Client/Server Programming

Chirag Mehta

Client/Server - The Big Picture

- Client/server as the successor of mainframe computing
- Client/server today: all-inclusive paradigm
  - Distributed and mix-and-match
  - PCs, servers, mainframes, embedded devices
  - Internet and intranets
Client/Server Characteristics

- Concept of a service
  - Clients as consumers of services provided by servers

- Shared resources
  - Learn to share; illusion of dedicated service

- Asymmetrical protocols

- Transparency of location
  - Location, location, location...not!

Client/Server Characteristics

- Mix-and-match
  - Hardware, OS, stacks, protocols, etc.

- Message-based exchanges

- Encapsulation of services

- Scalability and integrity
The Tiers in Client/Server

- Logical partitioning of a client/server system
- In a more general sense, tiers are functional, not physical, parts of the system
- Many “classic” client/server applications are 2-tier
- Sound design principles, performance, scalability, and maintenance consideration require that business logic be given its own tier—the “middle” tier
- Large, complex real-life systems may have many tiers

2-Tier Vs. 3-Tier Comparison

- System administration
- Security
- Encapsulation of data
- Performance
- Scale
- Application reuse
- Ease of deployment
- Legacy integration
- Internet/Web “enablement”
- Heterogeneous data sources
- Communications paradigm choice
- Hardware architecture flexibility
- Availability
Inside Client/Server Building Blocks

- Client building block
  - OS, User Interface (UI)

- Server building block
  - RDBMS, TP Monitors, groupware, distributed objects, web application server

- Middleware building block
  - Runs across clients and servers

Client/Server: What’s In the Middle?

- Middleware: a collection of technologies that serves as a glue that brings the parts of a client/server system together

- It is an enabler: it does not provide “business” services

- Middleware pipes: getting from here to there
  - RPC, ORBs, message queues, event brokers

- Both client-to-server and server-to-server
Client/Server and Operating Systems

What a Server Program Does
- Waits for client-initiated requests
- Executes many requests at the same time
- Takes care of VIP clients first
- Initiates and runs background task activity
- Keeps running
- Grows bigger and fatter
Base OS Services Needed by a Server Program
- Task Preemption
- Task Priority
- Semaphores
- Inter-process Communications (IPC)
  - Local/Remote Inter-process Communications
- Threads
- Inter-task Protection
- Multi-user High-Performance File System
- Efficient Memory Management
- Dynamically Linked Run-Time Extensions

Extended OS Services Needed by a Server Program
- Ubiquitous Communications
- Network Operating System Extensions
- Binary Large Objects (BLOBs)
- Global Directories and Network Yellow Pages
- Authentication and Authorization Services
- System Management
- Network Time Service
- Database and Transaction Services
- Internet Services
- Object-Oriented Services
Clusters

- Group of machines behaving as a single system
- Shared disk
  - Single disk image for all nodes in the cluster
- Shared nothing
  - Loosely coupled nodes

Directories

- Special-purpose database that contains information about resources available on a network
  - Attribute-value pairs: “name = John” and “number = 555-1212”
  - Optimized for reads, much weaker than general purpose database management system
- Objects typically stored in a network directory:
  - Users, applications, printers, file servers, switches, and routers
- X.500, LDAP
Global Directories: Keeping Track of NOS Resources

- E-mail
- Authentication and Security
- Network Management
- Distributed Objects
- File Services

Directory Service

Distributed Security Services

- Client/server environment introduces new security threats
- Network is highly accessible
- Servers expose services via standard protocols and API’s
- Single-system illusion: transparent access to all resources
Security Basics

- Authentication
  - Are you who you claim to be
  - Passwords, digital certificates, Kerberos

- Authorization
  - Are you allowed to use this resource
  - Access Control Lists (ACL)

- Audit Trails
  - Where have you been

Security on the Internet

- Integrity: Is My In-Transit Data Safe
  - Encryption (private key, public key)
  - Checksums (MD5, SHA-1)

- Non-Repudiation: Can You Prove It in Court?
  - Evidence of message creation
  - Evidence of message receipt
  - Timestamp
  - Storage facility
  - Adjudicator
Web Security: SSL

- Secure Socket Layer
  - Applications run unchanged—simply using a different socket implementation
- Transport level
- Encryption, server authentication, message integrity
- SSL is now TLS (Transport Layer Security)

Firewalls

- Packet-filtering firewalls
  - IP level
  - Permit, drop or reject based on IP address (destination and source) and port
  - Can be spoofed
- Proxy firewalls
  - Application level
Firewalls: Proxy/Router Combo

Virtual Private Networks (VPN)
Peer-to-Peer Communications

- Low-level, “close to the wire” communications paradigm
- Network exposed to the programmer
- Originally stack-specific, now typically support multiple stacks
- High-level programs use RPC, MOM, or ORB instead
- Sockets, TLI, NetBIOS, Named Pipes, CPI-C

Sockets

- Generic interface for network communications
- 1981, BSD 4.2
- De facto standard for network applications over TCP/IP
- WinSock, Java Sockets
- Stream (TCP), datagram (UDP), raw (IP)
- Socket = IP address + Port address
- HTTP: www.sjsu.edu:80; telnet: port 21, SMTP: port 25
- TLI (Transport Layer Interface): “cleaner” sockets
RPC

Message Oriented Middleware (MOM)

- Can't assume server is always there
- Mobile computing, inter-organization transfers, EDI, and Branch offices
- Deferred delivery
- Less than 4% today
- Clients must be able to handle responses
- Event-driven computing
- Publish and subscribe
Message Oriented Middleware (MOM)

Relational Database Management Systems
- A core part of many client/server systems
- Well-understood, accepted model
- No one questions the need for an off-the-shelf RDBMS
- 10-15 years ago: IT “cowboys” building home-grown data storage systems
- Unthinkable today
SQL: Structured Query Language

- Interactive query language for ad hoc database queries
- DML: Data Manipulation Language
- Database programming language
- Embedded or called via API
- Data definition and data administration language
- DDL: Data Definition Language
- Data protection language
- Validation, referential integrity, rollback (undo), locking

Database Server

- Responds to client SQL calls
- Manages the database
- Manages concurrent access
- Protects database against internal and external threats
  - Transactions, Locking, Journal log, security, etc.
- Vendor extensions
Embedded SQL API
- ISO SQL-92 defined standard
- Supports multiple languages
- Run SQL source through pre-compiler
- Target database must be available
- Pre-compilers are database-specific

SQL CLI
- No pre-compiler
- Create and execute SQL statements at run time
- More portable applications
- Became basis for ODBC and X/Open CLI
- Common error handling, data types, and system catalogs
What is JDBC?
- A Java API for executing SQL statements against a remote database.
- JDBC != “Java DataBase Connectivity”
  - Not an acronym!? 
  - It is really a trademarked name
- JDBC is a set of classes and interfaces
- Provides a standard API for database development in pure Java

What does JDBC do?
- Three basic things
  - Establish a connection with a database
  - Send SQL statements
  - Process the results
Why not ODBC?

- Open Database Connectivity
- ODBC uses a C interface
- A literal translation of the ODBC C API into a Java API would not be desirable
- ODBC is hard to learn

Why JDBC?

- JDBC API is a natural Java interface to the basic SQL abstraction and concepts.
- Java API like JDBC is needed in order to enable a “pure Java” solution.
- JDBC-ODBC bridge driver comes JDK1.1
- A low-level API and a base for higher-level APIs.