Processes – II

- Threads
- Virtualization
- Further Discussions on Clients and Servers
  - Design issues in clients
  - Design issues in servers
- Code Migration
- Agents
Mainly two ways for a client to connect to a server
  a) Application specific solution
  b) Application neutral– thin-client approach
a) Application Specific Solution

- For each remote service, the client m/c will have a separate counterpart that can connect to a networked application with its own protocol.
- Example: a PDA synchronizes agenda with a remote m/c.
b) Application neutral - thin-client approach

- Direct access to remote services by only offering a convenient user interface (client m/c = terminal with no local storage)
- A general solution to allow access to remote applications
## Client Design Issues - Transparency in a DS

<table>
<thead>
<tr>
<th>Transparency</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Access</td>
<td>Hide differences in data representation and how a resource is accessed</td>
</tr>
<tr>
<td>Location</td>
<td>Hide where a resource is located</td>
</tr>
<tr>
<td>Migration</td>
<td>Hide that a resource may move to another location</td>
</tr>
<tr>
<td>Relocation</td>
<td>Hide that a resource may be moved to another location while in use</td>
</tr>
<tr>
<td>Replication</td>
<td>Hide that a resource may be shared by several competitive users</td>
</tr>
<tr>
<td>Concurrency</td>
<td>Hide that a resource may be shared by several competitive users</td>
</tr>
<tr>
<td>Failure</td>
<td>Hide the failure and recovery of a resource</td>
</tr>
</tbody>
</table>
Client Side for Distribution Transparency

- A client is more than a user interface
- Access transparency
  - Client generates a stub based on an interface provided by the server (hides the m/c architecture and communication)
- Location, migration, relocation transparency
  - Naming – will discuss in detail next week
  - Client side cooperation
- Concurrency transparency
  - Usually handled by (intermediate) server

- **Failure transparency**
  - Client middleware

- **Replication transparency**
  - Client proxy
Servers Design Issues – Types of Servers

- **Iterative server**
  - The server itself handles the request, and if necessary, returns a response to the requesting client

- **Concurrent server**
  - Does not handle the request itself. Instead, it generates a thread or process to handle the request and to return the required response
Types of Servers – cont’d.

- **Server interruption**
  - A client exits the application at any time and restarts it again as if nothing has happened.
  - Sending out of band data (mandatory urgent data) using a different connection between a client and a server.
  - Using the same connection to send out of band data and normal data.
Types of Servers – contd.

- **Stateless server**
  - Does not keep information of the client states, changes its own state without informing the clients – web server
  - Some web servers keep some information about the client – using cookies

- **Stateful server**
  - Maintains information about the states of the clients – file server
Server endpoints – clients contact point at the server

a) Client-to-server binding using a daemon (a client first contacts a Distributed Comp. Env. (DCE) daemon for endpoint information)
b) Client-to-server binding using a superserver (client requested server creates another server for the service)
Server Clusters

**A collection of machines connected through a network, where each m/c runs one or more server**

The general organization of a three-tiered server cluster
Distributed Servers

- Dynamically changing set of machines, varying access points, appearing to the outside world as a single, powerful machine
- Designing robust, high-performance, stable server
  - Provide several access points so that the cluster can still be used when an access point fails
  - Offer a stable address to a distributed server
  - Group simpler machines transparently into a cluster instead of relying on the availability of a single machine
  - Use high performance machines with very high mean time to failure
Summary

Clients and Servers

- Design issues in clients
- Design issues in servers