Introduction to Client Server Systems

Michael E. Spring
Department of Information Science and Telecommunications
University of Pittsburgh
Pittsburgh, Pa 15260

Overview

• Distributed computing
  • Technological underpinnings
  • Focus
  • The basic paradigm
• Important concepts
  • Ports
  • Processes
  • Pipes
  • Protocols
• Important architectures
  • Two tier
  • Three tier
  • N-tier
• Tools

Client-Server Computing in Context

• Client Server computing is simply another tool in the evolution of computing tools:
  • Mainframe batch jobs
  • Time shared
  • Personal
  • Network based
  • Distributed
• Client-server computing is driven by
  • Technological capability – networks and workstations
  • Business needs – reengineering to incorporate IT
• Client-server computing demands
  • New knowledge and perspective from developers
  • New understandings by users
Distributed Computing

- A model or mode of computing that distributes functionality across systems
- Distributed computing is concerned with:
  - Authentication
  - Concurrency
  - Naming
  - Reliability
- Distributed computing can take numerous forms:
  - Master-slave
  - Peer to peer
- The most common paradigm for distributed computing to date is client server

Technological Underpinnings

- Desktop power and demands
  - GUI expectations
- Networking
  - Ubiquity of the Internet – one addressing scheme (IP)
  - Transparency of connections (TCP)
- Database sophistication and simplicity
  - Transaction management
  - Data distribution
  - Standard Query Language (SQL)

Technological Maturation of Client-Server

- Standards made it easier
  - Network support (TCP/IP)
  - Data access (SQL)
  - Programming support (RPC, RMI, CORBA)
  - Interface support (Java/XML)
- The Web made it understandable
  - A form is an input mechanism
  - CGI provides access to a process
  - A web page is an output mechanism
Technological Goals
Client-Server

- Workload distribution -- became important
  - Hub congestion
  - Edge underutilization
- Database isolation -- became essential
  - Data integrity
  - Transaction management
  - Data distribution

Important Standards

- Networking
  - IP
  - TCP
  - DNS
  - CORBA/DCOM/RPC/RMI/WebServices
- Data interchange
  - SQL
  - ODBC
  - XDR/ASN.1
  - Java/XML

Specific Enabling Technologies

- Network Access
  - Sockets
- Location of machines and Processes
  - IP addresses and Ports
- Reliable transport of messages
  - UDP and TCP
- Control of Processes
  - Forking
  - Signaling
  - Semaphores
  - Locks
  - Blocks
Multiple Layers

- Client-server computing involves
  - low level protocols
    - Addressing protocols – for multicasting and streaming
    - Transport protocols – for specialized information
  - application protocols **the focus of this course**
  - SQL/ODBC connections
- Selecting the application layer protocol, there are numerous levels of dealing with client server:
  - Hand rolled protocols and custom clients and servers
  - Standards based development of clients or servers
  - API/wrapper based approaches to interfaces
    - RPC and RMI
    - CORBA, DCOM, etc.

The Basic Client Server Paradigm

- The basic design for client-server systems involves two independent and autonomous processes: clients and servers.
  - **The client** initiates a request from a server
  - **The server** listens for a request from client processes and responds with the service it provides
- There are multiple variations on the theme
  - A server may serve one client at a time, or multiple clients “concurrently”
  - A client can make connects to multiple servers at once
  - A client may be a server to some other process and a server may be a client to some other process

The Basic Paradigm

The Basic Paradigm
**Ports, Processes, Pipes, and Protocols**

- To understand client server, it helps to understand the four P’s
  - A port refers to a machine address may be used for communication purposes (input and output)
  - A process is a set of instructions and the associated data arrayed in memory to undertake some task
  - A pipe is a communications channel between two processes
  - A protocol is an agreed up set of conventions between two communicating entities

---

**Ports**

- A machine on the internet is identified by an address. At the local area network level this address is a unique machine address – known as a MAC address
- A machine may also have an Internet Protocol (IP) address. This address has three forms:
  - A binary number between 0 and $2^{32}$
  - A dotted decimal form such as 136.142.116.68
  - A domain name form such as spring.sis.pitt.edu
- An IP address has 64,000 “subaddresses” or ports
  - Ports 1-128 are called the well known ports and are reserved for services such as ftp, telnet, smtp, and http.
  - Ports below 1024 are reserved to the machines sys admin
- Clients and servers communicate between specific machines on specific ports

---

**Processes**

- A process is an address space that contains data, instructions and resources for performing a computing task including but not limited to an application program
  - Server and client processes may execute on the same or different computing systems
  - It is not uncommon for a server to actually consist of multiple processes listening to the
  - Processes are owned by other processes
  - Processes have a variety of mechanisms for communication including locks, semaphores and most notably pipes
Pipes

• Any two processes may be connected by a pipe
• A pipe is a Unix concept where the output of one process is accepted as input by another
• For client server, a connection between two processes on different machines is made through the use of sockets
  • A socket is similar to a file handle in Unix
  • The connection between the two sockets is a pipe
• It is important to remember that no action is taken by the operating system on the contents of a pipe.
  • What comes out of a pipe is exactly what goes in.

Protocols

• A protocol is simply a way of doing something
  • Who speaks first
  • Who speaks last
  • What kinds of questions can be asked
  • What the format of the question should be
  • How errors will be handled
• In client server, we refer to many things as protocols:
  • The way of identifying a machine address (IP)
  • The way of ensuring reliable transmission (TCP)
  • The way of communicating (FTP, HTTP, SMTP, etc.)
• The focus of this course is on understanding the way to construct protocols of the last form

Architectural Variety

• The simplest model for a client server architecture is what is called a two tier model.
  • FTP is a two tier model
  • The world wide web is a two tier model
• Most business applications use a three tier model
  • The client handles the user interface
  • The second tier, handles the business rules
  • A third tier, accessible only through the second, handles the data storage
• New models for the web use an n-tier approach where multiple layers are created to handle different processes
Two Tier Architecture

- Generally, client and server processes execute on separate computer systems connected by a network
- The client process:
  - Connects to a known server process
  - Requests specific services from the server process
  - Is generally easier to conceptualize
  - Generally requires more code than the server
- The server process:
  - Provides services requested by a client process
  - May also operate as a client to another server
  - Is hard to conceptualize when concurrent models are used where the server responds to multiple clients in parallel
  - Is hard to write in that it must anticipate malicious or simply malformed as well as legitimate requests

Two Tier Schematic

Three Tier Architecture

- A three tier client server system has an intermediate layer that addresses some issues
- Three main components:
  - Client: a computer process that initiates a request from a server process through a front-end application
  - Server: a computer process that listens for a request from client processes, gathers the needed information to respond to the request, and processes that data in accord with appropriate rules before responding to the client
  - Database: a computer process that listens to the server for requests for data and returns the requested data
• The separation of processing tasks is the key difference between client/server computing and mainframe computing.

• Generally speaking, the client is responsible for managing user input and information display. However, the client may also do data manipulation when the cost of retransmission of data is high.

• In an N-tier system, communications middleware can support the transmission and translation of messages – e.g. proxy servers, and protocol servers.

• The backend of an n-tier system can support authentication, transaction management, and other specialized functions.
C/S Development Tools -- General

• Complete System Development Life Cycle
  • Team development support
  • Support for a variety of development tools (CASE, LIBRARIES, Generators, etc.)
  • Rapid Application Development (RAD)
  • Joint Application Development (JAD)
• Interface Development Environment
  • Automatic generation of framework
  • Standardized data marshalling
• Middleware Protocol support (SQL, ODBC)
• Middleware network support (TCP/IP, XDR, RPC/RMI)

C/S Development Tools Client

• GUI based development
  • Visual basic
  • C++
  • Java
• GUI builder for multiple platforms
  • Netbeans
• Object-oriented development with code re-usability occurs with all these systems

C/S Development Tools Server

• Debugger capable of threads and multiple processes
• Data directory/dictionary with central repository control is important as is multiple DB support
• Data access regardless of DB platform
The Case For Client-Server

- Distributed work needs distributed computing
- Makes it easier to support different client platforms
- Allows access to data by end users on personal devices
- Helps to utilize desktop devices productively
- Gives end users more control over computing
- Integrates enterprise-wide data sources
- Allows use of personal tools with corporate data
- Claims competitive cost/benefit performance

The Case Against Client-Server

- Application development more complicated
  - Added networking component
  - Testing and debugging are complex
  - Programmers need retraining
- Security is a difficult issue
- Requires a highly reliable ubiquitous network
  - Network failure disables application
  - Disconnected computation is limited
- Consumes a lot of network bandwidth

Business Expectations

- Flexibility and adaptability to change
- Improved employee productivity
- Improved organizational workflow
- New opportunities for competitive advantage
- Improved customer relations and satisfaction
- Systems is flexible and cost effective
Information Systems Expectations

- Systems are platforms independent
- Processing tasks are optimized among systems
- Reduced development and implementation costs
- Reduced development and implementation time
- Extended system life cycle through scalability and portability
- Reduced operational costs
- Changes the MIS emphasis from development to end-user support
- Enhanced information deployment and utilization

Implementing Client-Server Systems

- Weighting the pros and cons is important
  - Client server approaches are not optimal for all applications
- Education is a big part of client server
  - Practitioners need new tools
  - Users need new understandings
- A case needs to be made for client server
  - Business case – transition costs
  - MIS case – new staffing and organization
  - User case – new training and education