# Appendix D: Distributed Communication

- Sockets
- Remote Procedure Calls (RPCs)
- Remote Method Invocation (RMI)
- CORBA
- Object Registration

# Sockets

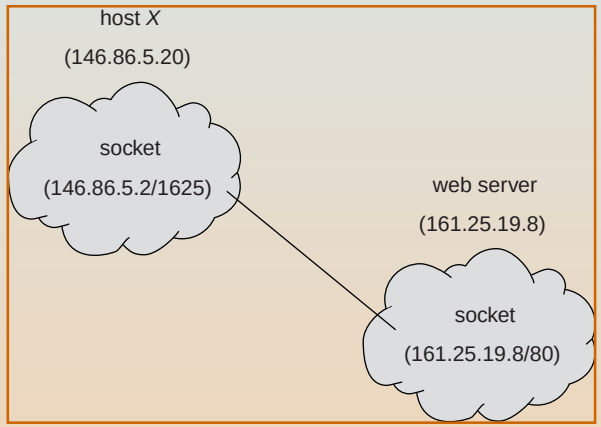
- Defined as an “endpoint for communication”
- Concatenation of IP Address + Port
- All Ports < 1024 are Considered “well-known”

  - TELNET uses port 23

  - FTP uses port 21

  - HTTP server uses port 80

# Communication Using Sockets




# Java Sockets

- Java Provides:
  - Connection-Oriented (TCP) Sockets
  - Connection-less (UDP) Sockets
  - Multicast Connection-less Socket

# Time-Of-Day Server/Client

- Server uses

```
s = new ServerSocket(5155)
```

  To Create the Socket on Port 5155

- To Accept Connections From Clients:

```
Socket client = s.accept()
```

- Connections are Often Serviced in Separate Threads

- The Client Connects to the Server Using:

```
Socket s = new
Socket("127.0.0.1",5155);
```

  Using the IP Address of the Server.

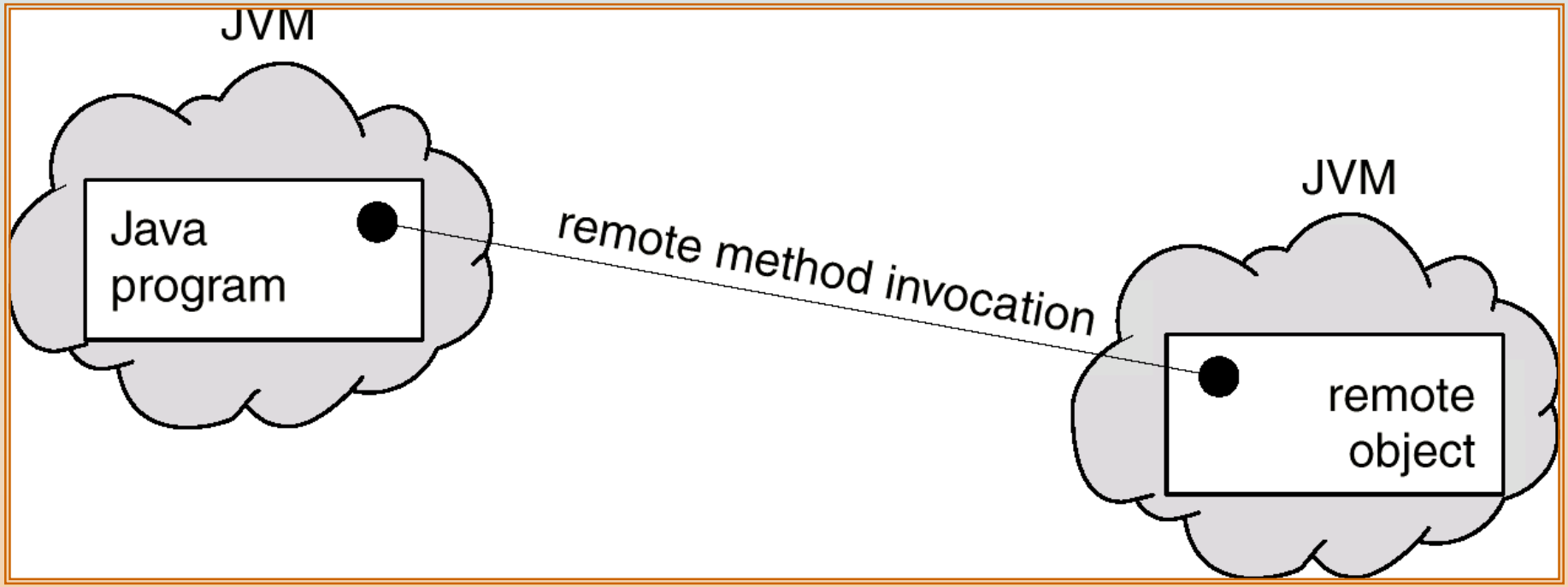
# Remote Procedure Calls (RPC)

- Sockets are Considered Low-level.
- RPCs Offer a Higher-level Form of Communication
- Client Makes Procedure Call to “Remote” Server Using Ordinary Procedure Call Mechanisms.

# Remote Method Invocation (RMI)

- Java's Version of RPCs
- A Thread May Invoke a Method on a Remote Object
- An Object is Considered "remote" if it Resides in a Separate Java Virtual Machine.

# Remote Method Invocation (BMI)



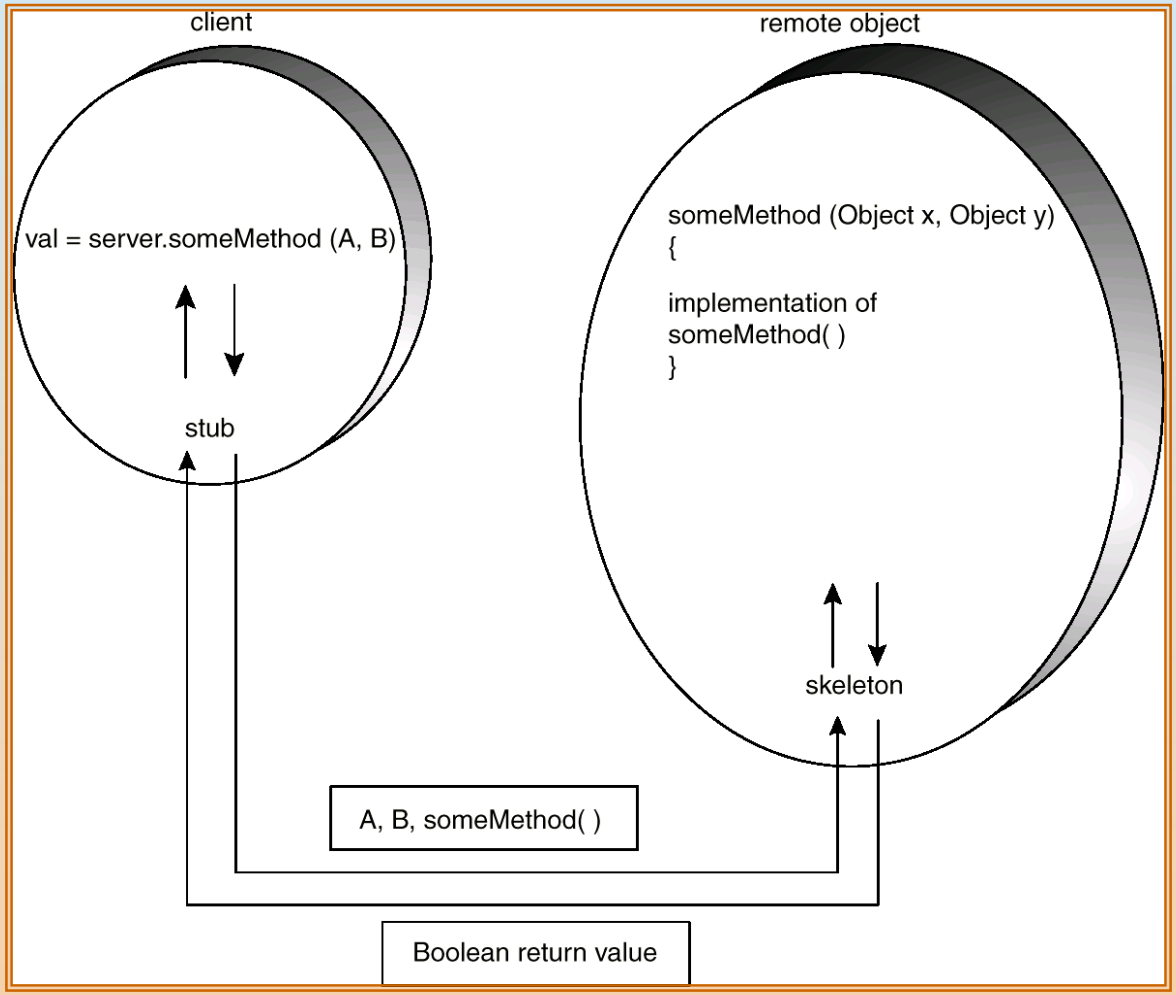
# RPC versus RMI

- RPC's Support Procedural Programming Style
- RMI Supports Object-Oriented Programming Style
- Parameters to RPCs are Ordinary Data Structures
- Parameters to RMI are Objects

# Stubs and Skeletons

- "Stub" is a Proxy for the Remote Object – Resides on Client.
- The Stub "Marshalls" the Parameters and Sends Them to the Server.
- "Skeleton" is on Server Side.
- Skeleton "Unmarshalls" the Parameters and Delivers Them to the Server.

# Marshalling Parameters




# Parameters

- Local (Non-Remote) Objects are Passed by Copy using Object Serialization
- Remote Objects are Passed by Reference

# Remote Objects

- Remote Objects are Declared by Specifying an interface that extends `java.rmi.Remote`
- Every Method Must Throw `java.rmi.RemoteException`

# MessageQueue interface

```
public interface MessageQueue
   extends java.rmi.Remote
{
   public void send(Object item)
       throws java.rmi.RemoteException;
   public Object receive()
       throws java.rmi.RemoteException;
}
```

# MessageQueue implementation

```
public class MessageQueueIMPL
   extends java.rmi.server.UnicastRemoteObject
   implements MessageQueue
{
   public void send(Object item)
       throws java.rmi.RemoteException
   { /* implementation */ }
   public Object receive()
       throws java.rmi.RemoteException
   { /* implementation */ }
}
```

# The Client

- The Client Must

  (1) Install a Security Manager:

```
System.setSecurityManager(
        new RMISecurityManager());
```

  (2) Get a Reference to the Remote Object

```
MessageQueue mb;
mb = (MessageQueue)Naming.

lookup("rmi://127.0.0.1/MessageServer")
```

# Running the Producer-Consumer Using RMI

- Compile All Source Files
- Generate Stub and Skeleton

```
rmic MessageQueueImpl
```

- Start the Registry Service

```
rmiregistry
```

- Create the Remote Object

```
java –Djava.security.policy=java.policy
      MessageQueueImpl
```

- Start the Client

```
java –Djava.security.policy=java.policy
      Factory
```

# Policy File

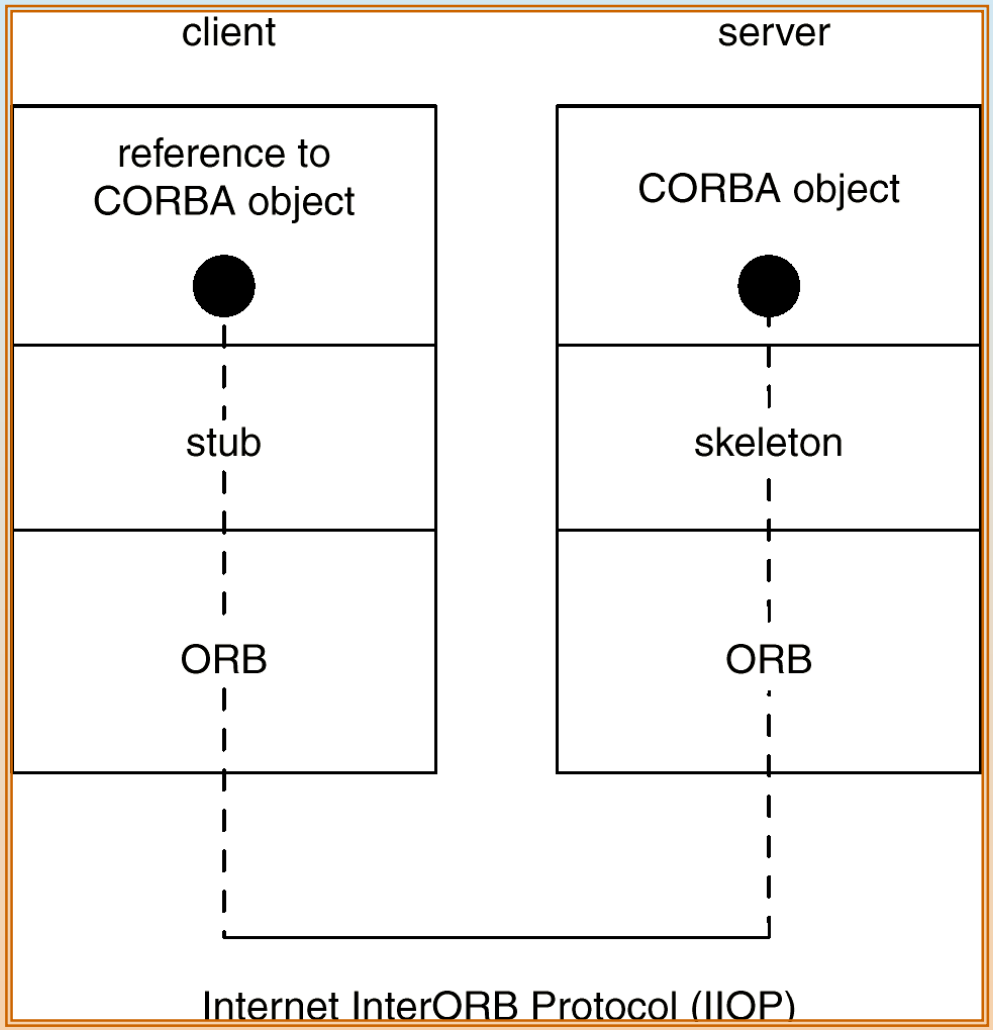
- New with Java 2

```
grant {
        permission java.net.SocketPermission
            "*:1024-65535","connect,accept";
};
```

# CORBA

- RMI is Java-to-Java Technology
- CORBA is Middleware that Allows Heterogeneous Client and Server Applications to Communicate
- Interface Definition Language (IDL) is a Generic Way to Describe an Interface to a Service a Remote Object Provides
- Object Request Broker (ORB) Allows Client and Server to Communicate through IDL.
- Internet InterORB Protocol (IIOP) is a Protocol Specifying how the ORBs can Communicate.

# Cobra Model


client
server
reference to CORBA object
CORBA object
stub
skeleton
ORB
ORB
Internet InterORB Protocol (IIOP)

# Registration Services

- Registration Service Allows Remote Objects to “register” Their Services.
- RMI,  CORBA Require Registration Services