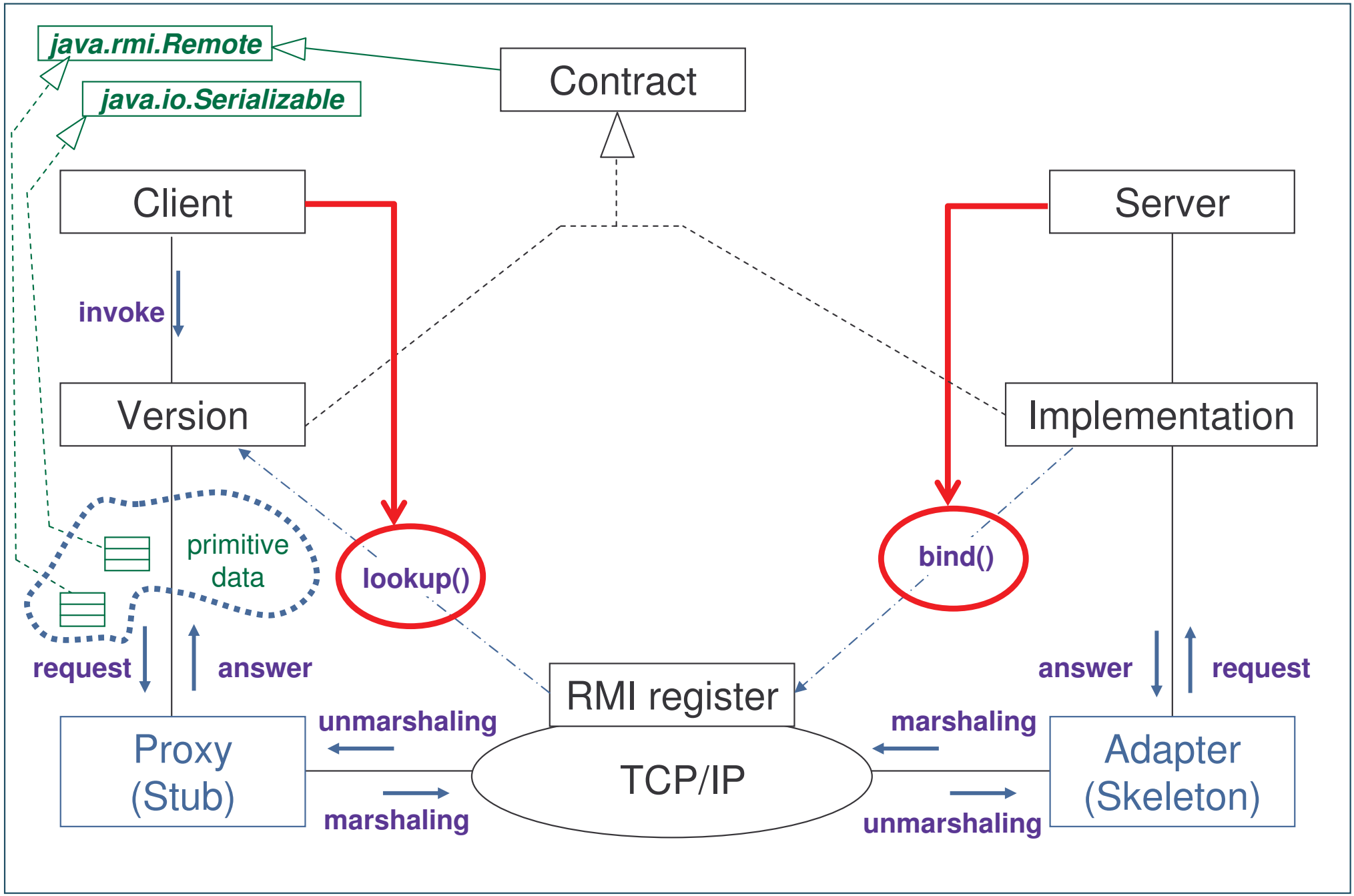
RMI

- RMI = Remote Method Invocation

- Java-specific mechanism for RPC and CORBA
- RPC = Remote Procedure Call
  - Platform-dependent
  - Language-dependent
- CORBA = Common Object Request Broadcast Architecture
  - RPC in heterogeneous language context
  - Internet Inter-ORB Protocol (IIOP)

RMI > Communication architecture



- **Five Steps:**

1. **Create and compile the classes:**
   Contract.java, Server.java, Implementation.java & Client.java
2. **Create the Adapter & the Proxy:**
   rmic Implementation ⇒ *Contract_Skel.class* & *Contract_Stub.class*
   (Contract-Stub visible by Client's CLASSPATH or downloadable)
3. **Start the RMI register:**
   rmiregistry
4. **Run the Server:**
   java server_Name
   (or java –Djava.rmi.server.codebase=proxy's_URL server_Name)
5. **Run the Client**

RMI > Client & Server


```
import java.rmi.*; import java.util.*; import java.rmi.server.*;
public interface RemoteDate extends Remote {
        public Date getRemoteDate() throws RemoteException;
        public final static String IDENTITY = "calendar";
}
public class Skeleton extends UnicastRemoteObject implements RemoteDate {
        public Skeleton() throws RemoteException { super(); }
        public Date getRemoteDate() throws RemoteException { return new Date(); }
}
public class Server {
        public static void main(String[ ] arg) {
                try {   Skeleton sk = new Skeleton();
                        Naming.bind(RemoteDate.IDENTITY,sk);
                } catch(Exception e) { System.err.println(e); System.exit(1); }
        }
}
public class Client {
        protected static RemoteDate cnc = null;
        public static void main(String[ ] arg) {
                try {   cnc = (RemoteDate) Naming.lookup(RemoteDate.IDENTITY);
                        System.out.println("Today: " + cnc.getRemoteDate().toString());
                } catch(Exception e) {System.err.println("Error! " + e.getMessage()); }
        }
}
```

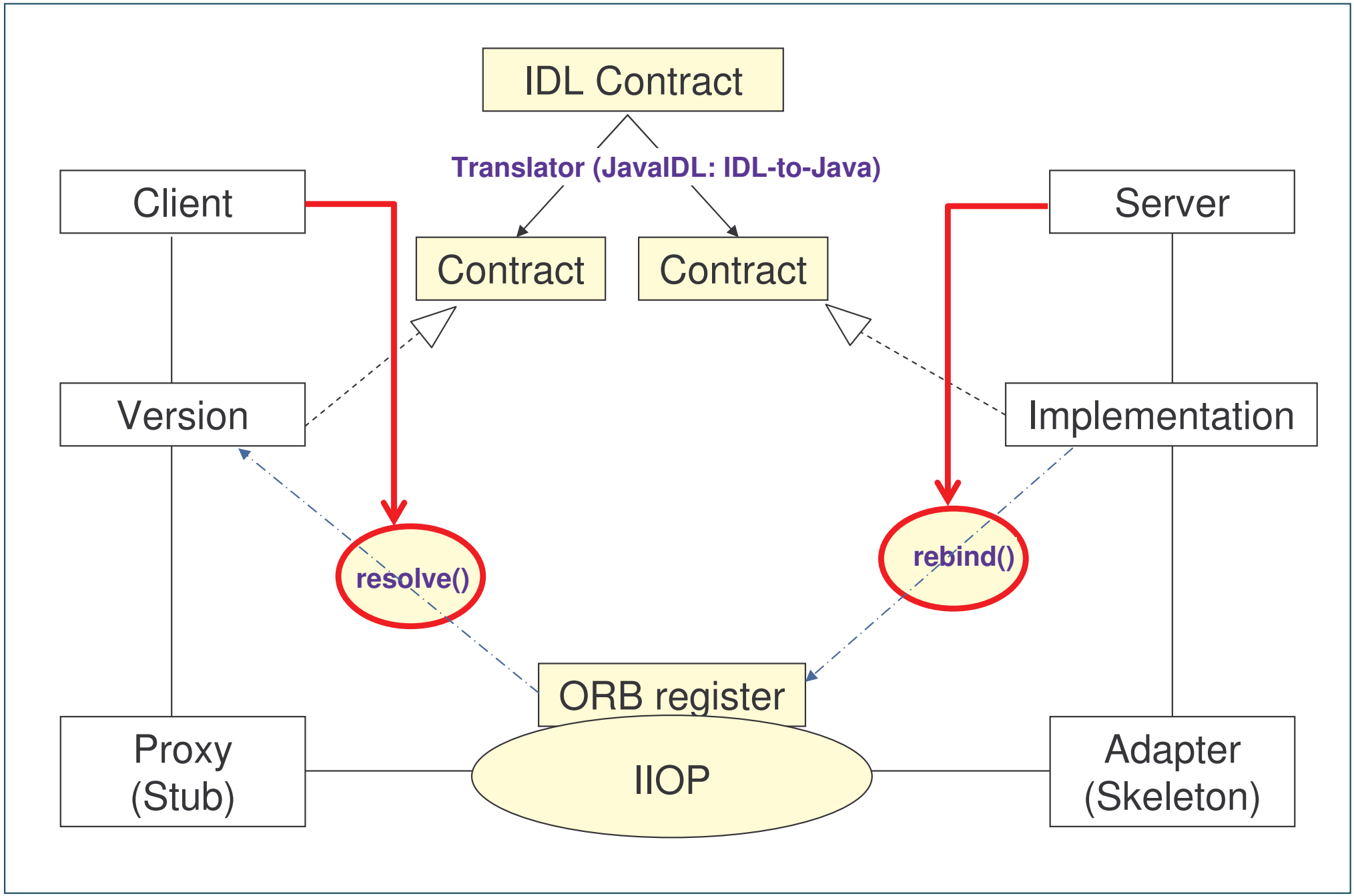
Method()

Contract

Implementation

Server

Client


IDL Contract
Translator (JavaIDL: IDL-to-Java)
Client
Server
Contract
Contract
Version
Implementation
resolve()
rebind()
ORB register
Proxy
(Stub)
IIOP
Adapter
(Skeleton)

RMI > CORBA-to-RMI interfaces


```
module myRmi {
        module clock {
                interface RemoteDate {
                        void getDate(out int dd, out int mm, out int yy);
                        void setDate(in int dd, in int mm, in int yy);
                }
        }
}
```

Method()

IDL Contract

idltojava –fclient InterfaceCORBA.idl

```
package myRmi.clock;
public interface RemoteDate extends org.omg.CORBA.Object {
        void getDate(intHolder dd, intHolder mm, intHolder yy);
        void setDate(int day, int mois, int an);
}
```

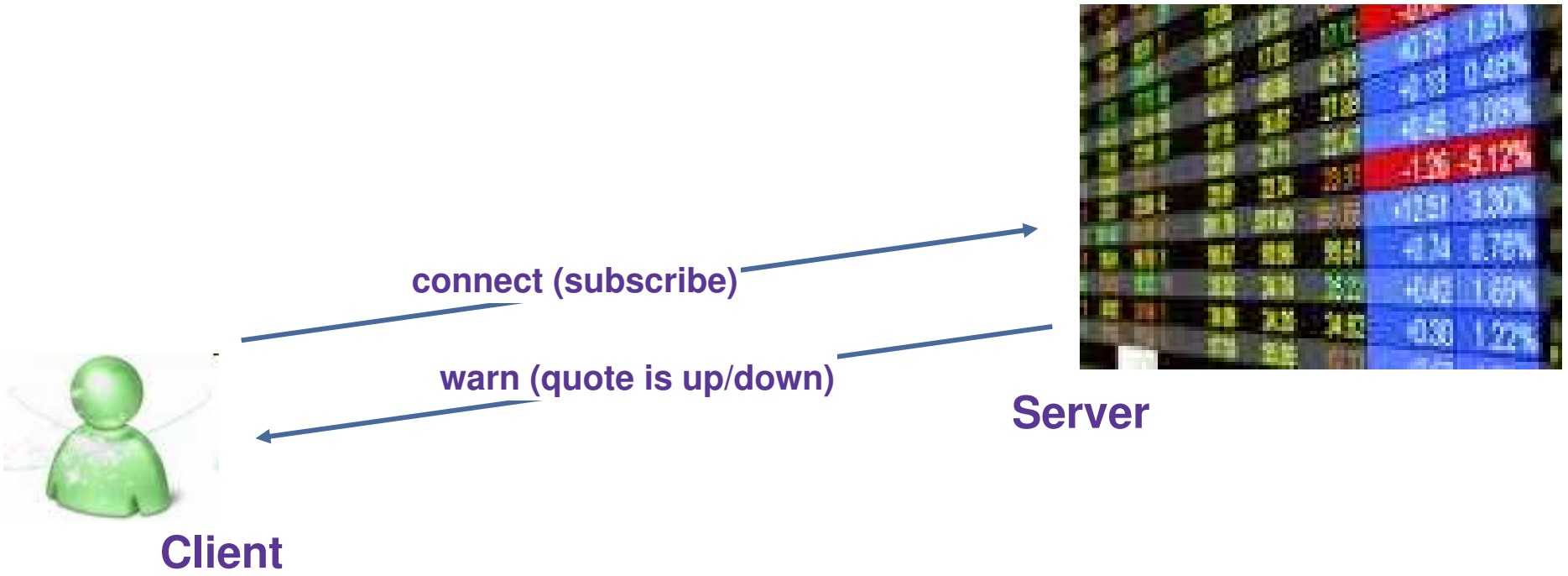
Java Client Contract

idltojava –fserver InterfaceCORBA.idl

```
package myRmi.clock;
public abstract class _RemoteDateImplBase extends org.omg.CORBA.portable. ObjectImpl
implements myRmi.clock.RemoteDate, org.omg.CORBA.portable.ServantObject {
        // ...
}
```

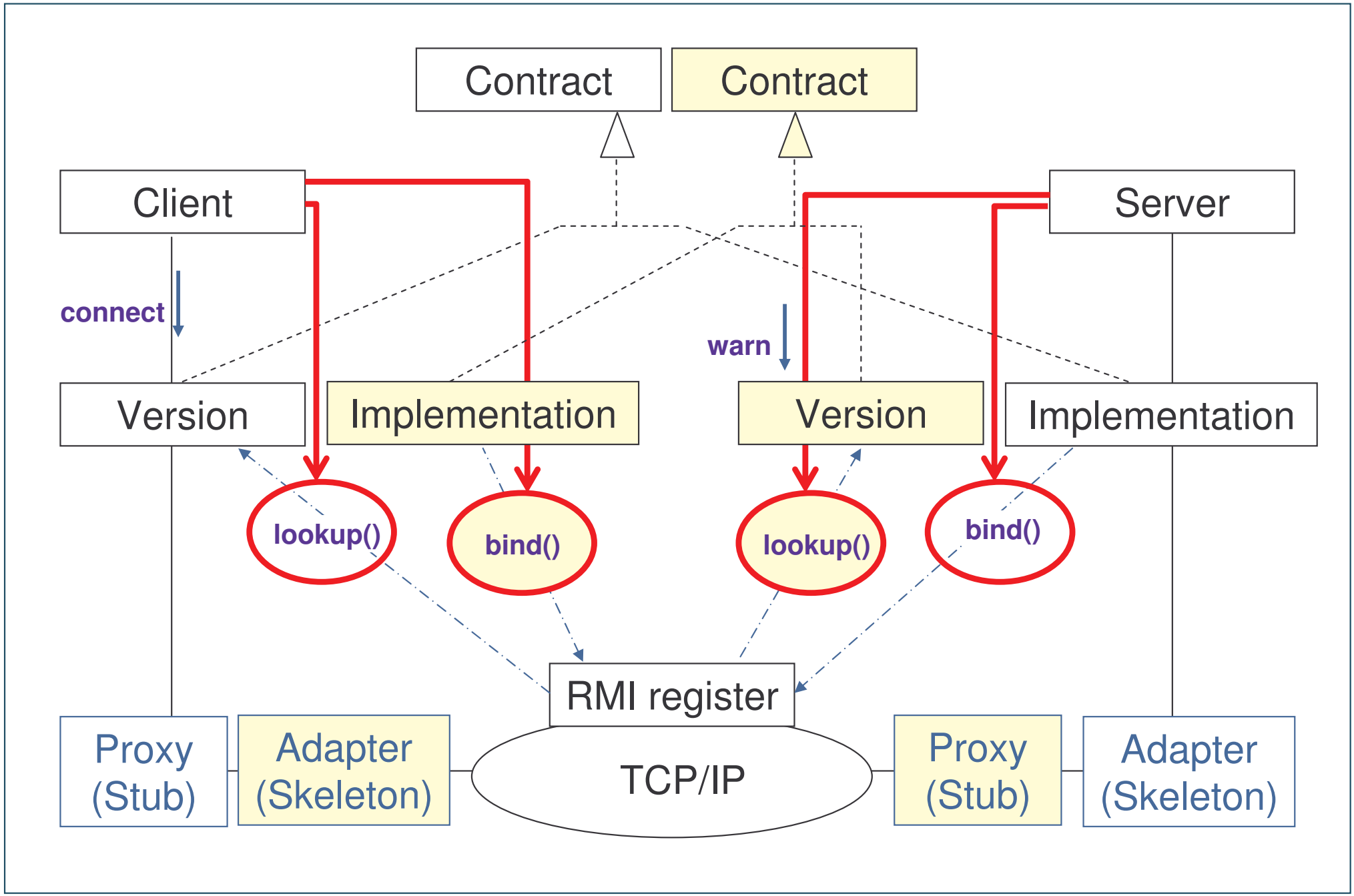
Java Server Contract

- On line trading: remote observation of stock options quotes



- Code available here: stock option, client, server, two contracts and their implementations (classes)
- Deliverables: improved classes (all)

RMI > Stock exchange communication scheme



- Three improvements:

1. User connects through a user-friendly Exchange system:
   - login is done with a nickname (which is a key)
   - listing of existing stock options is displayed at connection
   - subscription/termination is possible to one or more stock options
   - help is available to get the list of command codes or stock options
2. Subscribers to a stock option are warned when the quote changes:
   - quotes evolve randomly (probabilities to go up, to go down, and to stay constant are equal at any time)
   - warning messages give complete information about the stock options concerned
   - disconnection from the system causes termination of existing subscriptions
3. Administration interface is available:
   - to define new stock options
   - to delete dead options (with notification to subscribers)
   - to define a threshold so that any option which quote falls under this threshold is automatically deleted by the system