Software Design and Architectures
CS446 / ECE452 / CS646 / SE-2

Lecture 7 : Clients, Servers, Brokers, and Components

When a software system is designed to operate on a network of computers, the architecture shows it. There are patterns of interaction characteristic of network-distributed processes; and there is a characteristic technique for handling large open networks of distributed service. Such designs are often closely related to the design of the business processes which the software supports.

What's happening here is looking at the effect of certain physical architectural decisions on the logical architecture. Logically, certain processes exchange data and control with each other. Physically, the processes are resident on different nodes. This has two key effects:

- data flows from client to server much slower than between internal units (memory)
- the configuration of available services and user clients is less stable (than on a single machine)

Example: File Transfer Client and Server
Purpose is to provide secured access to file data poossibly available on a different machine. Alternatively: to overlay the file system with a more open and independent access facility (with a network).

[diagram]

The *protocol* is the conversational terms between a server and its users ("clients"). Data is transferred on a separate channel from the actual conversation between the server and client. The FTP protocol includes USER, PASS, CWD, MODE, RETR, STOR, ABOR, LIST,

Protocols are routinely defined over TCP and routinely use ASCII at present.

Example: Mail Client and Server
Purpose is to be a central point of archive and distribution for e-mail data.

[diagram]

There are three protocols on this diagram.

The SMTP (simple mail transfer) represents a conversation between a source of mail and a receiver. Except when the client is sending mail to the very machine on which the server is running, there are two SMTP servers involved, on of which is a client of the other (so its peer to peer). The main verbs are HELO, MAIL, RCPT, DATA, and QUIT. They mean "establish connection", "want to send mail from this address", "want to send mail to this address", "want to transmit the message", and "done". These are all sent by the client. The server sends numerically prefixed messages in response (e.g. "500 line too long" or "250 ok").

The IMAP (internet message access) protocol allows messages to be stored and managed on the server side. Messages are not accessible when the server is down: but the same email repository can be accessed from multiple machines. It's goals are

- Be fully compatible with Internet messaging standards, e.g. MIME.
- Allow message access and management from more than one computer.
- Allow access without reliance on less efficient file access protocols.
- Provide support for "online", "offline", and "disconnected" access modes *
- Support for concurrent access to shared mailboxes
- Client software needs no knowledge about the server's file store format.

Typical commands include SELECT a mailbox, CLOSE a mailbox, EXAMINE a mailbox (read only), request STATUS of a mailbox. Typical responses include status responses such as OK, ALERT the user, a CAPABILITY list; and server data.

The POP (post office) protocol allows messages to be stored and managed on the client side. Messages can be organized, viewed, archived, etc., independently of any server.

Example: corporate mainframe/server configuration

[diagram] Pressman p 785

Here historically the situation is that a central IS database managed on a single mainframe has been under pressure to export itself to a wider and more flexible environment. The response to this pressure is to put the mainframe on the network, and provide useful but controlled forms of access to the data by means of *servers*. It is then possible to migrate some of the responsibilities to clients of unspecified functionality, easing the management of the central repository.

Example: Google Server Farm

[diagram] ref. Neville-Manning

Example: Sherlock Translation Service

[screen-shot]

Assignment of Responsibilities
Recall the responsibilities which arise in the design of an information system (in particular). (This account could be changed for other situations and responsibilities arising in a different setting). We had:

- core repository storage
- data consistency and correctness (including distribution if necessary)
- access synchronization
- access control and security
- message handling, marshalling and unmarshalling
- work flow marshalling
- transaction control (business logic)

- user-level presentation and data entry

Typical Roles in a Client-Server System
At the physical boundary between client and server we must define the interface in terms of messages and a relatively slow connexion. The layer at which commuinication is taking place is an important consideration. The lower the layer, the less complexity in managing the network infrastructure, and the more flexibility, but the less control. Typical layers include

- file servers, exchanging only file data, to be handled by logic on the client or server side
- database servers, exchanging structured and managed application data
- transaction servers, exchanging and managing workflow at the application level
- groupware servers, centralizing the management of complex and persistent work artefacts (including multiple media)

Distribution of Roles
The physical view is essential and influential in this setting: which nodes contain serves and which contain clients. But we still haven't discussed where the functionality is to lie. Typical assignments include

- thin client, fat server: the client is doing little but displaying data and relaying data entries
- fat client, database server: the client knows about workflow and contains application layer software
- remote (autonomous) data managemetn: the server is responsible for data collection and formatting from external sources
- multi-server (web-services) distributed data management: the client is responsible for choosing and accessing multiple servers

There are various pressures relating the thin/fat client issue, and the technology is constantly changing.

- thin clients are cheap and fast and simple; inevitably the

Broker
The problem of distributed multiple servers is addressed by the use of *brokers*. If the broker is an autonomous local server, it's the DNS model. if the broker is part of the client, it's a web services model. When brokers communicate with each other to find a service and/or translate requests between clients and distributed servers, it's a distributed object model.

Peer-to-Peer
This term is used to refer to client/server architectures in which the roles of client and server are blurred. Obviously a client of one server can be itself a server of some third client. In fact this relationship can be circular. It has the advantage of openness (potentiallyany peer can join the network and cooperate with others) but considerable added complexity (all participants in the network must be as complex as servers, able to serve multiple clients simultaneously, maintain connexions, present consistent data, and so on. In strict C/S, clients can be very simple.)

[diagram]

Service Brokers

References:
Lotsa stuff on the web about mail protocols, e.g. http://cr.yp.to/smtp/client.html
The official description of FTP is RFC 959.
Pressman Chap. 28 (pp 784ff)
Buschmann sec. 2.3
Buschmann Chap 2 pp 99-124
M Shaw and P Clements. A field guide to boxology: preliminary classification of architectural styles for software systems. Proc. COMPSAC97, 21st Int'l Computer Software and Applications Conference, August 1997, pp. 6-13. Try http://spoke.compose.cs.cmu.edu/shaweb/p/pubs.htm