Client-Server Architecture (2-Tier)

Client: A requester of services  
Server: A provider of services  

X Windows is a typical example.

Serves as an alternative to file sharing  
- Ship the database and/or application to the user to process locally.

Interface (client) is typically on user’s desktop PC.

Database or application (server) is typically on a more powerful server.

Limitations (DB view): Connections are “static” with a fair amount of overhead.
Three-Tier Architecture

A middle tier (sometimes called middleware) is added between the interface and the DBMS (or other application).

Middle tier can include things like Job queueing and scheduling.

Middle tier can keep a limited number of connections open to the DBMS, while receiving requests from many clients.
  • Thus can handle many users.

Client can disengage once the request has been delivered.
WWW Access to Legacy Codes

Background:

- Many existing large-scale simulations and other legacy codes that cannot be rewritten for the WWW.

Problem:

- Users of this codebase would like client/server style access to legacy codes from the WWW.

Caveats:

- Legacy codes must not have an existing GUI.
- Legacy codes may be tied to a server.
Solution to Legacy Code Access Problem

A three-tier architecture can solve the problem

- **Interface tier:** Java applet, etc., perhaps embedded in a browser page.
- **Server tier:** The application running on some server, invocable via command line and/or input files.
- **Middleware tier:** “The server” which
  1. accepts requests from the interface, including name of application and parameters for application.
  2. invokes the application with appropriate parameters/inputs. Often involves perl scripts, “wrappers” for legacy codes, and other “glue” for manipulating data.
  3. takes output (files) from application and returns appropriate information to the interface tier.

The information returned could be in the form of HTML pages generated by the server and sent back to the interface user’s browser.
Server Issues

- How to server and client exchange information?
- Who is the user on the server end?
- What may he/she do?
- Whose permissions are used to do it?
- Where do the files go?
Possible Solutions

Authentication:
- Use server’s standard account mechanism
- Server does its own authentication scheme
- No authentication

What can be done, and who actually does it?
- Anything the user can do (with user’s permissions)
- Use restricted shell to a predefined list of commands (with restricted account’s permissions)
- Use a configuration file that lists commands (with server’s permissions)

Where do the files go?
- To the user’s account
- To the server’s account (perhaps to sub-directories named for users)
Example Implementations

Original WBCsim:

- Ad hoc server
- Predefined list of commands
- Authentication is to know one of a list of names
- Files go to subdirectory with matching user name

New WBCsim:

- Telnet daemon
- Guest account running restricted shell
- Commands are those accessible locally to guest account
- Files go to guest account

Symphony:

- Symphony server
- Server machine’s authentication
- Can do whatever account permits

Watershed Decision Support System:

- ESRI server “MapObjects”