CS343 - Operating Systems

Module-3B

IPC in Client Server Systems

Dr. John Jose

Assistant Professor

Department of Computer Science & Engineering

Indian Institute of Technology Guwahati, Assam.

http://www.iitg.ac.in/johnjose/

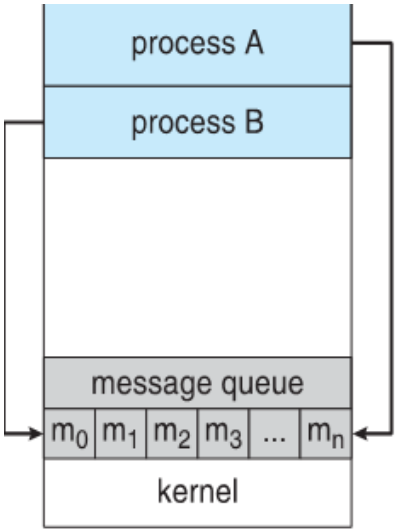
Session Outline

- Review of Inter Process Communication (IPC)
- Local Procedure Calls
- Sockets
- Remote Procedure Calls
- Pipes
- Remote Method Invocation

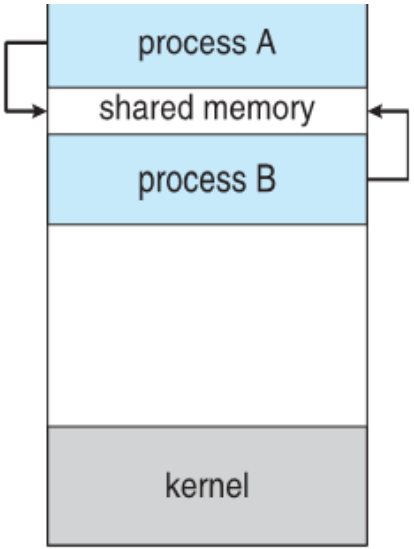
Communications Models

- Cooperating processes need **interprocess communication** (**IPC**)
- Two models of IPC:

Message passing

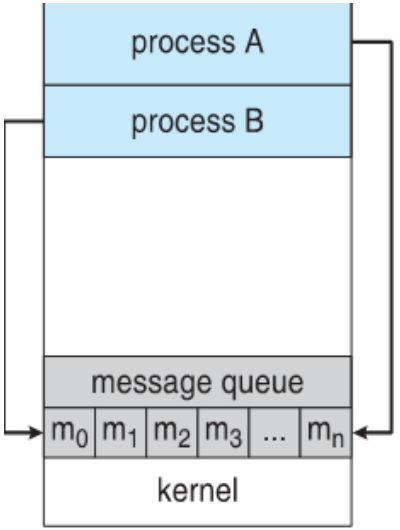


Shared memory

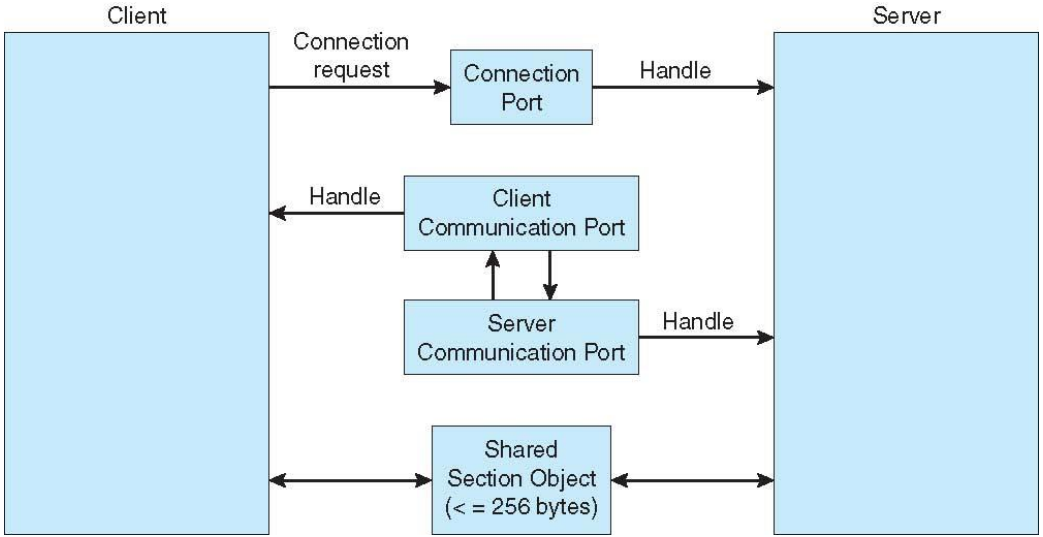


IPC – Message Passing

- Mechanism for processes to communicate and to synchronize their actions
- IPC using message passing facility provides two operations:
  - **send**(message)
  - **receive**(message)
- The message size is either fixed or variable
- Design Issues
  - Direct or indirect
  - Synchronous or asynchronous
  - Automatic or explicit buffering



Local Procedure Calls in Windows



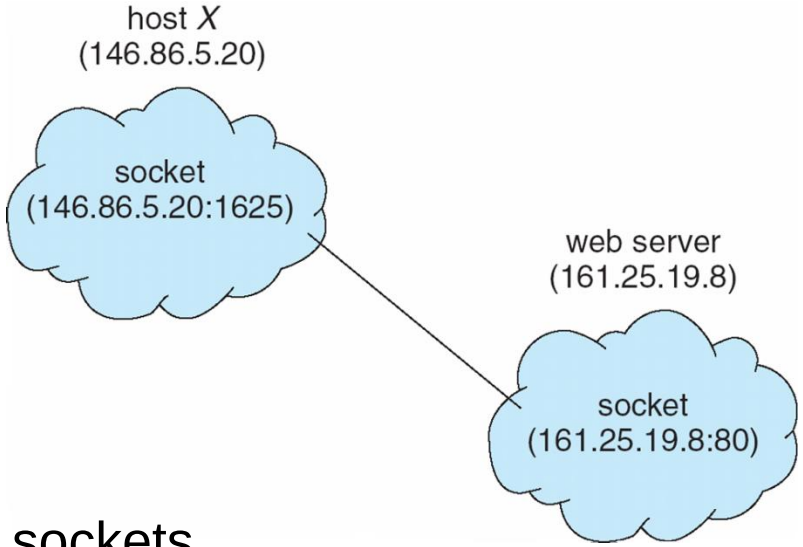
Communications in Client-Server Systems

- Sockets
- Remote Procedure Calls
- Pipes
- Remote Method Invocation

Sockets

- A **socket** is defined as an endpoint for communication
- Concatenation of IP address and **port** – a number included at start of message packet to differentiate network services on a host
- The socket **161.25.19.8:1625** refers to port **1625** on host **161.25.19.8**
- Communication consists between a pair of sockets
- All ports below 1024 are **well known**, used for standard services
- Special IP address 127.0.0.1 (**loopback**) to refer to system on which process is running

Socket Communication

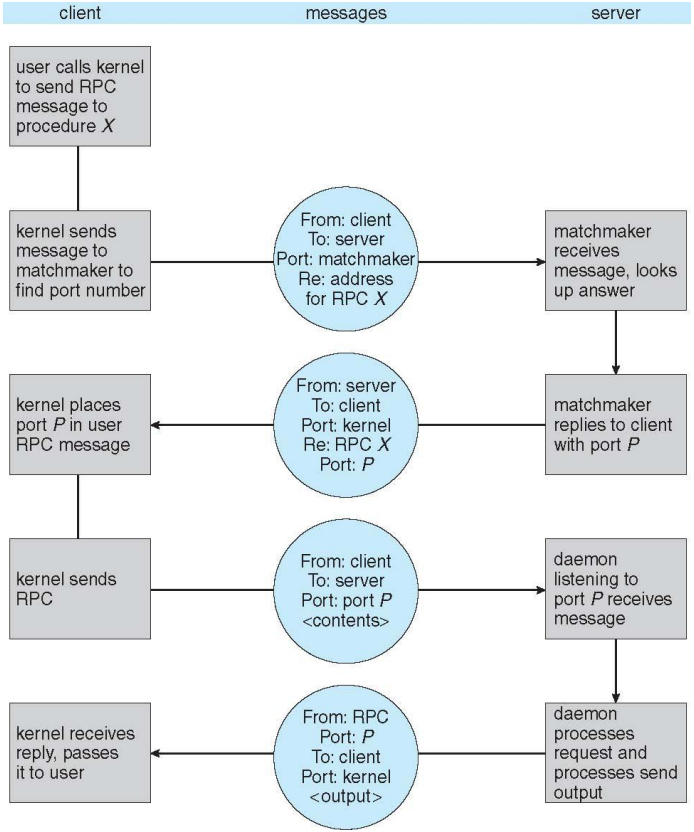


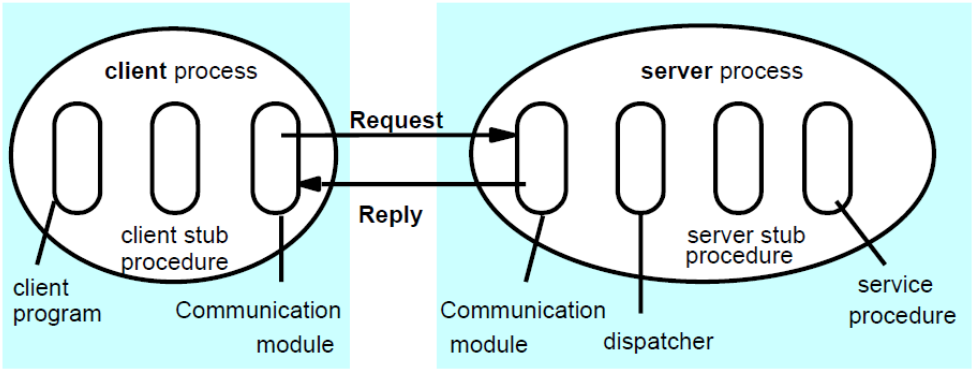
- Three types of sockets
  - **Connection-oriented** (**TCP**)
  - **Connectionless** (**UDP**)
  - **MulticastSocket** class– data can be sent to multiple recipients

Remote Procedure Calls

- Remote procedure call (RPC) abstracts procedure calls between processes on networked systems
- RPC uses ports for service differentiation
- **Stubs** – client-side proxy for the actual procedure on the server
- The client-side stub locates the server and **marshalls** the parameters
- The server-side stub receives this message, unpacks the marshalled parameters, and performs the procedure on the server

Execution of RPC



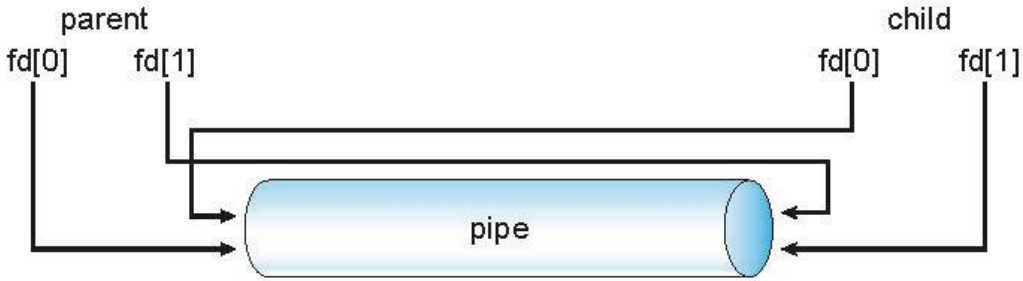


Pipes

- Acts as a conduit allowing two processes to communicate
- Issues:
  - Is communication unidirectional or bidirectional?
  - In the case of two-way communication, is it half or full-duplex?
  - Must there exist a relationship (i.e., **parent-child**) between the communicating processes?
  - Can the pipes be used over a network?
- Ordinary pipes – cannot be accessed from outside the process that created it. Typically, a parent process creates a pipe and uses it to communicate with a child process that it created.
- Named pipes – can be accessed without a parent-child relationship.

Ordinary Pipes

- Ordinary Pipes allow communication in standard producer-consumer style
- Producer writes to one end (the **write-end** of the pipe)
- Consumer reads from the other end (the **read-end** of the pipe)
- Ordinary pipes are therefore unidirectional
- Require parent-child relationship between communicating processes

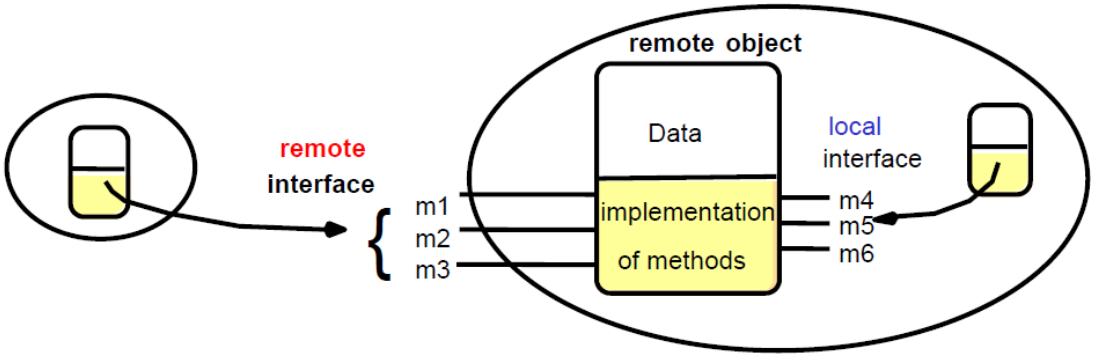


Named Pipes

- Named Pipes are more powerful than ordinary pipes
- Communication is bidirectional
- No parent-child relationship is necessary between the communicating processes
- Several processes can use the named pipe for communication
- Provided on both UNIX and Windows systems

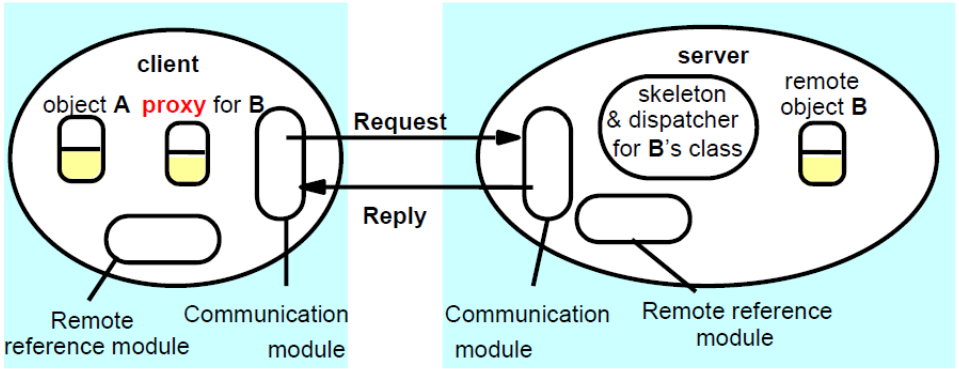
Remote Method Invocation

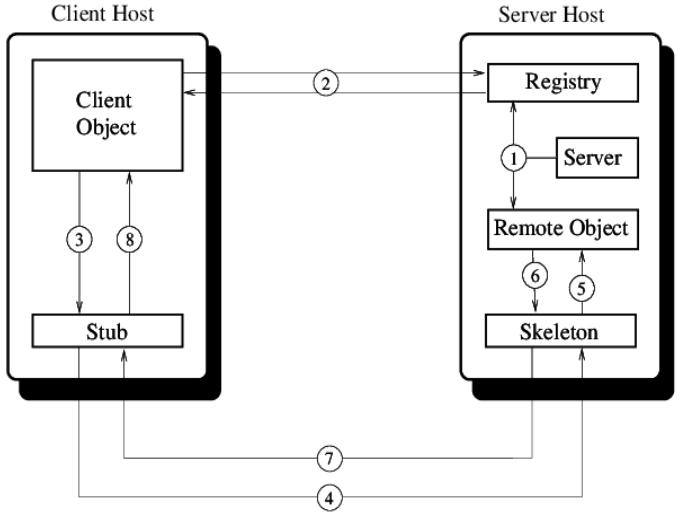
- RMI is a Java feature similar to RPCs.
- Allows a thread to invoke a method on a remote machine.
- RMI can be between two methods under two JVMs in the same machine.



Remote Method Invocation

- Client
- Stub and skeletons
- Parcel – remote method + marshalled parameters
- RMI registry





johnjose@iitg.ac.in
http://www.iitg.ac.in/johnjose/