more remote objects on a server and invokes methods on them
  - RMI clients can locate remote objects through an RMI registry, assuming the remote objects have registered with the RMI server
- The details of remote communication between server and client are handled by RMI
  - Remote communication looks like regular Java method invocations to the programmer
- The client can pass a class to a remote server and have it execute methods on that class
Since RMI clients and servers are both written in Java, we can pass Java objects between the two programs. These objects are passed through serialization.
Outline

• Distributed Programming
• RMI
• CORBA
• Web Services
• Test Yourself
An interface definition language (IDL) provides a specification to describe a software component’s interface.

This is usually a language-independent way to specify a shared interface to allow two programs to communicate with each other.

IDLs are commonly used in remote procedure calls (RPCs) when the languages used for the client and server programs can be different.

The mapping between variables in each of the languages to a common format must be defined in the IDL.
IDLs

- Object Management Group (OMG) IDL
  - Used by CORBA
- RESTful Service Description Language (RSDL)
- JSON Web Service Protocol (JSON-WSP)
- Web Service Description Language (WSDL)
- Apache Thrift (originally Facebook Thrift)

- Is there an IDL for RMI?
  - No because RMI uses Java code on both the client and server, only requiring serialization of Java objects
IDL Process

- The first step to creating a distributed application is to define the IDL.
- This will define the objects and variables that are able to be transmitted between the client and the server during a method invocation.
- The methods, return types, and parameters are also defined in the IDL.
- IDL defines a list of variable types:
  - Integers, floating point values, characters, strings, booleans, bytes, structures, unions, enumerations, arrays, or a generic type.
IDL Example

```
sum.idl
1    module SumApp {
2        interface SumServer {
3            long sum(in long a, in long b);
4        }
5    }
```

- A module corresponds to a package in Java or a namespace in C++
- An interface corresponds to an interface in Java or an abstract class in C++
- The method in the middle will get compiled to an abstract method/virtual function
- Here is the compiled Java code for the above IDL:

```
SumServerOperations.java
1    package SumApp;
2
3    /**
4        * SumApp/SumServerOperations.java .
5        * Generated by the IDL-to-Java compiler (portable), version "3.2"
6        * from sum.idl
7        * Sunday, April 12, 2015 7:20:27 PM PDT
8        */
9
10   public interface SumServerOperations {
11      int sum (int a, int b);
12   } // interface SumServerOperations
```
CORBA Overview

- The Common Object Request Broker Architecture (CORBA) is the Object Management Group’s (OMG) open, vendor-independent architecture and infrastructure that computer applications use to work together over networks
  

- CORBA is operating system- and language-independent

- CORBA uses IDL to map programming language specifics to OMG standards, with mappings from IDL to C, C++, C++11, Java, Ruby, COBOL, Smalltalk, Ada, Lisp, Python, and IDLscript defined
CORBA Request

- CORBA clients communicate with an IDL stub, which is automatically generated from the IDL file
- CORBA servers communicate with an IDL skeleton, which is automatically generated from the IDL file
- The Object Request Broker (ORB) in the middle performs the communication and passes the data along in a standardized format from the stub to the skeleton

![Diagram showing the CORBA Request process]

Figure 1: A request passing from client to object implementation

Copyright © The Object Management Group
Outline

- Distributed Programming
- RMI
- CORBA
- Web Services
- Test Yourself
Web Services Overview

- Web services are client and server applications that communicate over HTTP to implement RPCs
  - They are language- and platform-independent
  - Data is transmitted in a standardized XML format

- Big Web Services
  - Java API for XML Web Services (JAX-WS)
  - Use XML messages following Simple Object Access Protocol (SOAP)
  - Operations offered by the service are included in the Web Services Description Language (WSDL)

- RESTful Web Services
  - Java API for RESTful Web Services (JAX-RS)
  - No requirement of XML messages or WSDL service

- Additional information can be found at http://docs.oracle.com/javaee/6/tutorial/doc/gijti.html
JAX-WS

- **Server side**
  - Developer specifies the web service operations by defining methods in an interface
  - Create one or more classes that implement those interfaces
  - Create a WSDL document to let clients know the web services that are exposed

- **Client side**
  - Create a proxy, which is a local object representing the service
  - Create a program that invokes methods on the proxy

- JAX-WS will generate and parse the SOAP messages and convert the API class to and from SOAP
Representational State Transfer (REST) is an architectural style of client-server applications centered around the transfer of resources through requests and responses using Uniform Resource Identifiers (URIs)

- Communication is over HTTP without the requirement of SOAP, XML, or WSDL
- A REST resource might be XML, HTML, plain text, PDF, JPEG, JSON, or other types
- This resource can be retrieved, updated, or deleted based on permissions
- REST uses HTTP, which is stateless, but state can be maintained with cookies, URI rewriting, and hidden form fields
JAX-WS vs JAX-RS

- JAX-WS is used in enterprise applications with advanced Quality of Service (QoS) requirements
  - Supports the WS-* set of protocols, providing standards for security and reliability (through SOAP)

- JAX-RS is used for web applications using the REST architectural pattern
  - Stateless operations (can withstand server restart)
  - Scalability
  - Architectural simplicity (use off the shelf components)
  - Resources can be accessed using a variety of formats, so it is not limited solely to SOAP
Outline

• Distributed Programming
• RMI
• CORBA
• Web Services
• Test Yourself
Test Yourself

- If the client was in C++ and the server was in Java, which RPC would you recommend using?
- If the client and server were both in Java, which RPC would you recommend using?
- If the network administrator didn’t want to open any additional ports through the firewall, which RPC would you recommend using?
- If the data to transmit was in a PDF, which RPC would you recommend using?