CS420: Operating Systems

Client/Server Communication

James Moscola
Department of Physical Sciences
York College of Pennsylvania
Communications in Client-Server Systems

• Three additional strategies for communicating between processes (local and/or remote processes)
  - Sockets
  - Remote Procedure Calls
  - Pipes
Sockets

- **A socket** is defined as an endpoint for communication
- A socket is a concatenation of IP address and a port
- The socket **161.25.19.8:1625** refers to port **1625** on host **161.25.19.8**
- Communication links exist between a pair of sockets
  - All connections must be unique
Socket Communication

host X
(146.86.5.20)
socket
(146.86.5.20:1625)

web server
(161.25.19.8)
socket
(161.25.19.8:80)
Remote Procedure Calls

- Remote procedure call (RPC) abstracts procedure calls between processes on networked systems
  - Allows a client to invoke a procedure on a remote host the same as it would locally

- Stubs – client-side proxy for the actual procedure on the server
  - Typically, a separate stub exists for each unique remote procedure

- 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

- User calls kernel to send RPC message to procedure X
- Kernel sends message to matchmaker to find port number
- Kernel places port P in user RPC message
- Kernel sends RPC
- Kernel receives reply, passes it to user
- From client to server:
  - Port: matchmaker
  - Re: address for RPC X
- Matchmaker receives message, looks up answer
- Matchmaker replies to client with port P
- From server to client:
  - Port: kernel
  - Re: RPC X
  - Port: P
- Daemon listening to port P receives message
- Daemon processes request and processes send output
- From RPC to client:
  - Port: P
  - To: client
  - Kernel <output>
Pipes

- Pipes serve as a conduit allowing two processes to communicate

- Implementation 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?

- Two common types of pipes, ordinary pipes and named pipes
Ordinary Pipes

- **Ordinary pipes** allow communication in standard producer-consumer style
  - Opened and treated similarly to a file

- 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 **unidirectional**

- Ordinary pipe cannot be accessed from outside the process that creates it
  - Requires 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