Client/Server Software Engineering

Andrew R. Dalton
Appalachian State University
10 September 2003
What are Clients and Servers?

- Clients and Servers are independent programs that communicate to accomplish a task.
- Servers typically are started first and wait for an incoming request from a client and continue to operate after a request is processed.
- Clients typically contact servers to perform a task, then die.
Communication?

- The communication between clients and servers can be implemented in numerous ways.
  - Inter-Process Communication Mechanisms where the client and server reside on the same machine.
    - Pipes
    - FIFOs
    - Domain Sockets
    - Shared Memory
  - Networks where the client and server reside different machines (Sockets, RPC, CORBA)
Inter-Process Communication

- Client/Server architectures can be implemented using inter-process communication.
- IPC implementations do not allow for distribution of the work to be done among multiple systems.
- Implementation using these mechanisms are less portable than network solutions.
- Implementations are dependent upon Operating System specific support for IPC.
Network Communication

- Networked client/server architectures make client/server applications much more flexible and portable.
- Protocols are established to govern the communication between the client and the server.
- The client and server can be on different machines of different architectures and different operating systems.
- Can reuse existing inter-computer network protocols.
Networks Protocols

• The two primary protocols clients and servers are built on are Transmission Control Protocol (TCP) and User Datagram Protocol (UDP).

• It is important to understand the behavior of the underlying network protocols prior to the development of a client/server application that depends on them.
Transmission Control Protocol (TCP)

- TCP is a reliable connection-oriented network protocol.
  - “Reliable” means that the protocol itself handles errors, lost packets, duplicated packets, out-of-order packets, etc...
  - “Connection-Oriented” means that there is a concept of a connection. There is an overhead involved with the establishment (Three-Way Handshake) and end of the connection.

- Typically used for client/server applications where it is important that no data is lost.
User Datagram Protocol (UDP)

- UDP is an unreliable connectionless network protocol.
  - Applications using UDP do not have to pay the overhead cost of the establishment and end of any “connections.”
  - Applications do not benefit from the reliability of TCP (packets can be lost, received out-of-order, duplicated, etc..)
  - Applications that use UDP typically do not need the reliability of TCP and can live with the potential errors introduced on the network.
Daytime Server

• A “daytime” server is a server that typically listens to a well-known port (13) for connections and returns the current system's time in human-readable format to the connector.

• Typically systems support both a TCP and UDP version of this server; however, many system administrators disable this service for security reasons.

• We will use this as an example throughout this presentation.
Daytime Server (cont.)

```
> telnet localhost daytime
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
09 NOV 2003 22:24:35 EST
Connection closed by foreign host.
```
Implementation of a TCP Daytime Server

- See example code.
- Why does the example code not use the well-known daytime port 13?
- What are the possible shortfalls of this design?
- How can these shortfalls be resolved?
Implementation of a TCP Daytime Client

- See example code.
- See interaction between example TCP daytime server and client.
- See interaction between system TCP daytime server and example client.
- Is the output the same? If not, why?
Implementation of a UDP Daytime Server

- See example code.
- Why does the example code not use the well-known daytime port 13?
- What are the possible shortfalls of this design?
- How can these shortfalls be resolved?
Implementation of a UDP Daytime Client

- See example code.

- Why must the client send an initial packet to the server?

- See interaction between example UDP daytime server and client.

- See interaction between system UDP daytime server and example client.

- Is the output the same? If not, why?
Software Components for Client/Server Systems

- Client/Server applications are typically not seen as monolithic applications. They tend to be broken down into distinct subsystems:
  - User Interaction/Presentation Subsystem (client)
  - Application Subsystem (client or server)
  - Database Management Subsystem (server)
- “Middleware” consists of common software that exists on both the client and the server.
“Fat” and “Thin” Clients

• The application subsystem implements the requirements of the application.
• This can be done either on the server or the client.
• When the bulk of the implementation is done on the client, we call it a “Fat” client.
• When the build of the implementation is done on the server, we call the client a “Thin” client.
• Is our daytime client a “fat” or “thin” client?
Homework

• Develop a high-level design of an instant messenger client/server architecture.
  - Be prepared to be able to discuss what subsystems might be involved and how those subsystems would be distributed ("Fat" client or "Thin" client?)
  - How does your design scale? Maybe it works well for a few dozen users, but what about thousands (or more)?