The Client Server Model
and Software Design

Prof. Chuan-Ming Liu
Computer Science and Information Engineering
National Taipei University of Technology
Taipei, TAIWAN
Introduction

- Short review for OSI model
- Peer-to-peer communication
- Client-server paradigm
Layering

- Divide a task into pieces and then solve each piece independently (or nearly so).
- Establishing a well-defined interface between layers makes porting easier.

Major Advantages:
- Code Reuse
- Extensibility
Layering Example: Federal Express

- Letter in envelope, address on outside
- FedX guy adds addressing information, barcode.
- Local office drives to airport and delivers to hub.
- Sent via airplane to nearest city.
- Delivered to right office
- Delivered to right person
FedX Layers

Letter → Addressed Envelope → Truck → Airport

NTUT, TAIWAN
Layered Software Systems

- Network software
- Operating systems
- Windowing systems
Unix is a Layered System

- Applications
- Libraries
- System Calls
- Kernel
OSI Reference Model

The *International Standards Organization* (*ISO*) proposal for the standardization of the various protocols used in computer networks (specifically those networks used to connect open systems) is called the *Open Systems Interconnection Reference Model* (1984), or simply the *OSI model*. 
OSI Model

Although the OSI model is a just a model (not a specification), it is generally regarded as the most complete model (as well it should be - nearly all of the popular network protocol suites in use today were developed before the OSI model was defined).
OSI <-> Network Software

Although this course is about network programming (and not about networking in general), an understanding of a complete network model is essential.
OSI 7 Layer Model:

1. Physical
2. Data-Link
3. Network
4. Transport
5. Session
6. Presentation
7. Application

- High level protocols
- Low level protocols
Simplified Network Model

- Process
- Transport
- Network
- Data Link

Interface Protocols

Peer-to-peer Protocols
What’s a Protocol?

- An agreed upon convention for communication.
- Both endpoints need to understand the protocol.
- Protocols must be formally defined and unambiguous!
- We will study lots of existing protocols and perhaps develop a few of our own.
Interface protocols describe the communication between layers on the same endpoint.

Peer-to-peer protocols describe communication between peers at the same layer.
Physical Layer

- Coordinates the functions required to transmit a bit stream over a physical medium

- Concerns
  - Physical characteristics of interfaces and medium
  - Representation of bits
  - Data rate (Transmission rate): bits/sec
  - Synchronization of bits
Physical Layer

- **Line configuration**
  - Point-to-point
  - Multipoint

- **Physical topology**
  mesh, star, ring, or bus.

- **Transmission mode**
  simplex, half-duplex, or full-duplex
Physical Layer

From data link layer:
- L2 data: 10101000000010

To data link layer:
- L2 data: 10101000000010
Data Link Layer

- Transforms the physical layer to a reliable link
- Makes the physical layer appear error free to upper layer
- Responsible for
  - Framing frames
  - Physical addressing (physical address)
    - Header defines the sender and/or receiver
    - Receiver is the device connected to the next
Data Link Layer

- Flow control
- Error control
- Access control

Multi-home: computer having two or more NICs
Data Link Layer

From network layer:
- L3 data
  - Data link layer
    - T2
    - H2
  - Frame
    - L2 data
      - To physical layer

To network layer:
- L3 data
  - Data link layer
    - T2
    - H2
  - Frame
    - L2 data
      - From physical layer
Node-to-node Delivery
Network Layer

- Responsible for the source-to-destination delivery of a packet

Responsibilities:
- Logical addressing (IP address)
- Routing
Network Layer

From transport layer

L4 data

Network layer

H3

Packet

L3 data

To data link layer

To transport layer

L4 data

Network layer

H3

Packet

L3 data

From data link layer
End-to-end Delivery
Transport Layer

Source-to-destination (end-to-end) delivery of the entire message

Functions include:

- Service-point addressing (port)
- Segmentation and reassembly
- Connection control
  - connectionless v.s. connection-oriented
- Flow control
- Error control
Transport Layer
Reliable end-to-end delivery of a message
Session Layer

Network dialog controller establishes, maintains, and synchronizes the interaction between communicating systems.

Functions include:
- Dialog control
- Synchronization (checkpoint)
Session Layer
Presentation Layer

Concerned with the syntax and semantics of the information exchanged between two systems.

Responsibilities include:
- Translation
- Encryption
- Compression
Presentation Layer

From application layer

L7 data

Encoded, encrypted, and compressed data

H6

L6 data

To session layer

To application layer

L7 data

Decoded, decrypted, and decompressed data

H6

L6 data

From session layer
Application Layer

Interface for users to access the network

Services include

- Network virtual terminal
- File transfer, access, and management (FTAM)
- Mail services
- Directory services
Application Layer

User

Application layer

X.500  FTAM  X.400

L7 data

To presentation layer

User

Application layer

X.500  FTAM  X.400

L7 data

From presentation layer
Summary of Layers

- **Application**: To allow access to network resources
- **Presentation**: To establish, manage, and terminate sessions
- **Session**: To move packets from source to destination; to provide internetworking
- **Transport**: To transmit bits over a medium; to provide mechanical and electrical specifications
- **Network**: To organize bits into frames; to provide hop-to-hop delivery
- **Data link**: To process-to-process message delivery and error recovery
- **Physical**: To translate, encrypt, and compress data
TCP/IP Protocol Suite

- **Physical Layer**
- **Data link Layer**
- **Network Layer**
- **Transport Layer**
- **Application Layer**

Correspond to OSI model

Last three layers in OSI

No specific protocol defined in physical and data link layers in TCP/IP
TCP/IP and OSI model

Application

SMTP  FTP  DNS  SNMP  NFS  RPC  TFTP

Presentation

Session

Transport

TCP  UDP

Network

ICMP  IGMP  IP  RARP  ARP

Data link

Protocols defined by the underlying networks

Physical
Network Layer

- Internetwork layer
- Support
  - IP: Internetworking Protocol
  - ARP: Address Resolution Protocol
  - RARP: Reverse Address Resolution Protocol
  - ICMP: Internet Control Message Protocol
  - IGMP: Internet Group Message protocol
Internetworking Protocol (IP)

- Transmission mechanism
- *Datagram*: data unit to be sent in IP
- Unreliable and connectionless
- *Best-effort* delivery service
- Host-to-host protocol
Transport Layer

TCP and UDP: delivery of a message from a process to another process

User Datagram Protocol (UDP)
Transmission Control Protocol (TCP)
TCP/IP

- Provides basic mechanisms used to transfer data
- Allows a programmer to establish communication between applications
- Provides peer-to-peer communication
- One organizational method to use TCP/IP is the client-server paradigm
Motivation

Scenario: A user tries to start two programs on separate machines and have them communicate.

Program 1 starts
Send message to its peer

No connection can be set up
No response; exit

Program 2 starts
Not running; Refuse
Client-Server Model

One side in any pair of communicating application must start execution and wait for the other side to contact it.

Since the client-server model places responsibility for rendezvous problem on application, TCP/IP does not need to provide mechanisms that automatically create a running program when a message arrives. Instead, a program must be waiting to accept communication before any request arrive.
Terminology and Concepts

- Clients and Servers
- Privilege and Complexity
- Standard v.s. Nonstandard Client Software
- Parameterization of Clients
- Connectionless v.s. Connection-oriented Servers
- Stateless v.s. Stateful Servers
- Identifying a Client
Clients and Servers

**Client**

* An application that initiates peer-to-peer communication, e.g. web browser
* Easier to build than servers
* System privileges usually unnecessary

**Server**

Program that waits for incoming communication requests from a client
Privilege and Complexity

Server software often needs to access objects that OS protects.

Server cannot rely on the usual OS check since its privilege status allows access to any file.

Security issues:
- Authentication
- Authorization
- Data Security
- Privacy
- Protection
Privilege and Complexity

The combination of special privileges and concurrent operation usually makes servers more difficult to design and implementation.
Standard v.s. Nonstandard Client Software

**Standard application services**
- Defined by TCP/IP
- Assigned well-known, universally recognized protocol port

**Non-standard application services**
- All other services which is not standard
- Or, locally-defined application services
Standard v.s. Nonstandard Client Software

Be aware of the standard when outside the local environment

Standard application service examples

- Remote login, TELNET protocol
- E-mail client, SMTP or POP protocol
- File transfer client, FTP protocol
- Web browser, HTTP protocol
Standard vs. Nonstandard Client Software

Non-standard application service examples

- Music or video transfer
- Voice communication
- Distributed database access
Parameterization of Clients

Generality for client software
e.g. TELNET protocol using port number

Fully parameterized client
application allows more input parameters
Parameterization of Clients

When designing client application, include parameters that allow the user to fully specify the destination machine and port number.
Connectionless v.s. Connection-Oriented Servers

TCP/IP provides two types of interaction between client and server:
- Connectionless style
- Connection-oriented style

The distinction is critical
Connection-Oriented Servers

TCP:

* full reliability
* verifying the data arrives
* automatically retransmits segments
* checksum over the data
* data arriving in order
* no duplicated packets
* control the flow
* reporting the underlying network problem to sender
Connectionless Servers

**UDP**

* no guarantees about reliable delivery
* software contains code to detect and correct errors occurred by transmission
* work well when the underlying network running well
* aware of testing when using UDP
Connectionless v.s.
Connection-Oriented Servers

- TCP is preferable to UDP
  - TCP simplifies programming
  - TCP relieves the programmers of responsibility for detecting and correcting errors
  - Adding reliability to UDP is a nontrivial work
Connectionless v.s. Connection-Oriented Servers

- Application programs use UDP only if
  - Application protocol specifies the UDP must be used
  - Application protocol relies on hardware broadcast or multicast for delivery
  - Overhead for reliability is unnecessary
Stateless v.s. Stateful Servers

**State information**

A server maintains about the status of ongoing interaction with clients.

Servers that do not keep any state information are called *stateless servers*; otherwise, called *stateful servers*. 
Stateless v.s. Stateful Servers

Keeping a small amount of information in a server can reduce the size of messages that the client and server exchange and allow the server to respond quickly.

The point for statefulness is **efficiency**.

The motivation for statelessness lies in **protocol reliability**.
Stateless File Server Example

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>op</td>
<td>Operation (read or write)</td>
</tr>
<tr>
<td>name</td>
<td>Name of the file</td>
</tr>
<tr>
<td>pos</td>
<td>Position in the file</td>
</tr>
<tr>
<td>size</td>
<td>Number of bytes to transfer</td>
</tr>
<tr>
<td>data</td>
<td>Present only in write operation</td>
</tr>
</tbody>
</table>

Message waits for client to access.

File server stores or extracts data from the server.

Client sends message to file server.
Stateful File Server Example

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>op</td>
<td>Operation (read or write)</td>
</tr>
<tr>
<td>name</td>
<td>Name of the file</td>
</tr>
<tr>
<td>pos</td>
<td>Position in the file</td>
</tr>
<tr>
<td>size</td>
<td>Number of bytes to transfer</td>
</tr>
<tr>
<td>data</td>
<td>Present only in write operation</td>
</tr>
</tbody>
</table>

Waits for client to access

File server

disk

reads

stores or extracts data from the server
# Stateful File Server Example

<table>
<thead>
<tr>
<th>Client</th>
<th>File Name</th>
<th>Current Position</th>
<th>Last Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>test.program.c</td>
<td>0</td>
<td>read</td>
</tr>
<tr>
<td>2</td>
<td>tcp.book.text</td>
<td>456</td>
<td>read</td>
</tr>
<tr>
<td>3</td>
<td>dept.budget.txt</td>
<td>38</td>
<td>write</td>
</tr>
<tr>
<td>4</td>
<td>teris.txt</td>
<td>125</td>
<td>read</td>
</tr>
</tbody>
</table>

State information table
Identifying a Client

Stateful servers use two general approaches to identify clients:

- **Endpoints**
- **Handles**

**Endpoint identification**

- Operate automatically
- Relies on transport protocol, not application protocol
- Endpoint information may change
Identifying a Client

server

using the endpoint information to look up the state table

IP address and port number

Transport protocol
Identifying a Client

- Handles
  - remains constant across multiple transport connections
  - A small integer
  - Independent of the underlying transport protocol
    - Change of transport connection does not invalidate handles
    - Visibility to the application – drawback
Identifying a Client

- Back to stateful server
  - Can not retain state forever and application protocol requires termination
  - Using endpoint identification can be confused by a crash
Identifying a Client

- The point of state is efficient
  - Reducing the amount of data transferred
  - Intuitive design
- Difficulty
  - Maintain the correctness when delay, duplication allowed
  - State information may be incorrect when computer restarts
Identifying a Client

In general, in a real internet, where machines crash and reboot, and messages can be lost, delayed, duplicated, or delivered out of order, stateful designs lead to complex application protocols that are difficult to design, understand, and implement correctly.
Statelessness is a Protocol

Issues

- Statelessness or not centers on the application protocol more than implementation.
- The issue of statelessness focuses on whether the application protocol assumes the responsibility for reliable delivery.
- *Idempotent* – design issue for statelessness operation always has the same result.
Servers as Clients

Programs do not fit exactly into the definition of client or server

Avoid circular dependencies among servers