cis3.2 — electronic commerce — 12 sep 2005 — lecture # 5

today

topics:

- clients and servers
- servers
- architecture
- middleware
- protocols
- services

reading:

- today: Ince chapter 3 (sections 1 through 5)
- for next time: Ince chapter 3 (the rest of the chapter)

clients and servers

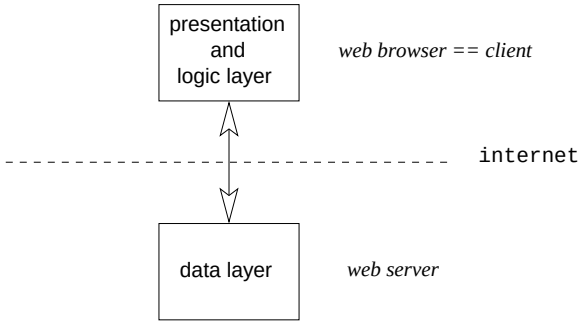
- server = computer which carries out a *service*
- client = computer which requires the service

servers

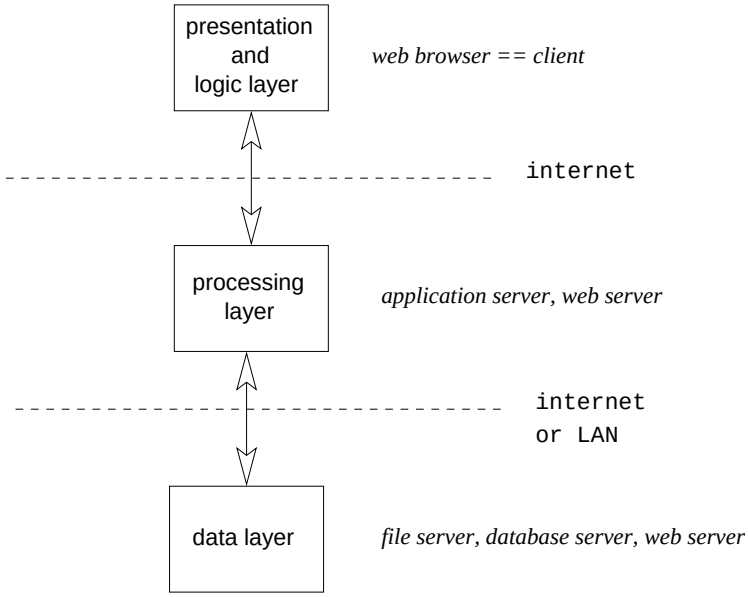
- file servers (provides files for clients)
- database servers (specialized file server: provides databases – structured files – for clients)
  - what is a *database*?
  - key concepts: SQL (structured query language), hierarchy, *records*, *fields*
- web servers (specialized file server: provides files that make up the components of a web site, e.g., html documents, video clips, etc)
- groupware
  - manages scheduling for individuals and groups of co-workers/collaborators
  - provides reports (e.g., billing) for collaborators
  - supports mailing lists for collaborators
  - e.g. Lotus Notes
- mail servers (provides mail sending, receiving, storage)
- object servers (provides access to "distributed objects")
- print servers (manages a print *queue*)
  - adds requests to the queue
  - schedules requests
  - instructs printer regarding requests
  - provides status on requests to clients
- application servers (provides access to particular applications, e.g., game servers)

architectures

- *two-tier architecture:*



- *three-tier architecture:*



- advantages:
  - isolates data storage technology
  - places more burden on server (instead of client) and distributes tasks amongst server(s)
  - follows object-oriented and modular paradigms

middleware

- whatever stands between between a client and the service that the client is requesting... (it can be a fine line)
- API = application programmer interface
- example: browser accesses web content through middleware
- two types:
  - *general* (e.g., transports raw data around the internet, synchronizes replicated files, administers distributed data sets)

  - *service* (e.g., queries a database, accesses distributed objects, accesses newsgroups)

- *message-oriented middleware (MOM)*
  - software that manages transactions that pass between client and server(s)
  - advantages:
    * means client doesn't have to be connected to server all the time (e.g., email, sales, inventory)
    * simple interaction model

protocols

- rules governing interactions
- *handshake* (operating system and data format, i.e., what will be passed and how)
- example application protocol:

```
Det shirts     ; ask for inventory details,

DetList        ; return inventory details
shirt blue 1999
shirt red 1234
```

- exists at lowest level of interaction between client and server — user will probably never see it!
- examples: POP3 (mail)
- secure sockets layer (SSL): security through cryptography
- protocol stack: hierarchy of embedded services (and corresponding protocols)