Principles of Distribution Middleware

What is Middleware?

- ***Layered between Application and OS/Network***
- ***Makes distribution transparent***
- ***Resolves heterogeneity of***
  - ***Hardware***
  - ***Operating Systems***
  - ***Networks***
  - ***Programming Languages***
- ***Provides development and run-time environment for distributed systems.***

Categories of Middleware

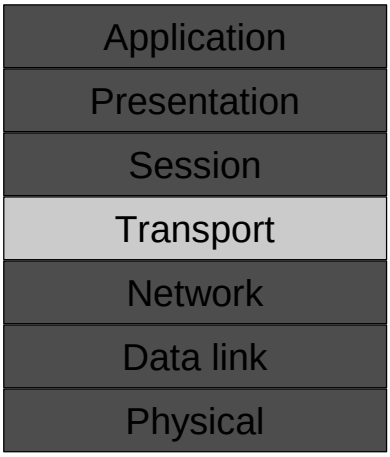
- ***Message-Oriented Middleware***
  - ***IBM MQSeries***
  - ***DEC Message Queue***
  - ***NCR TopEnd***
- ***Transaction-Processing Middleware***
  - ***IBM CICS***
  - ***BEA Tuxedo***
  - ***Encina***
- ***Object-Oriented Middleware***
  - ***OMG/CORBA***
  - ***DCOM***
  - ***Java/RMI***
- ***These are converging! We focus on OO.***

ISO/OSI Reference Model

Application
Presentation
Session
Transport
Network
Data link
Physical

Transport Layer

- ***Level 4 of ISO/OSI reference model.***
- ***Concerned with the transport of information through a network.***
- ***Two facets in UNIX networks:***
  - ***TCP and***
  - ***UDP.***



Transmission Control Protocol (TCP)

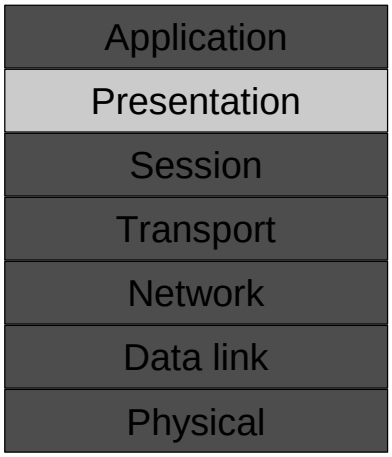
- ***Provides bi-directional stream of bytes between two distributed components.***
- ***UNIX rsh, rcp and rlogin are based on TCP.***
- ***Reliable but slow protocol.***
- ***Buffering at both sides de-couples computation speeds.***

User Datagram Protocol (UDP)

- *Enables a component to pass a message containing a sequence of bytes to another component.*
- *Other component is identified within message.*
- *Unreliable but very fast protocol.*
- *Restricted message length.*
- *Queing at receiver.*
- *UNIX rwho command is UDP based.*

ISO/OSI Presentation Layer

- *At application layer: complex data types*
- *How to transmit complex values through transport layer?*
- *Presentation layer issues:*
  - *Complex data structures and*
  - *Heterogeneity.*



Complex Data Structures

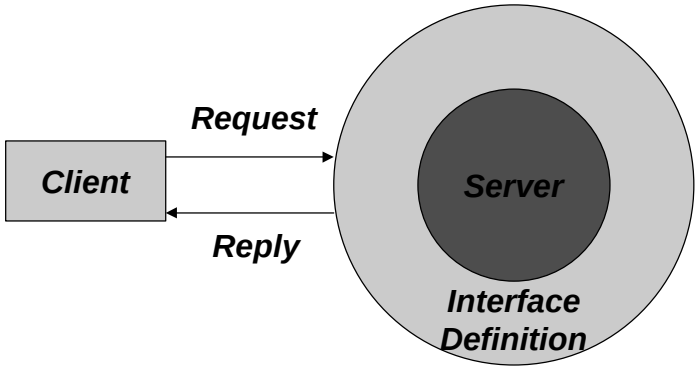
- *Marshalling: Disassemble data structures into transmittable form*
- *Unmarshalling: Reassemble the complex data structure.*

```
class Person {
 private:
  int dob;
  char * name;
 public:
  char * marshal() {
    char * msg;
    msg=new char[strlen(name)+10];
    sprintf(msg,"%d,%s,%d", dob,
            strlen(name),name);
    return(msg);
  };
};
```


Type Safety

- *How can we make sure that*
  - *servers are able to perform operations requested by clients?*
  - *actual parameters provided by clients match the expected parameters of the server?*
  - *results provided by the server match the expectations of client?*
- *Middleware acts as mediator between client and server to ensure type safety.*
- *Achieved by interface definition in an agreed language.*

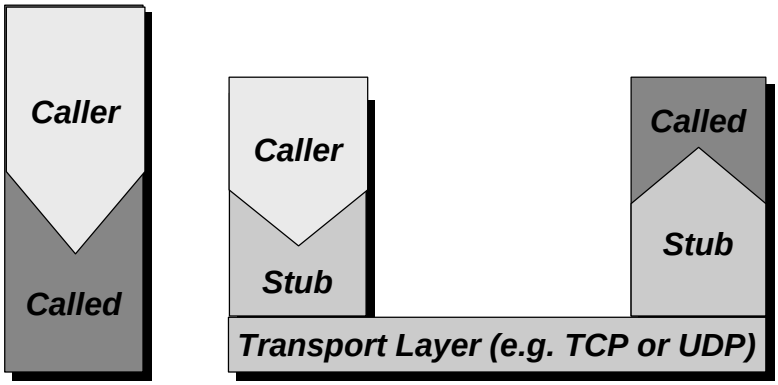
Facilitating Type Safety



Stubs

- Creating code for marshalling and unmarshalling is tedious and error-prone.
- Code can be generated fully automatically from interface definition.
- Code is embedded in stubs for client and server.
- Client stub represents server for client, Server stub represents client for server.
- Stubs achieve type safety.
- Stubs also perform synchronization.

Local Call vs. Remote Request



Synchronization

- **Goal: achieve similar synchronization to local method invocation**
- **Achieved by stubs:**
  - **Client stub sends request and waits until server finishes**
  - **Server stub waits for requests and calls server when request arrives**

Facilitating Access Transparency

- *Client stubs have the same operations as server objects*
- *Hence, clients can*
  - *make local call to client stub*
  - *or local call to server object*

  without changing the call.
- *Middleware can accelerate communication if objects are local by not using the stub.*

Facilitating Location Transparency

- *Object identity*
- *Object references*
- *Client requests operation from server object identified by object reference*
- *No information about physical location of server necessary*
- *How to obtain object references?*

Higher-level Services

- *Facilitate higher levels of transparency*
- *Are distributed objects, too.*
  - *Location Services provide object references*
    - *Naming*
    - *Trading*
  - *Lifecycle Service*
  - *Replication Service*
  - *Concurrency Control Service*
  - *Transaction Service*
  - *Management Service and others…*