CPSC 310
Software Architecture: Client/Server Architectural Style

Dr. Gail Murphy

By the end of this class, you should be able to:

- Define what is meant by the term “software architectural style”
- Describe some characteristics of a client/server architecture

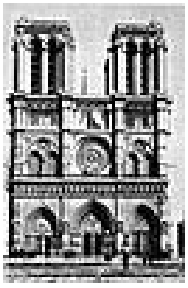
Software Architecture: What is It?

- Your development team has nailed down a version of the requirements for a system
- You have built an analysis model
- Now you must think about how you are really going to implement the system
- Need to choose an overall structure for the system; i.e, evaluate and choose an architecture for the system

Software Architeture

- A software architecture for a system describes
  - the subsystems and (large-scale) components that comprise the system
  - the organization of those subsystems and components
  - the global control structures
  - the protocols for communication
  - the management of data

Architectural Style (Buildings)



Early Gothic Architecture
Notre Dame, Paris

18th Century Georgian Architecture
"Old Corner Bookstore"
Boston MA

4

October 3, 2001 © G. Murphy

Buildings can be classified according to style. Asking for "Victorian features" in a house can help an architect produce a design that meets your requirements. Style in software architecture plays a similar role, although instead of appearance, the qualities of interest are reusability, adaptability, etc.

If a building meets a Gothic architecture, it means it has a few key features and rules for combining these features so that architectural integrity is preserved.

Architectural Style (Software)

- Handbook of basic structures and an analysis of their tradeoffs that can be used for structuring a system
- Selection of a style helps dictate qualities the system will have
  - e.g., modifiable? secure? scalable? fast? configurable? reliable? etc.
- About ½ dozen styles, including call/return, pipe and filter, repository, client/server, etc.

Imagine…

- Your software development company has been contracted to build a course delivery system (think WebCT)
- It has to support:
  - Display of lecture notes
  - Private/secure grade access
  - Multiple users
  - Bulletin board
  - etc.

One possibility is to have everyone log into a server and to provide some client that accesses a central repository.

Better is to try to distribute the processing: take advantage of the user's desktop.

Client/Server Architecture

- A "server" subsystem provides services to multiple instances of "client" subsystem
- Client and server are connected by a network
- Control is typically a client requests services from the server
- Server provides data access and maintains data integrity
- To handle load, can have more than one server

Consider Two Use Cases

1. Displaying the main course page
   - User requests the main course page
   - Main course page is displayed
2. Reading a message on the bulletin board
   - User requests the bulletin board
   - The headers of unread messages for that user are displayed
   - User requests to view a particular message
   - Message is displayed

What functionality belongs in client and what functionality belongs in server. Pretty clear if we have chosen a web client/server system. What goes where for the first use case? May be pretty simple to implement this use case: if it's a static web page, standard infrastructure will do. Run an off-the-shelf web server. Put the page on the server. Run a standard web browser on the client. Ask for the URL: voila its done!

What about the second use case? Now we actually need specialized functionality in the browser. How do we do this?

We'll briefly go over the servlet approach. Assume you have gathered the information about what user is asking for the information.

Servlets

- Interfaces a server application with the web
- Allows you to write your server application as a Java program
- Associate a Java program with a URL
- The program runs in a servlet container within the web server
  - Lightweight: Separate thread for each request

Hello World Example

```
import java.io.*;
import javax.servlet.http.*;

public class SimpleHello extends HttpServlet {

  public void doGet(HttpServletRequest req,
                    HttpServletResponse res)
  throws IOException {
    response.setContentType("text/html");
    PrintWriter out = response.getWriter();
    out.println(
      "<html>" +
      "<head><title>Hello</title></head>" +
      "<p>Hello, world!</p>" +
      "</body>" +
      "</html>"
    );
   }
}
```


To support our second use case, the servlet would have to be able to now what course web was being accessed and what user was requesting unseen bulletin board messages. These can be passed as parameters.

http://<host>/<servlet>?name=somebody

Can then have a bulletin board object that knows about all messages. It can also know about a set of users, and each user can remember which messages it has read. Ask the user for the unread messages (ask bulletin board for ids of known messages). Ask bulletin board for headers of unread messages. Format a page. Send it back.

A request can then come into the same or another servlet to ask for a particular message. A page can be formulated and sent back.

When we program a client/server application with the web and servlets, we have to think about what servlets we want to support and what the structure of each servlet should be. Does a servlet have an architecture? Sort of. Its fuzzy where the line is between architecture and design. It gets further confused because one of the kinds of architectural style is the object-oriented architectural style.

Do Servlets Have Architecture?

- Maybe
- It's a fuzzy line where architecture ends and more detailed software design begins
- Even more confusing because one architectural style is the object-oriented style

Building the Course Web System

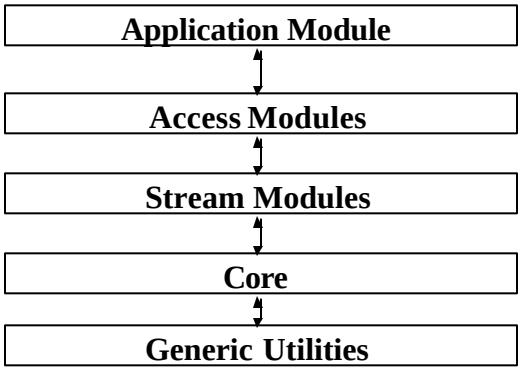
- If we choose a client/server architecture, can take advantage of lots of standard infrastructure in the web
- Client is just the browser
- Server is a set of servlets sharing a database
- Get a scalable solution where it is (relatively) easy to upgrade the application

(Original) Architecture of the Web?

- library (libWWW) to hide all hardware, OS, and protocol dependencies

LIBWWW

Layered Architecture



45 minutes in at end of slide

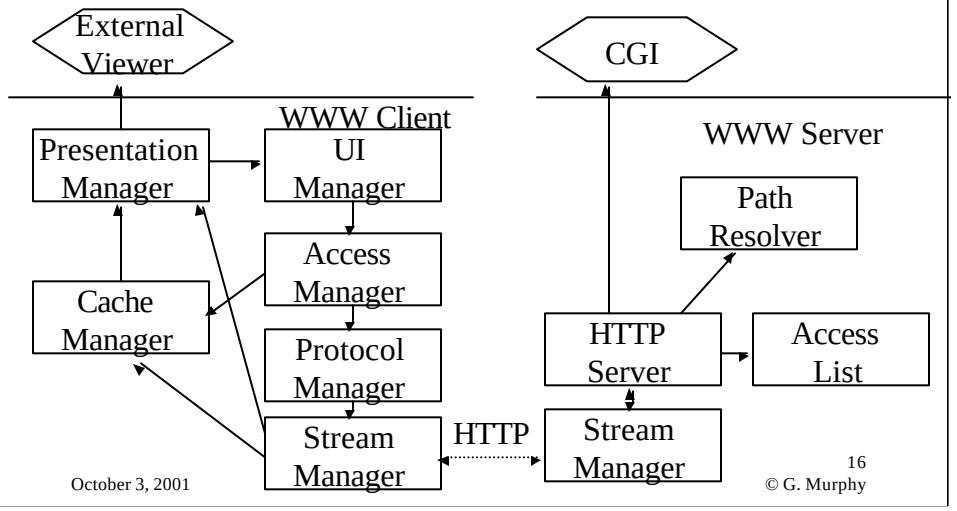
Ask students what the diagram tells them (without explaining it).

Meaning of Layers

- These notes describe a bit about the layers in the libwww diagram:
  - Generic Utilities has building blocks for system such as network management, container classes, and string utilities. This layer allows other layers to be platform independent, easing the task of porting the library.
  - The Core has all the basic functionality for a WWW applications such as network access, logging, etc. This layer just provides a standard interface with actual functionality provided by "plug-ins" and "call-outs". Plug-ins are modules registered by an application at run-time and do things like sending data. Call-out functions are arbitrary application-specific functions that can be called before or after requests to protocol modules.
  - The stream layer abstracts a stream of data.
  - The access layer provides network-protocol-aware modules like HTTP, NNTP (news), WAIS (wide-area information system), FTP, TELNET, etc.
  - The top layer consists of WWW application modules such as caching, registering proxy servers, history maintenance, etc.

A Client/Server Architecture using LIBWWW

[BCK98, p.156]



Not all parts of the client-server are built from libWWW. (e.g., UI manager is not part of libwww). Managers don't relate directly to layers in the libwww.

UI gets user's request as a URL and passes it to the access manager. The access manager checks the cache and if it is cached, it is retrieved and given to the Presentation Manager. If its not cached, the protocol manager determines the type of request and invokes the appropriate protocol suite to service the request. When the stream manager gets a response, it hands it to the presentation manage rfor display.

Include discussion of how this is changing with servlets, etc.

Ask how each quality on the next slide is met.

Summary

- Architecture determines the fundamental structure of a system
- Earliest design decisions
- Architectural styles guide choice of architecture: encode common patterns for achieving certain quality goals
- Client/Server architectural style can help provide a scalable, responsive system
- Many good building blocks to support development of client/server systems