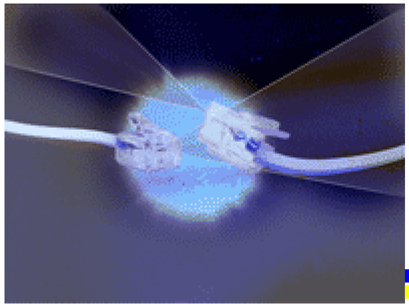
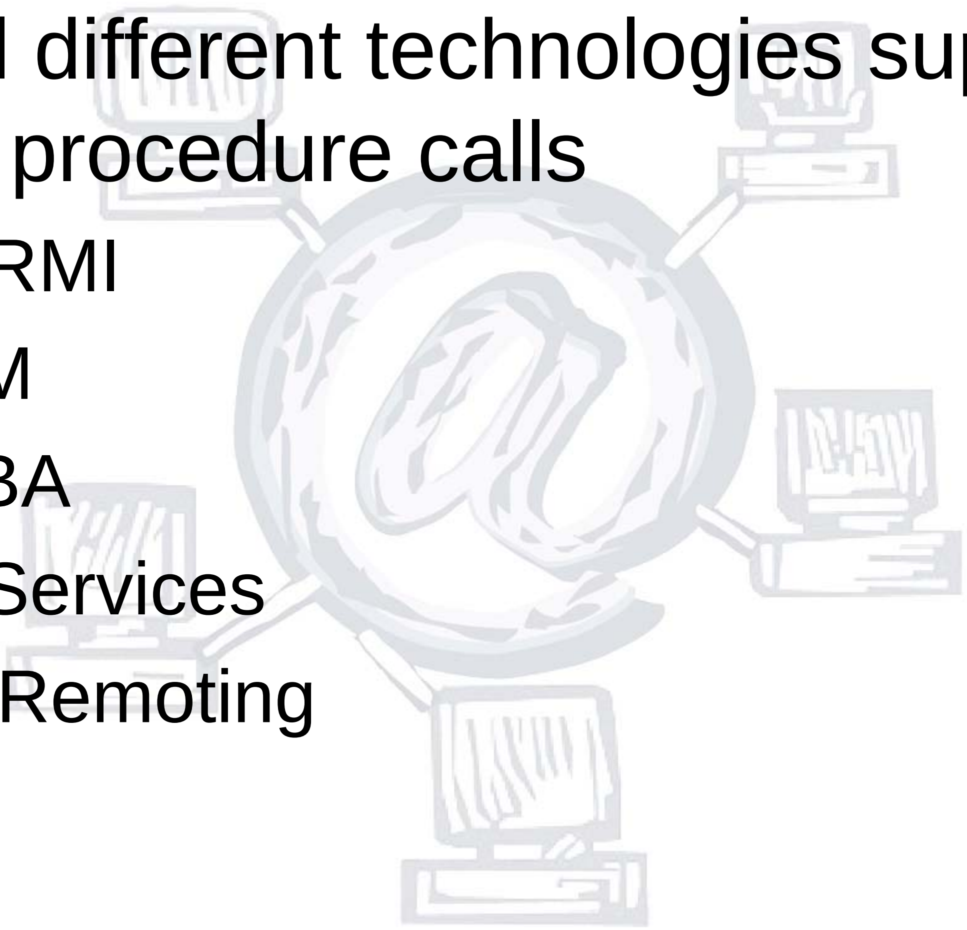
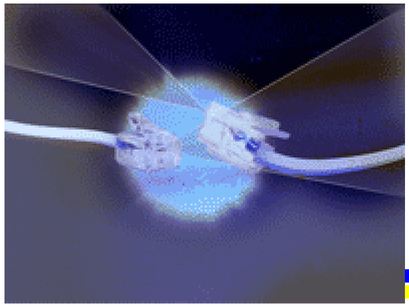


RPC Middleware

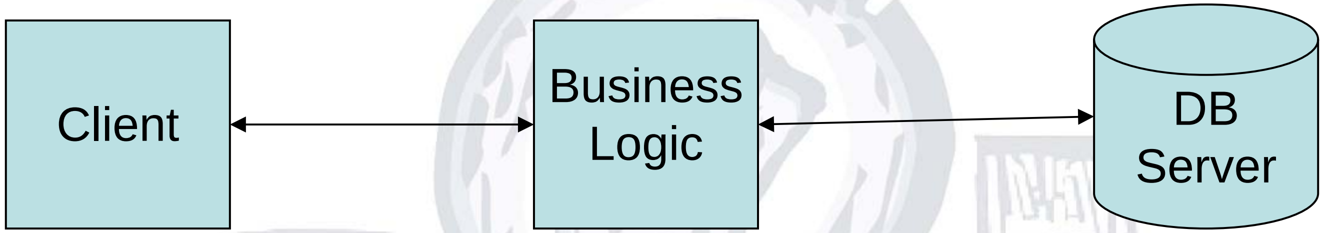


- Several different technologies support remote procedure calls
  - Java RMI
  - DCOM
  - CORBA
  - Web Services
  - .NET Remoting

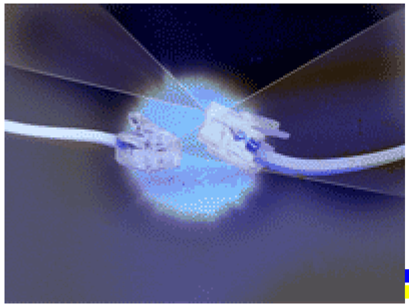


Why Call It Middleware?

- Supports implementation of the "middle" tier of a three-tiered software architecture

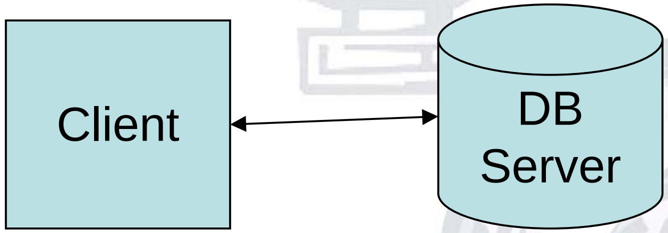


- Different paradigms
  - Object-oriented RPC mechanism
  - Message-oriented middleware (MOM)

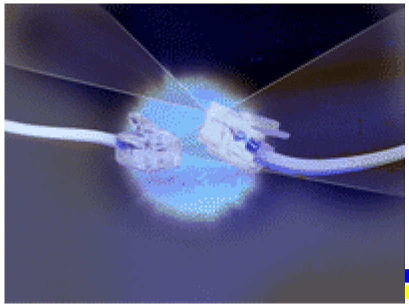


Two-Tiered Architecture

- Older architecture for distributed apps

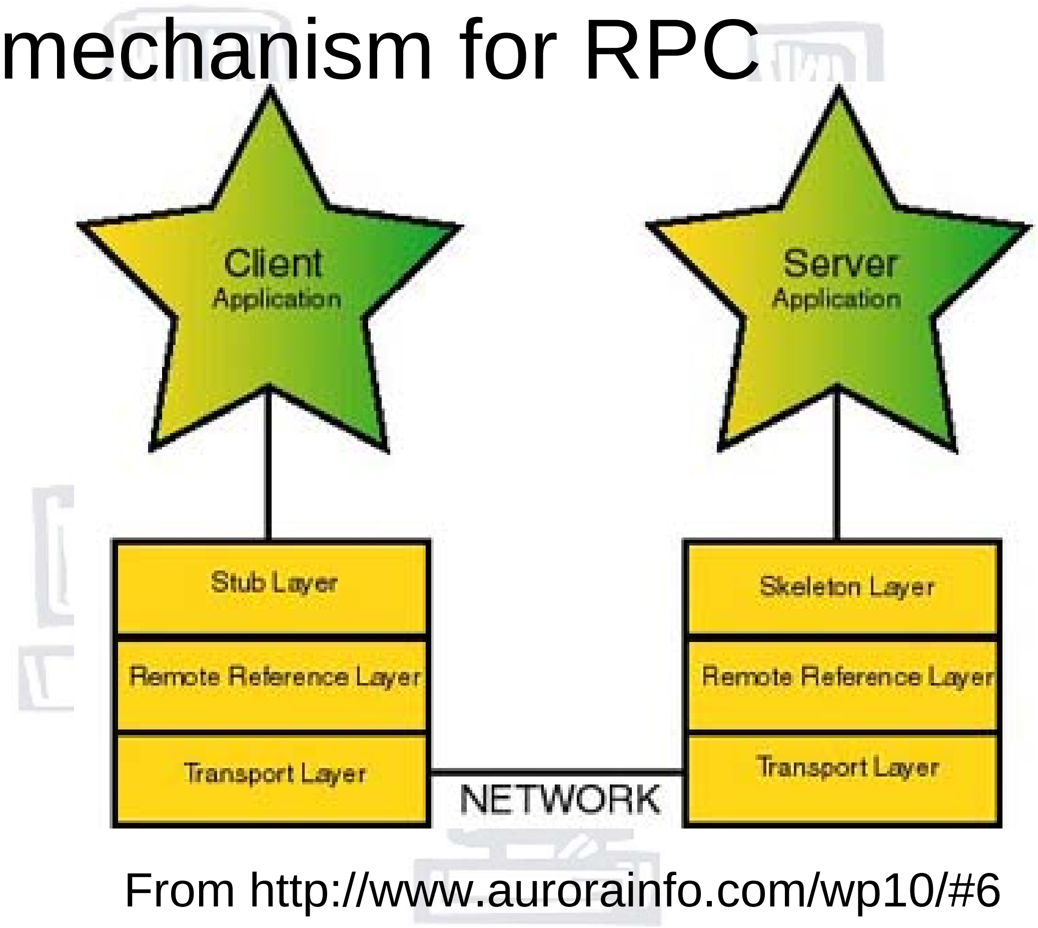


- Communication with DB server using some sort of network protocol
  - ODBC
  - JDBC
  - Proprietary from DB vendor

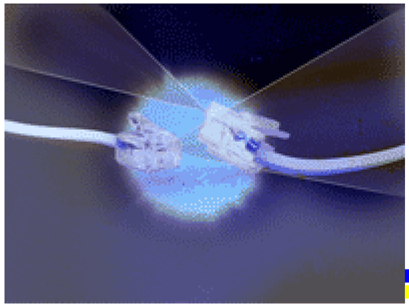


Java Remote Method Invocation (RMI)

- Java’s mechanism for RPC

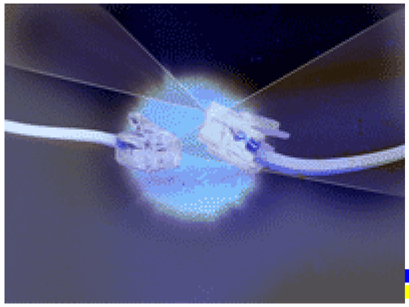


From http://www.aurorainfo.com/wp10/#6



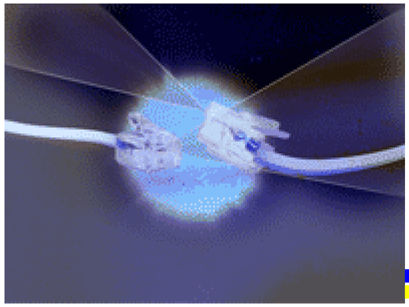
RMI (cont.)

- API component – stub/skeleton layer
  - Stub object
    - Local to client
    - Acts as surrogate for remote object
  - Skeleton object
    - Local to server
    - Driver for calls to object on server
  - Both generated from description of object's interface



RMI (cont.)

- Presentation layer component - Remote Reference Layer
  - Responsible for marshalling / demarshalling parameters
  - Intercepts calls from stubs and directs into network interface
  - Directs calls from network interface to correct skeleton

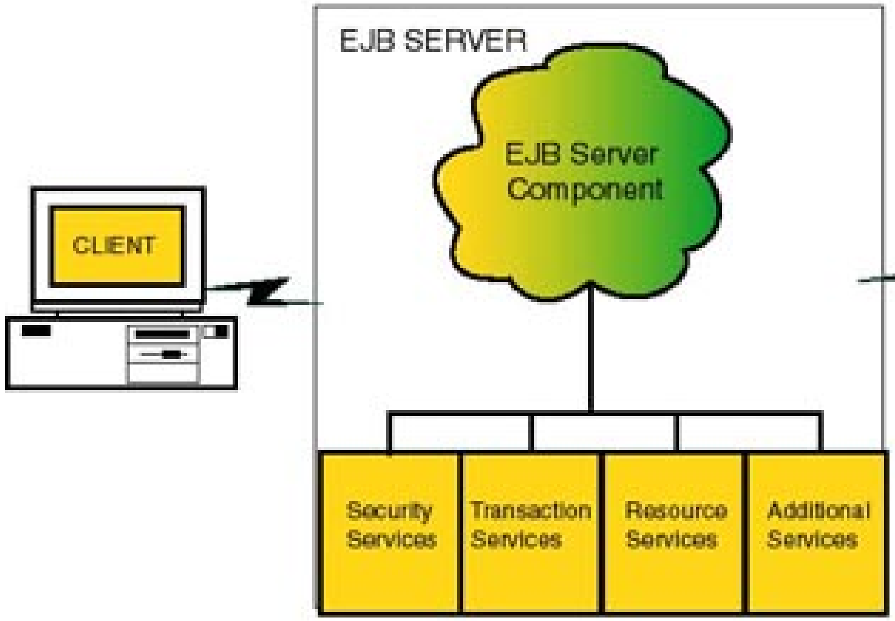


RMI (cont.)

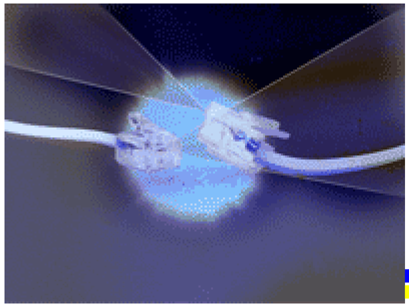
- Network interface component – Transport Layer
  - Any networking protocol supported by Java
- So the Remote Reference Layer must implement any RPC-specific networking functions (i.e. BLAST, CHAN, SELECT)

Enterprise Java Beans (EJB)

- Popular use for RMI in a three-tiered architecture

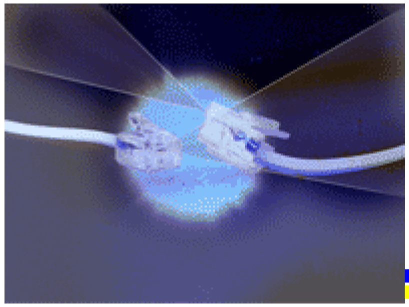


From http://www.aurorainfo.com/wp10/#6



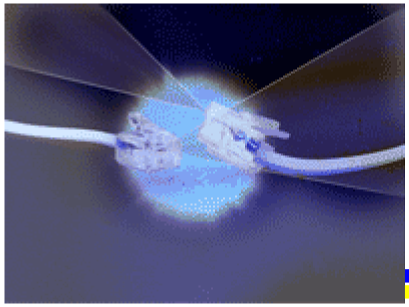
EJBs (cont.)

- Based on Java Beans
  - Spec for designing a software component with a standard interface
    - Allows manipulation by development tools
- EJBs must implement additional interfaces which allow them to be managed by an *EJB container* or *EJB application server*



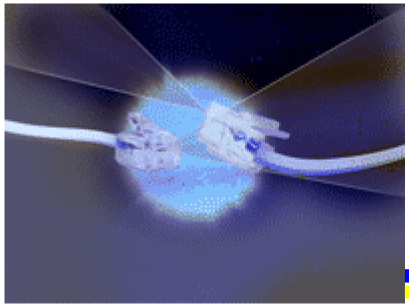
EJBs (cont.)

- Bean divided into two components
  - – Generic EJB object that implements interfaces required to interoperate with app server
  - – Bean class extends generic object, implements actual business logic
- Bean thus has two interfaces
  - – Home interface, used by app server
  - – Remote interface, used by client
    - RMI based



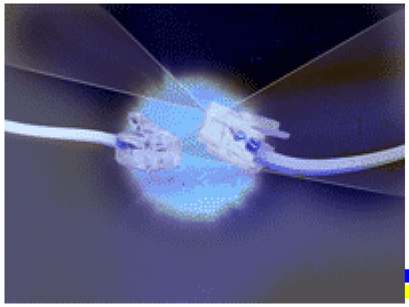
Java Naming and Directory Interface (JNDI)

- Home interfaces for EJBs used to generate *home stubs*
- These are made available via JNDI
- Client locates the home stub for a desired EJB using JNDI, invokes *create* method
- Server returns EJB object stub to client, containing EJB interface (including bean methods)



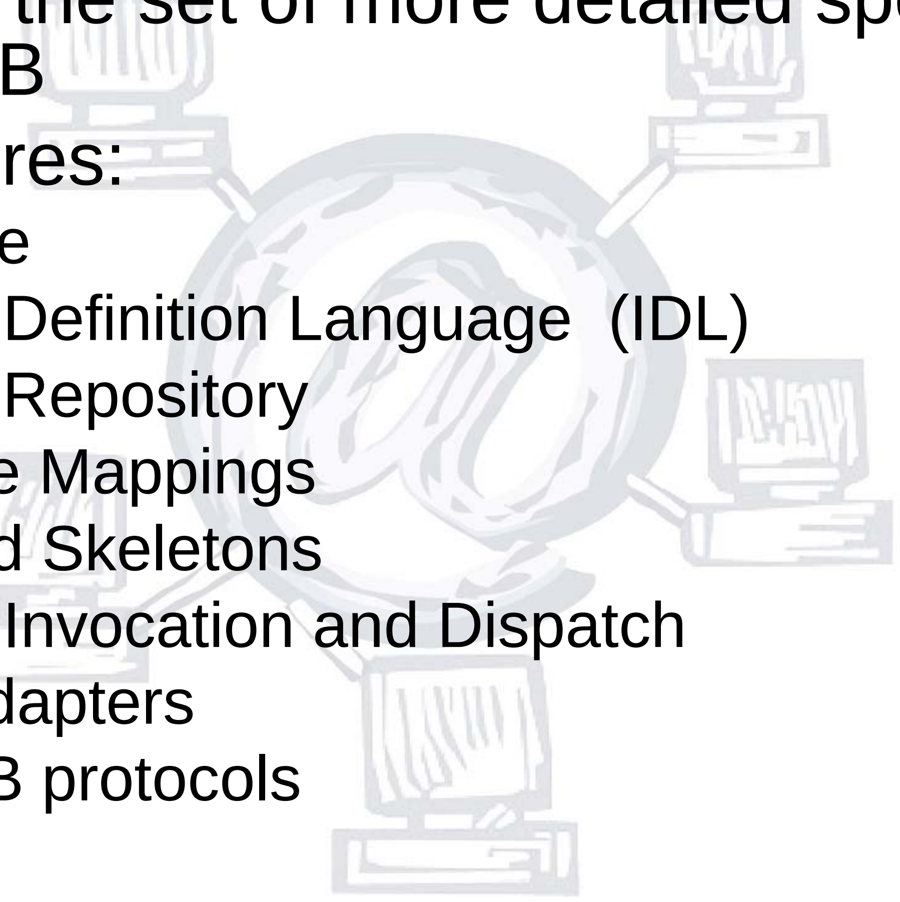
Common Object Request Broker Arch. (CORBA)

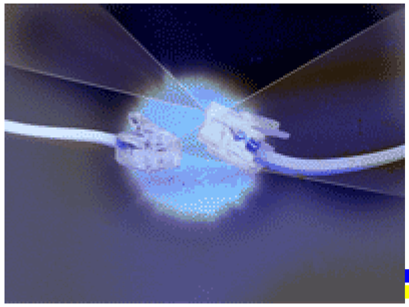
- Set of specs developed by Object Management Group (OMG)
  - Consortium of over 700 developer, vendor, and end user groups
- OMG created an Object Management Architecture (OMA)
- One component of OMA is the Object Request Broker (ORB)
  - Responsible for facilitating remote communications



CORBA (cont.)

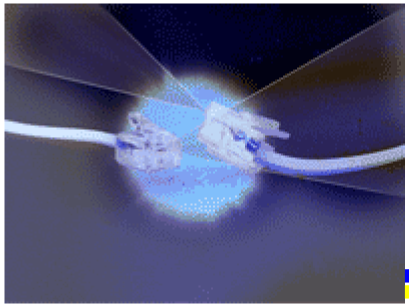
- CORBA is the set of more detailed specifications for the ORB
- Main features:
  - ORB Core
  - Interface Definition Language (IDL)
  - Interface Repository
  - Language Mappings
  - Stubs and Skeletons
  - Dynamic Invocation and Dispatch
  - Object Adapters
  - Inter-ORB protocols





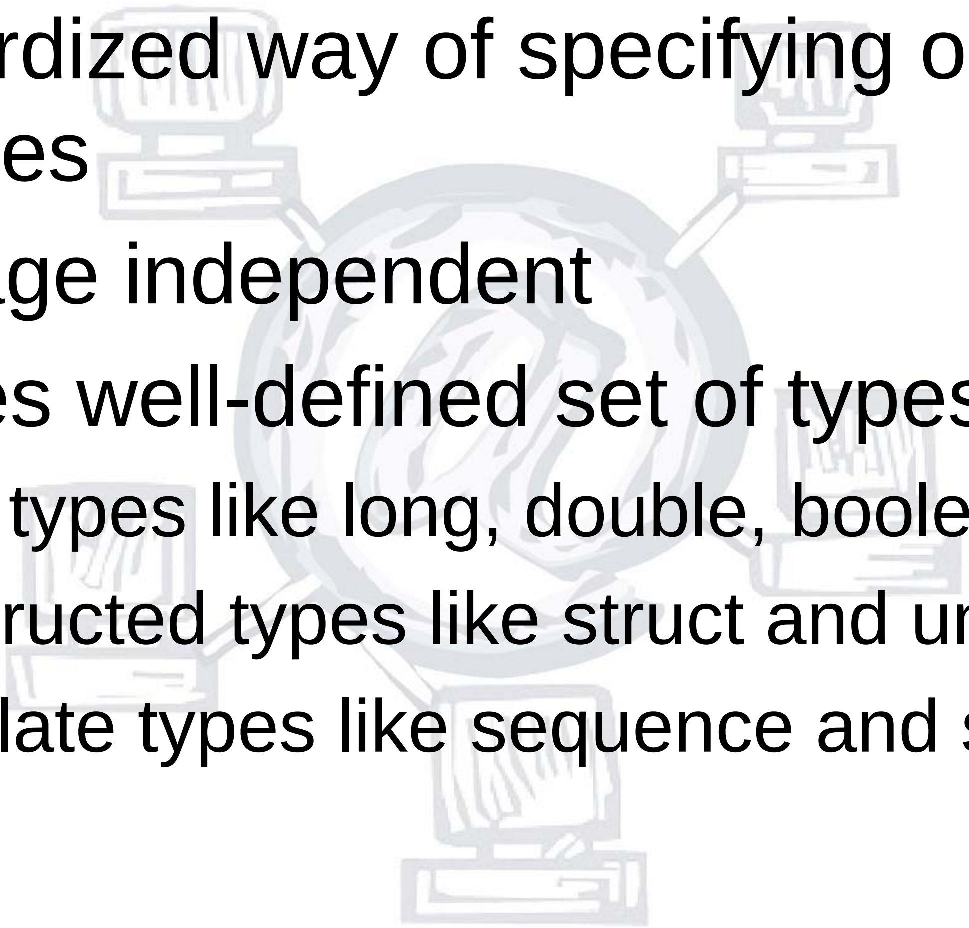
ORB Core

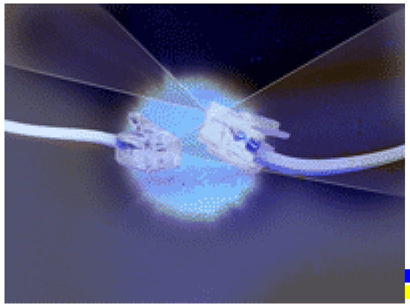
- Similar in function to EJB app server
- Deliver requests to objects, return responses to clients
- Hides details of object
  - Location
  - Implementation (language, OS, hardware)
  - Execution state (active, suspended, deleted)
  - Communication mechanism



OMG Interface Definition Language (IDL)

- Standardized way of specifying object interfaces
- Language independent
- Provides well-defined set of types
  - Basic types like long, double, boolean
  - Constructed types like struct and union
  - Template types like sequence and string

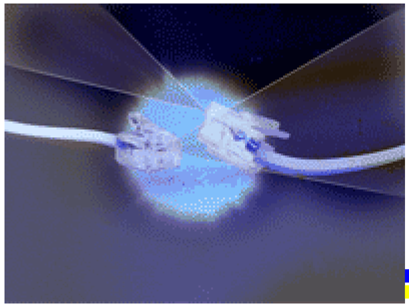




IDL (cont.)

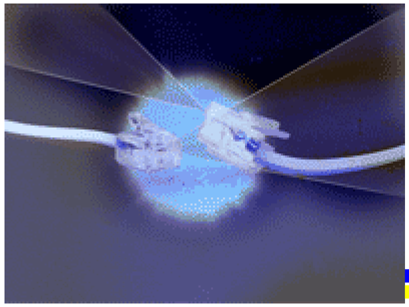
- Types used to define interfaces
- IDL supports inheritance, so interfaces can be defined that extend other interfaces
- IDL also supports definition of exceptions
- Example:

  -
    ```
    module Stats {
      interface EUStats {
              string getMainLangs(in string countryname);
              long getPopulation(in string countryname);
              string getCapital(in string countryname);
      };
    };
    ```



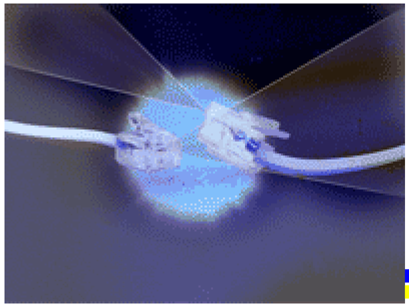
Language Mappings

- Use a tool to create stubs and skeletons for a specific language from IDL
- Defined mappings: Ada, C, C++, COBOL, Java, LISP, Perl, PL/1, Python, Smalltalk, XML, COM bridge
- Mapping CORBA to non-objected oriented languages is difficult and cumbersome to use, but possible



Stubs and Skeletons

- Similar to those used byRMI
- Stub is *proxy* or surrogate for remote object
  - Responsible for marshalling/unmarshalling
  - Provides interface to ORB
- Skeleton is driver for server side object



Inter-ORB Protocols

- Designed for interoperability
  - Guarantee that IDL types and object references are consistent between different implementations
- General InterORB Protocol (GIOP)
  - Transport independent specification
- Internet InterORB Protocol (IIOP)
  - Specifies how GIOP is mapped to TCP/IP