Client-Server Applications

Prof. Sanjeev Setia
Distributed Software Systems
CS 707

Client Server Systems

[Diagram showing user, client, request, response, server, process, system, hardware]
Client/Server Application

Overview

- Common communication patterns in distributed applications
  - Client-Server
  - Group (Multicast)
  - Function-shipping/Applets

- Client: process that requests service
- Server: process that provides service
- Client usually blocks until server responds
Overview cont’d

- Client usually invoked by end users when they require service
- Server usually waits for incoming requests
- Server can have many clients making concurrent requests
- Server usually a program with special privileges

Client and Server Functions

- Clients
  - interacts with users through a user interface
  - performs application functions
  - interacts with client middleware using middleware API
  - receives response and displays it if needed

- Servers
  - implement services
  - invoked by server middleware
  - provide error-recovery and failure-handling services
Middleware

**Definitions**

- Middleware is a set of common business-unaware services that enable applications and end-users to interact with each other across a network
- distributed system services that have standard programming interfaces and protocols ... services “sit in the middle” above OS and network software and below industry-specific applications
- the “/” in client/server applications
- software nobody wants to pay for
Examples

- ftp, email
- Web browsers
- Database drivers and gateways
- OSF’s DCE (Distributed Computing Environment)
- OMG’s CORBA (Common Object Request Broker Architecture)
- Microsoft .NET

Functional View of Middleware

- Information exchange services
- Application-specific services
  - specialized services, e.g. transactional services and replication services for distributed databases, groupware services for collaborative applications, specialized services for multimedia applications
  - business-unaware
- Management and support services
  - needed for locating distributed resources and administering resources across the network
**Commercial Middleware**

- Middleware components that provide only one service
  - HTTP for retrieving remote documents, SUNRPC for RPC, etc.
- Middleware environments that combine many services
  - Integrates RPC, security, directory, time and file services
  - DCE, CORBA, Microsoft DCOM, .NET, Java
- Compound middleware environments that combine many middleware environments into a single framework, e.g. transaction management + RPC/RMI

**Application Software Architectures**

- Many applications can be considered to be made up of three software components or logical tiers
  - user interface
  - processing layer
  - data layer
- Client/server architectures
  - single-physical tiered, two-physical tiered
  - multi-tiered
Processing Level

The general organization of an Internet search engine into three different layers:

1. User-interface level
2. Processing level
3. Data level

"Gartner Group" Configurations

<table>
<thead>
<tr>
<th></th>
<th>Presentation</th>
<th>Logic</th>
<th>Data</th>
<th>Presentation</th>
<th>Logic</th>
<th>Data</th>
<th>Presentation</th>
<th>Logic</th>
<th>Data</th>
<th>Presentation</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>distributed</td>
<td>remote</td>
<td>data</td>
<td>distributed</td>
<td>remote</td>
<td>data</td>
<td>distributed</td>
<td>remote</td>
<td>data</td>
<td>distributed</td>
<td></td>
</tr>
</tbody>
</table>

Distributed Software Systems 13

Distributed Software Systems 14
**Distributed Data**

Example: Distributed Database

```
<table>
<thead>
<tr>
<th>presentation</th>
<th>logic</th>
<th>data</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>data</td>
</tr>
</tbody>
</table>
```

**Remote Data**

Example: Network File Systems

```
<table>
<thead>
<tr>
<th>presentation</th>
<th>logic</th>
<th>data</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

**Distributed Programs**

Example: World Wide Web

<table>
<thead>
<tr>
<th>presentation</th>
<th>html, Java</th>
</tr>
</thead>
<tbody>
<tr>
<td>logic</td>
<td></td>
</tr>
<tr>
<td>logic</td>
<td>CGI, httpd server</td>
</tr>
<tr>
<td>data</td>
<td></td>
</tr>
</tbody>
</table>

**Distributed Presentation**

Example: X Windows

<table>
<thead>
<tr>
<th>presentation</th>
</tr>
</thead>
<tbody>
<tr>
<td>presentation</td>
</tr>
<tr>
<td>logic</td>
</tr>
<tr>
<td>data</td>
</tr>
</tbody>
</table>
Remote Presentation

Example: telnet

Three-tier architectures
Motivation for multi-tier architectures

- Frees clients from dependencies on the exact implementation of the database
- It allows “business logic” to be concentrated in one place
  - Software updates are restricted to middle layer
- Performance improvements possible by batching requests from many clients to the database
- Database and business logic tiers could be implemented by multiple servers for scalability

Modern Architectures

An example of horizontal distribution of a Web service.