07

Distributed System
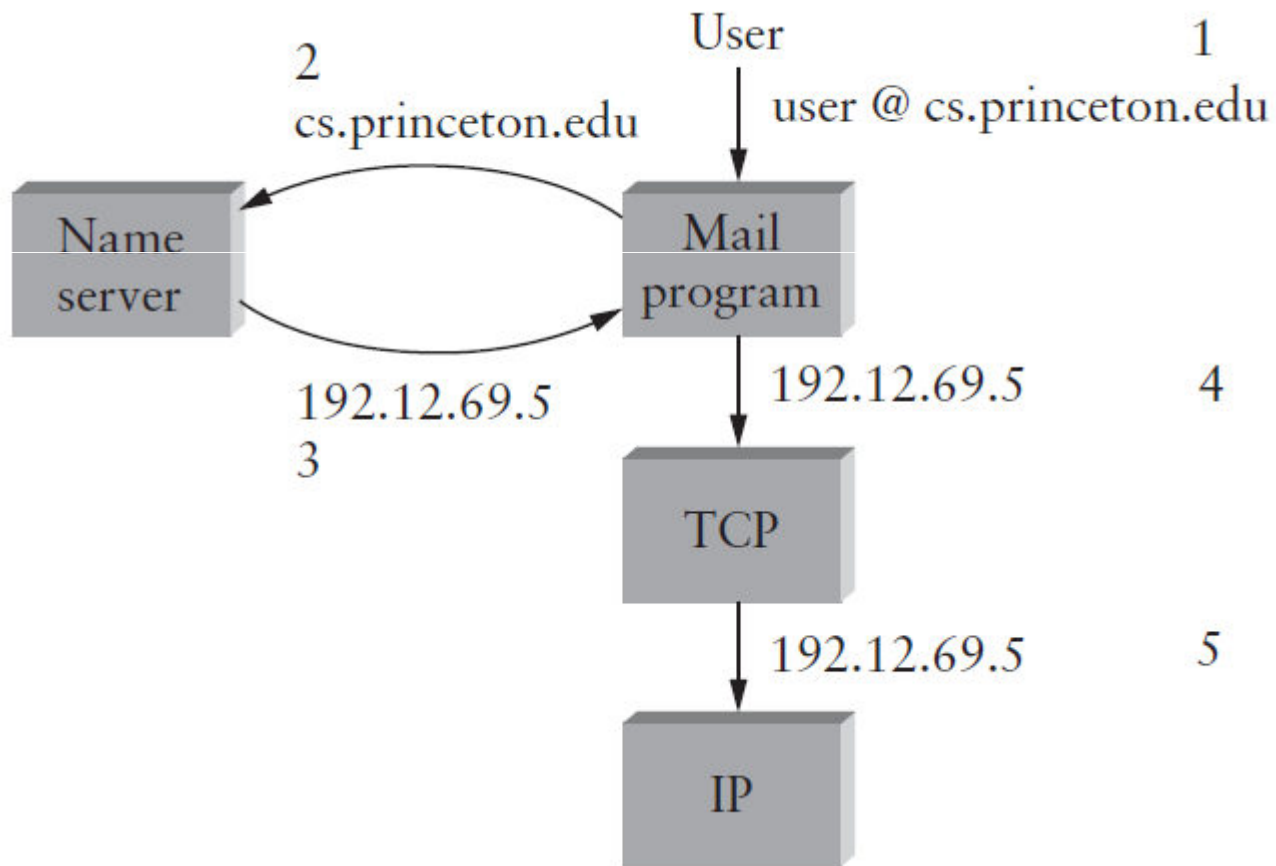
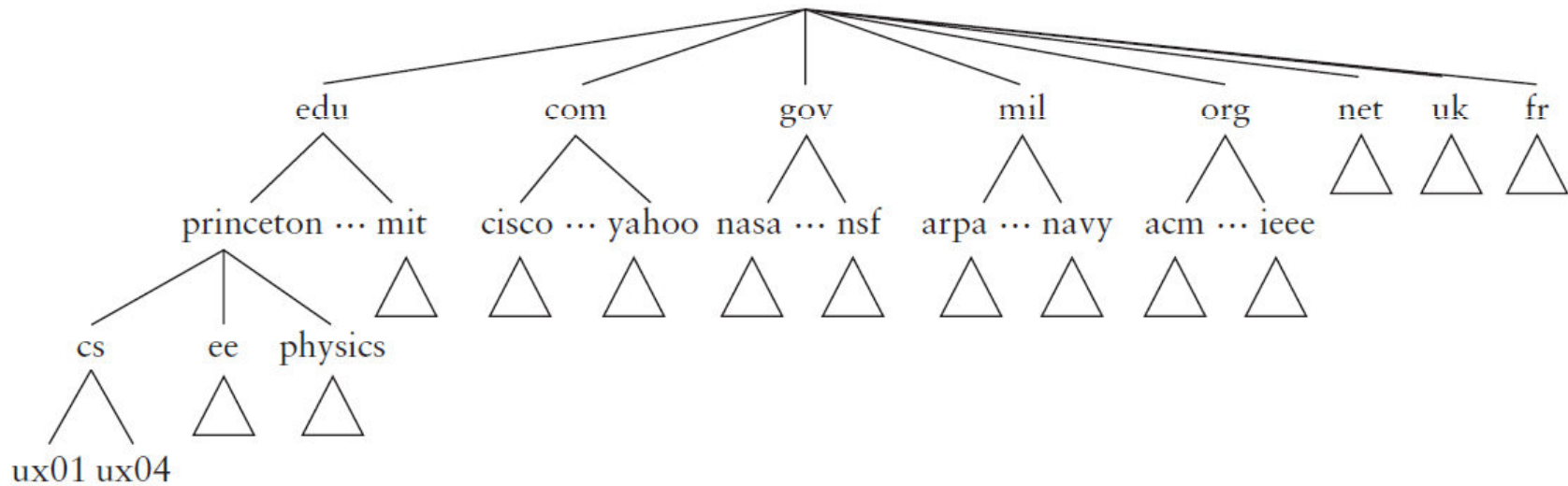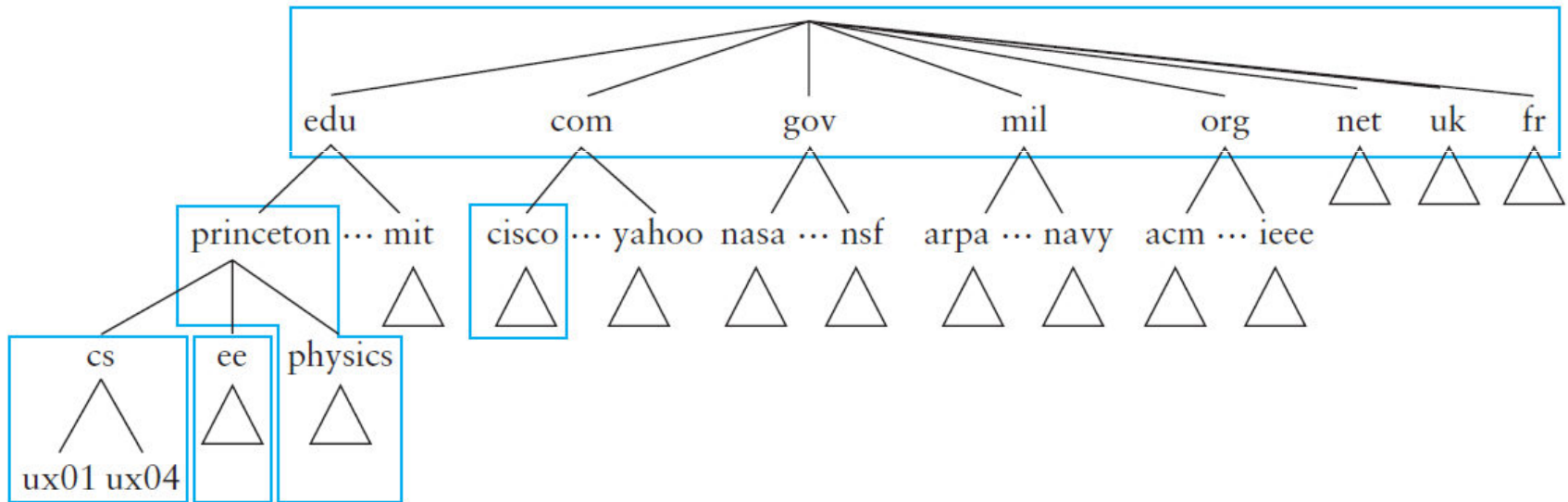# What happened behind these?
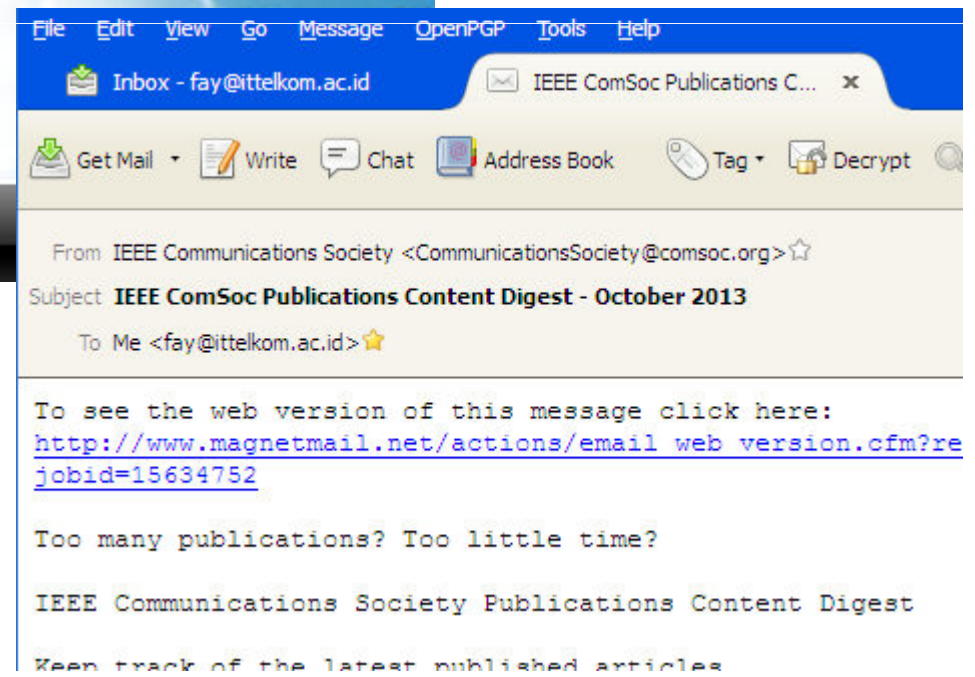
Application protocol

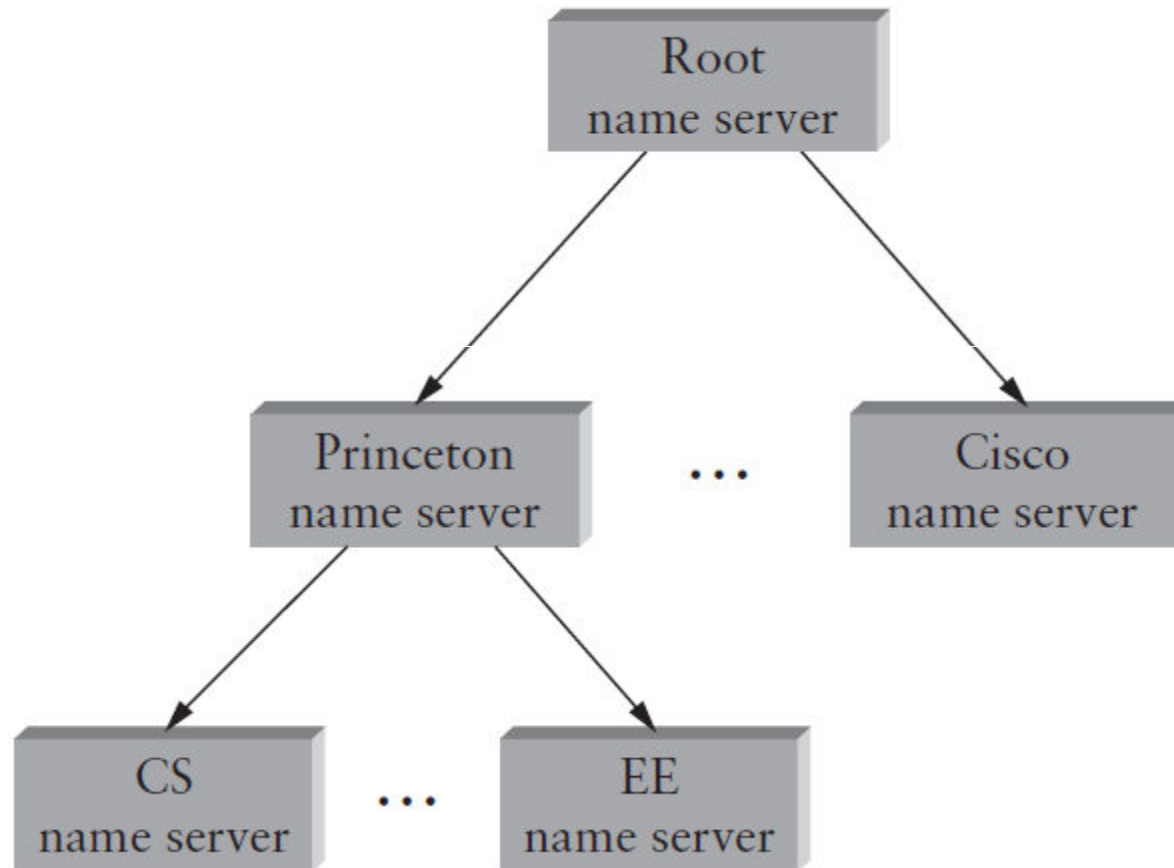- Domain Name Service – usage example

Domain name hierarchy

Domain name partitioned into zones

Domain name server hierarchy
each server serving specific zone

Domain name resolution
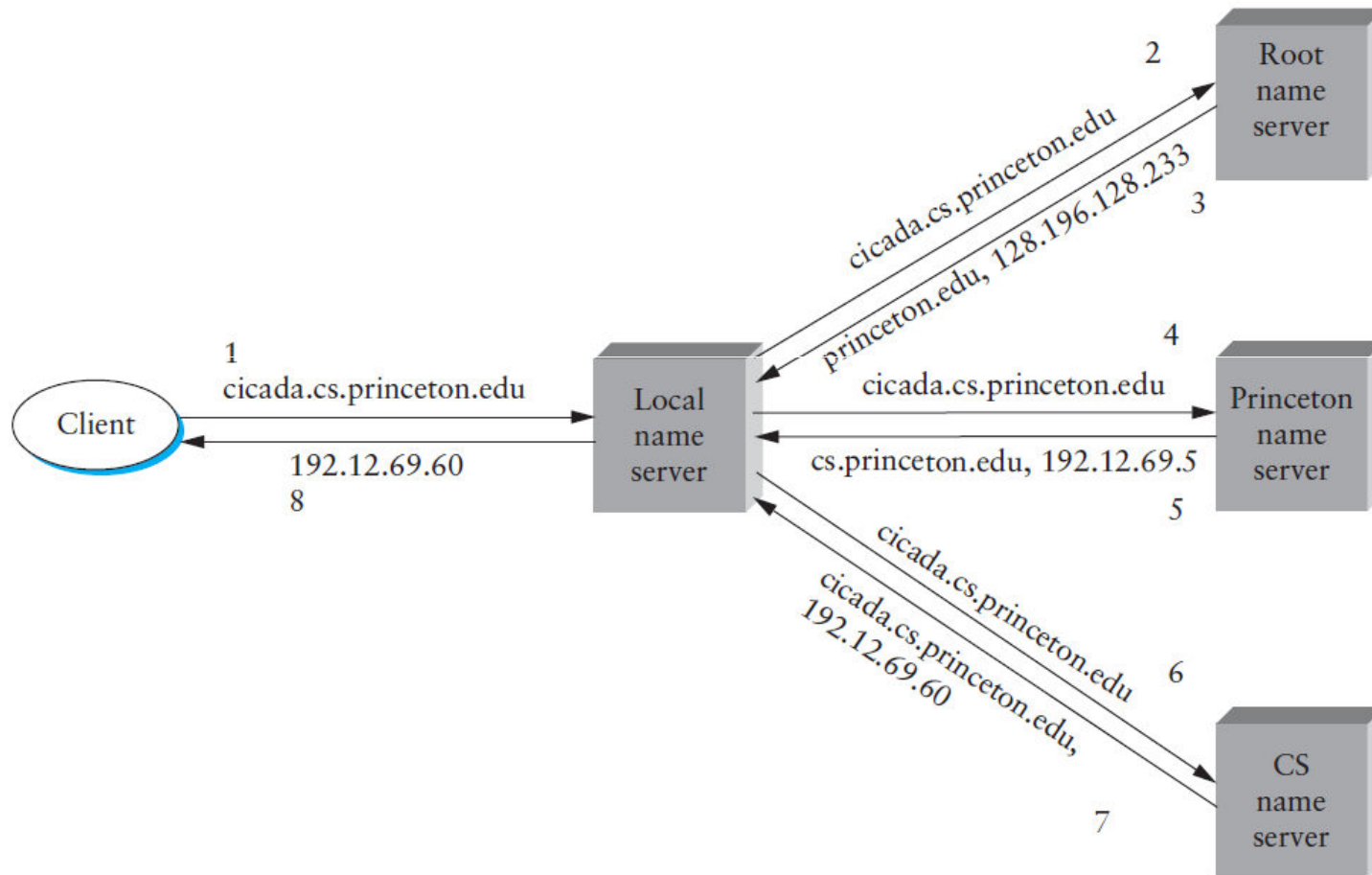
World Wide Web

- HTTP (Hypertext Transfer Protocol)
  - Protocol for transfering Internet Object
  - Utilize Request – Response Protocol
  - Message types (Method) : HEAD, OPTIONS, GET, PUT, POST, DELETE, TRACE, CONNECT
  - Message format (Request) : request line + headers [+ contents]
  - Message format (Response): status line + headers [+contents]

HTTP Session

```
%telnet localhost 80
Trying ::1...
Connected to localhost.
Escape character is '^]'.
GET / HTTP/1.0

HTTP/1.1 200 OK
Date: Mon, 16 May 2011 10:56:33 GMT
Server: Apache/2.2.17 (FreeBSD) mod_ssl/2.2.17 OpenSSL/0.9.8q mod_python/3.3.1 Pyt
hon/2.6.6 PHP/5.3.5 with Suhosin-Patch
Last-Modified: Sat, 20 Nov 2004 20:16:24 GMT
ETag: "6e55a5-2c-3e9564c23b600"
Accept-Ranges: bytes
Content-Length: 44
Connection: close
Content-Type: text/html

<html><body><h1>It works!</h1></body></html>Connection closed by foreign host.
%
```
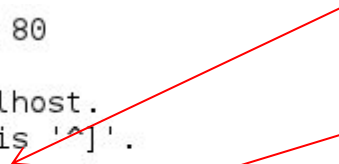
Request line

Status line

Electronic mail

- Sequences of mail gateways store and forward email messages
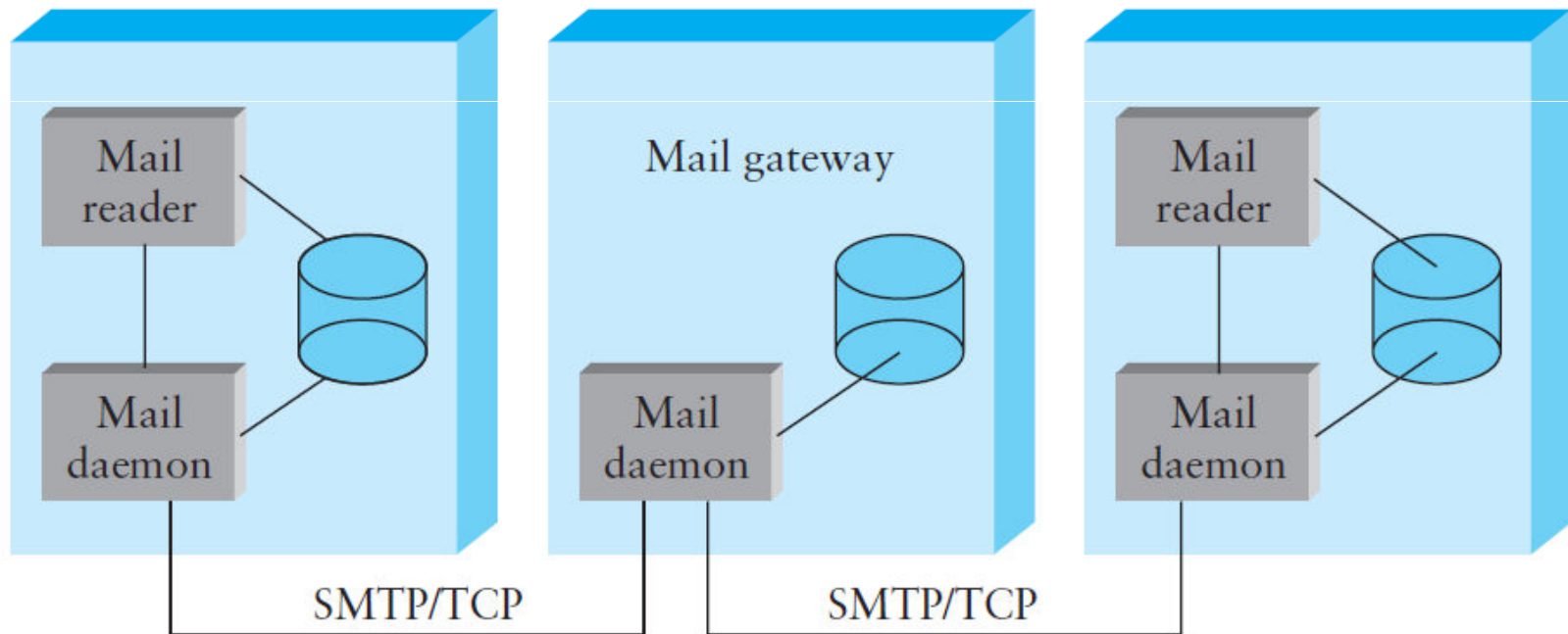
SMTP session

```
HELO cs.princeton.edu
250 Hello daemon@mail.cs.princeton.edu [128.12.169.24]

MAIL FROM:<Bob@cs.princeton.edu>
250 OK

RCPT TO:<Alice@cisco.com>
250 OK

RCPT TO:<Tom@cisco.com>
550 No such user here

DATA
354 Start mail input; end with <CRLF>.<CRLF>
Blah blah blah...
...etc. etc. etc.
```

Distributed system

- Why we developed distributed system?
  - Availability powerfull yet cheap microprocessor, continuing advance in communication system.
- What is distributed system
  - A Collection of independent computers that appears to its user as a single coherent system
  - Software :
    - A piece of software that ensure that:
      - A Collection of independent computers that appears to its user as a single coherent system

Distributed system

- Two aspect:
  - (1) Independent computers and
  - (2) Single system → middleware
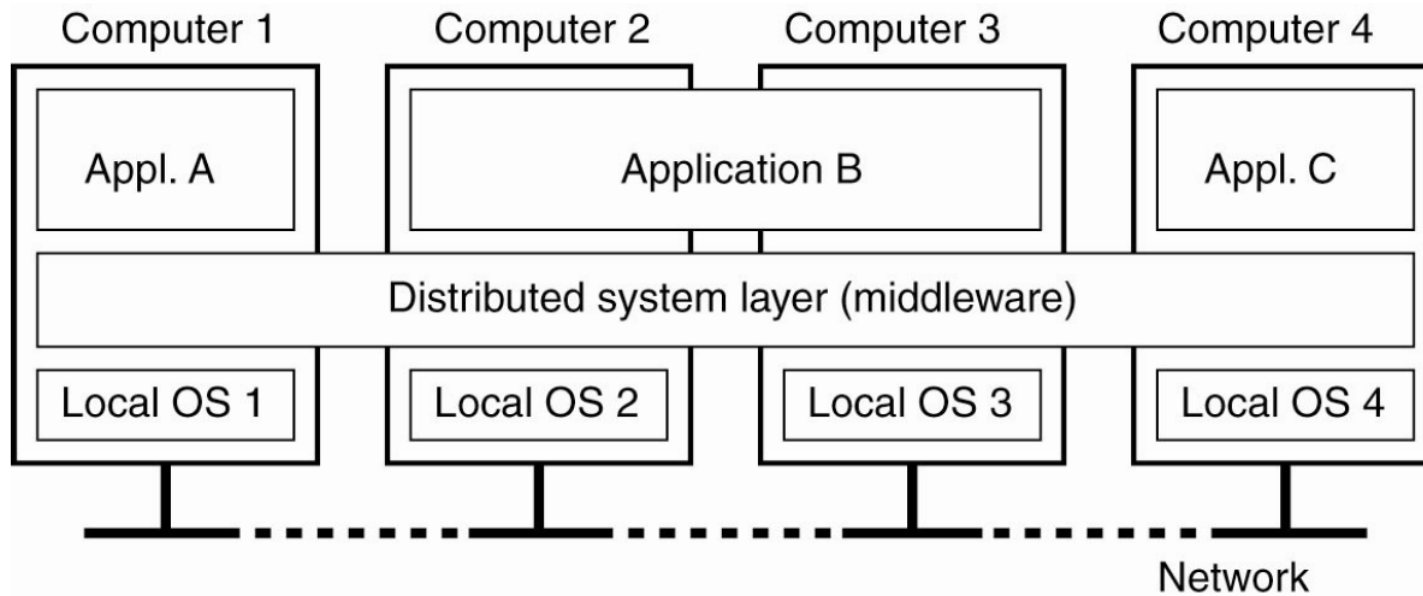
Middleware is any software that allows other software to interact.

- Some types of middleware:
  - **Message Oriented Middleware**. includes asynchronous store and forward application messaging capabilities
  - **Object Middleware**. Object Request Brokers
  - **RPC Middleware.** provides for calling procedures on remote systems, represents synchronous interactions between systems and is commonly used within an application.
  - **Database Middleware**. allows direct access to data structures and provides interaction directly with databases.
  - **Transaction Middleware**.
  - **Portals**. "front end" integration.

Distributed System

- Characteristics
  - Differences between various computers and how they interacts are mostly <u>hidden</u> to user.
  - Users and application can interact with distributed system in <u>consistent</u> and <u>uniform</u> way, regardless where and when interaction take place

Transparency in Distributed System

Transparency	Description
Access	Hide differences in data representation and how a resource is accessed
Location	Hide where a resource is located
Migration	Hide that a resource may move to another location
Relocation	Hide that a resource may be moved to another location while in use
Replication	Hide that a resource is replicated
Concurrency	Hide that a resource may be shared by several competitive users
Failure	Hide the failure and recovery of a resource

Advantages of distributed system over centralized system

- **Economics**, a collection of microprocessors offer a better price/performance than a mainframe
- **Speed**, a distributed system may have more computing power than a mainframe. Ex 10.000 CPU chip each runinng at 50MIPS. Not possible to build a CPU with 500.000 MIPS
- **Inherent distribution**, some application are inherently distributed.
- **Reliability**, if one machine crashes, the system as a whole can survive.
- **Incremental growth**, computing power can be added in small increments.
- **Another driving force**, PC availability, the need for people to collaborate and share information.
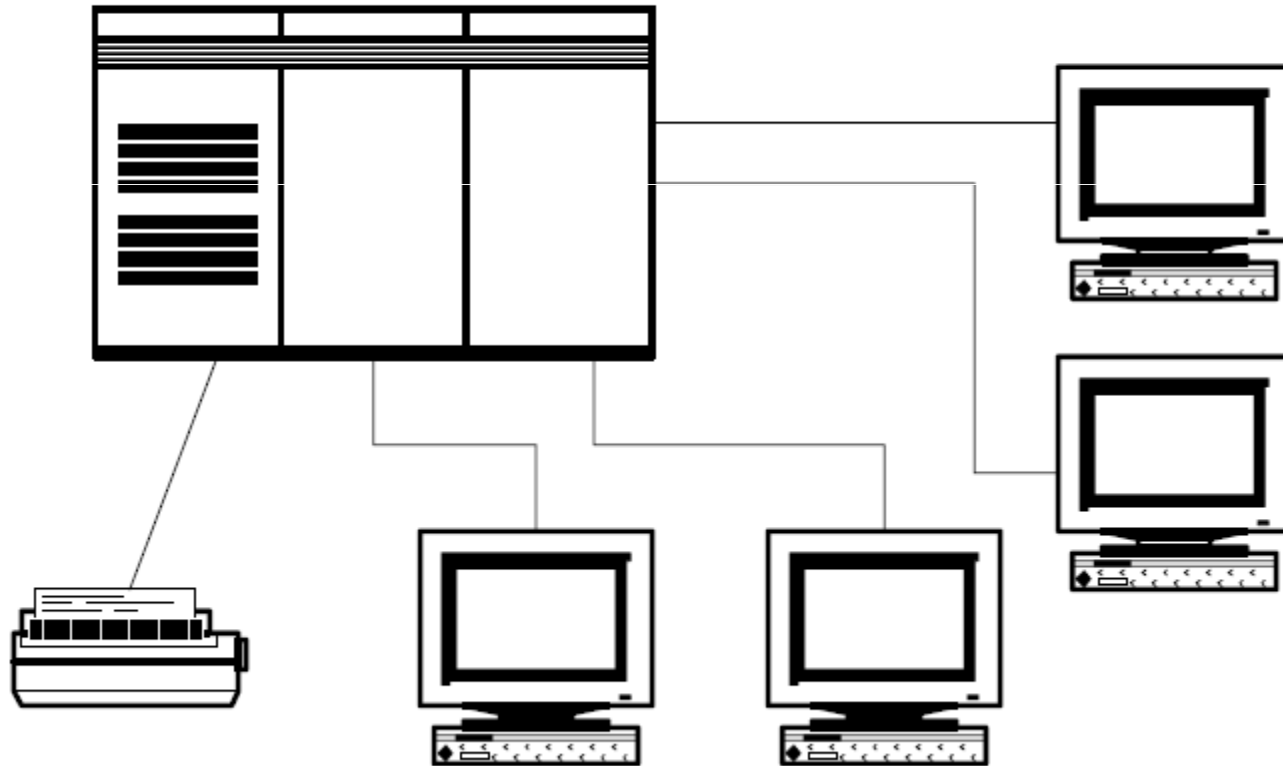
Advantages of distributed system over independent PCs

- Data sharing, allow many users to acces common data, e.g. database

- Resource sharing, expensive peripheral, e.g. color printer

- Communication, enhance human to human communication, e.g. email

- Flexibility, spread the workload over the available machines

Disadvantages of distributed system

- Difficult to develop software
- Network: saturation, lossy transmission
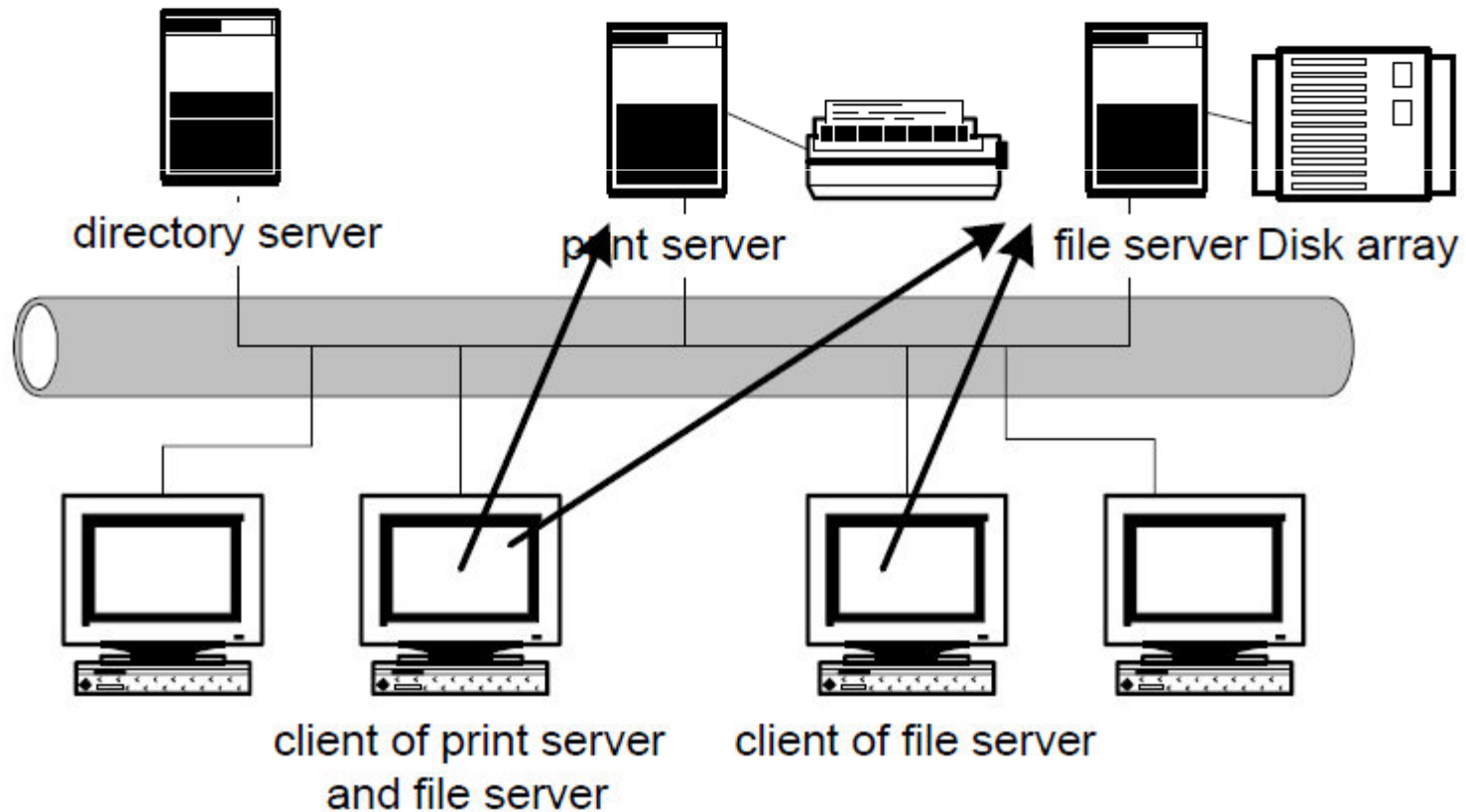- Security: easy access also applies to secret data
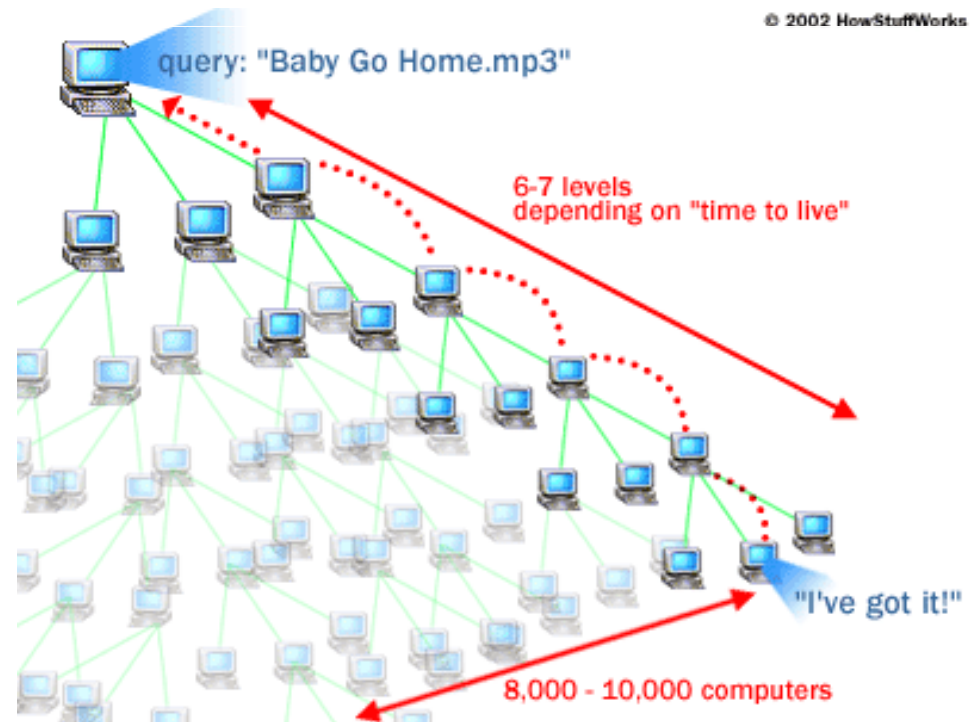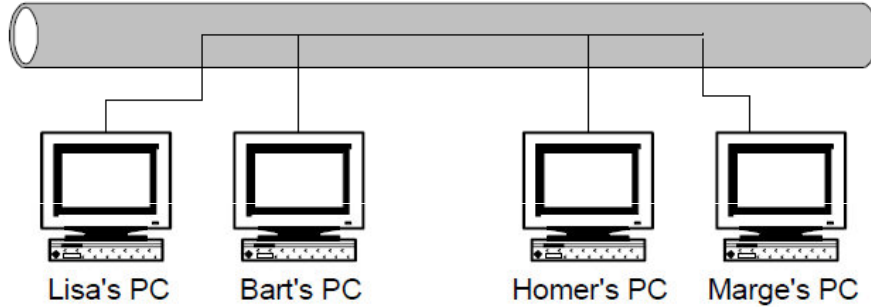
Service Model
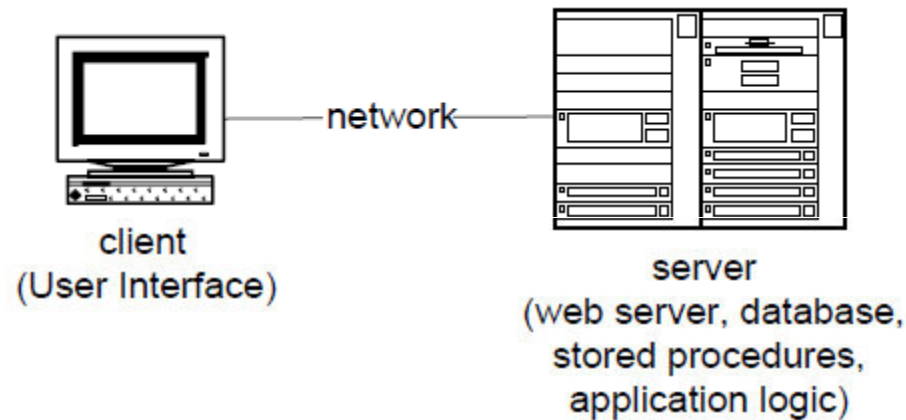
- Centralized Model

Service Model

- Client-server model

directory server   print server   file server Disk array

client of print server
and file server   client of file server

Service model

- Peer-to-peer model

Lisa's PC   Bart's PC   Homer's PC   Marge's PC

© 2002 HowStuffWorks

query: "Baby Go Home.mp3"

6-7 levels
depending on "time to live"

"I've got it!"

8,000 - 10,000 computers

Service model

- Multi-tier client-server model

- is a client–server architecture in which presentation, application processing, and data management functions are **logically separated**

Visual overview of a Three-tiered application

Presentation tier

The top-most level of the application is the user interface. The main function of the interface is to translate tasks and results to something the user can understand.

```
>GET SALES
 TOTAL
```
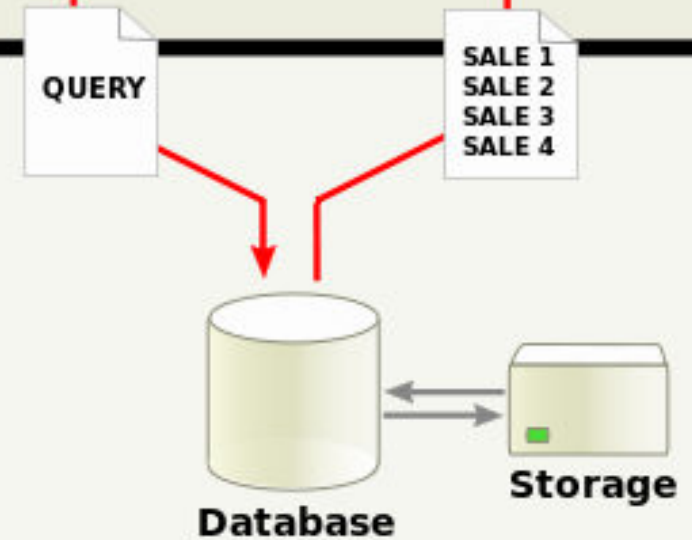
```
>GET SALES
 TOTAL
4 TOTAL SALES
```

Logic tier

This layer coordinates the application, processes commands, makes logical decisions and evaluations, and performs calculations. It also moves and processes data between the two surrounding layers.

GET LIST OF ALL SALES MADE LAST YEAR
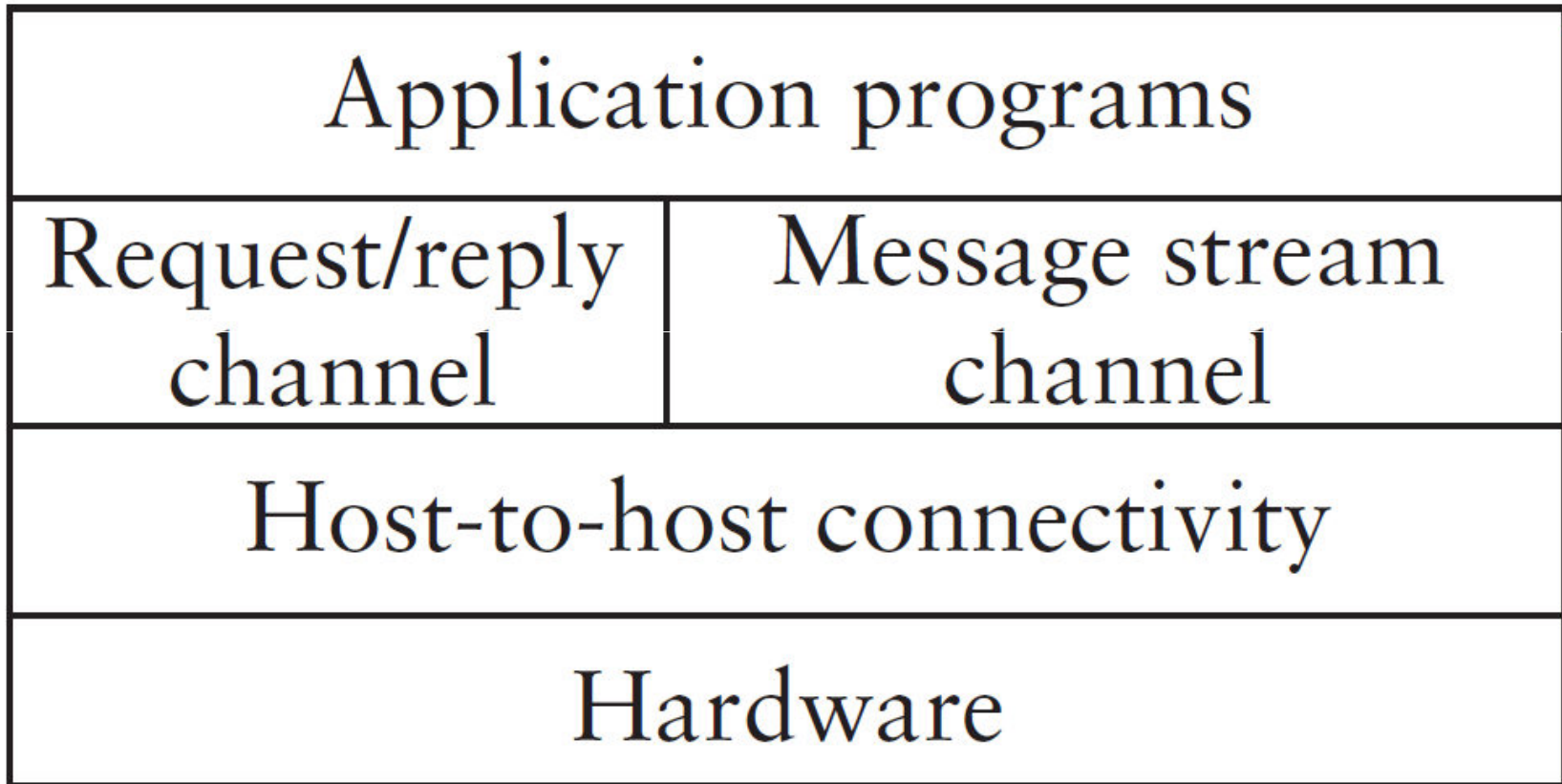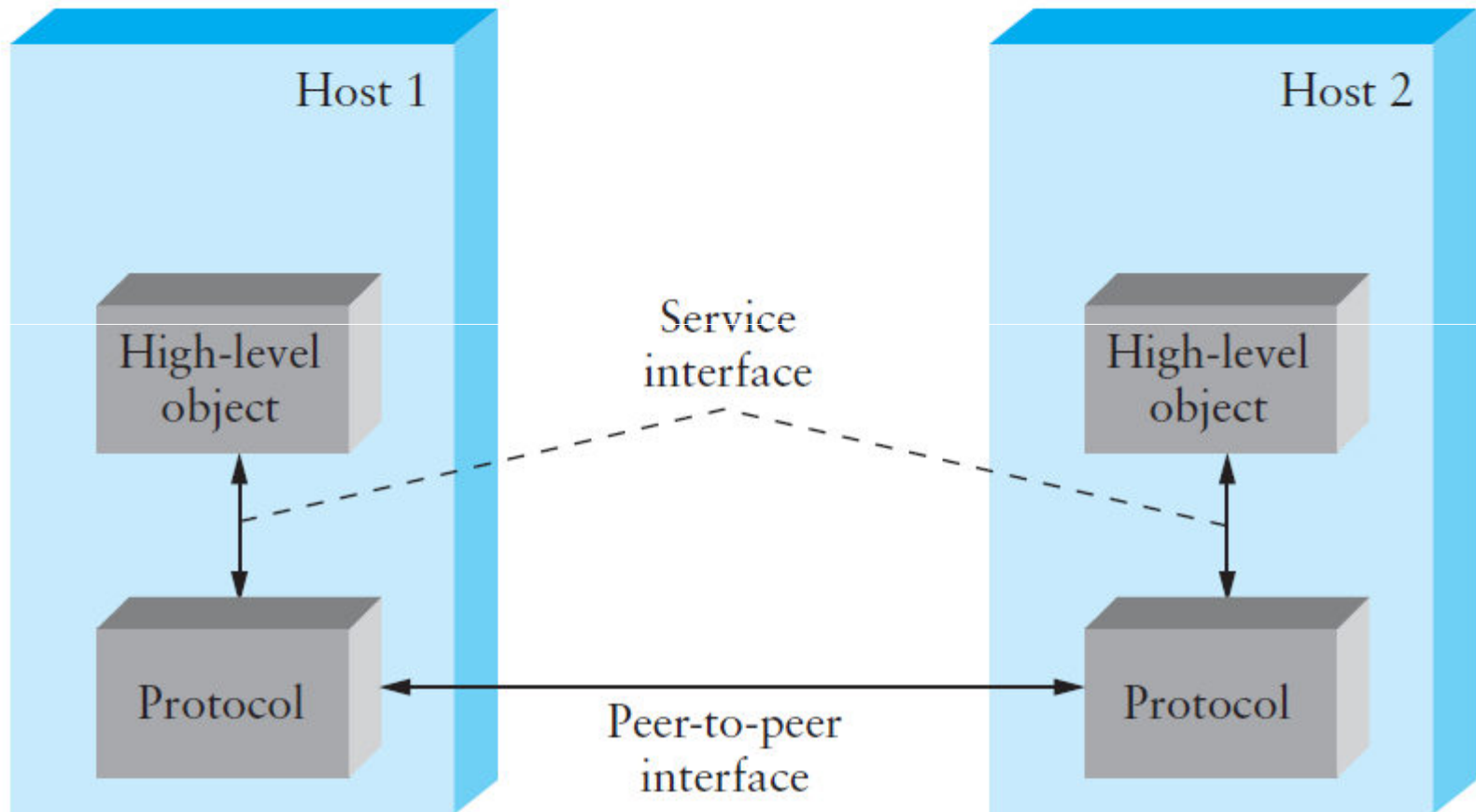
ADD ALL SALES TOGETHER

Data tier

Here information is stored and retrieved from a database or file system. The information is then passed back to the logic tier for processing, and then eventually back to the user.
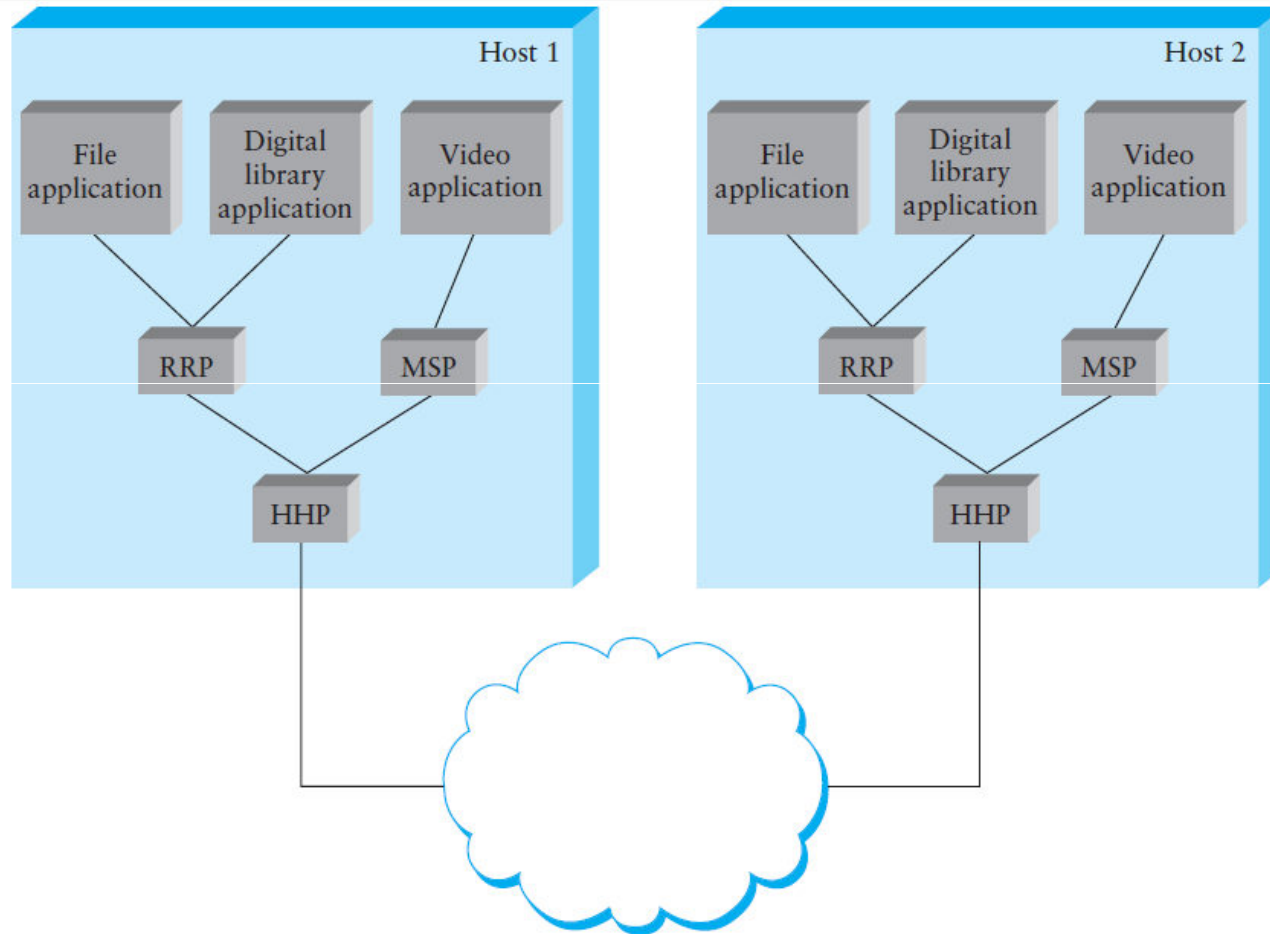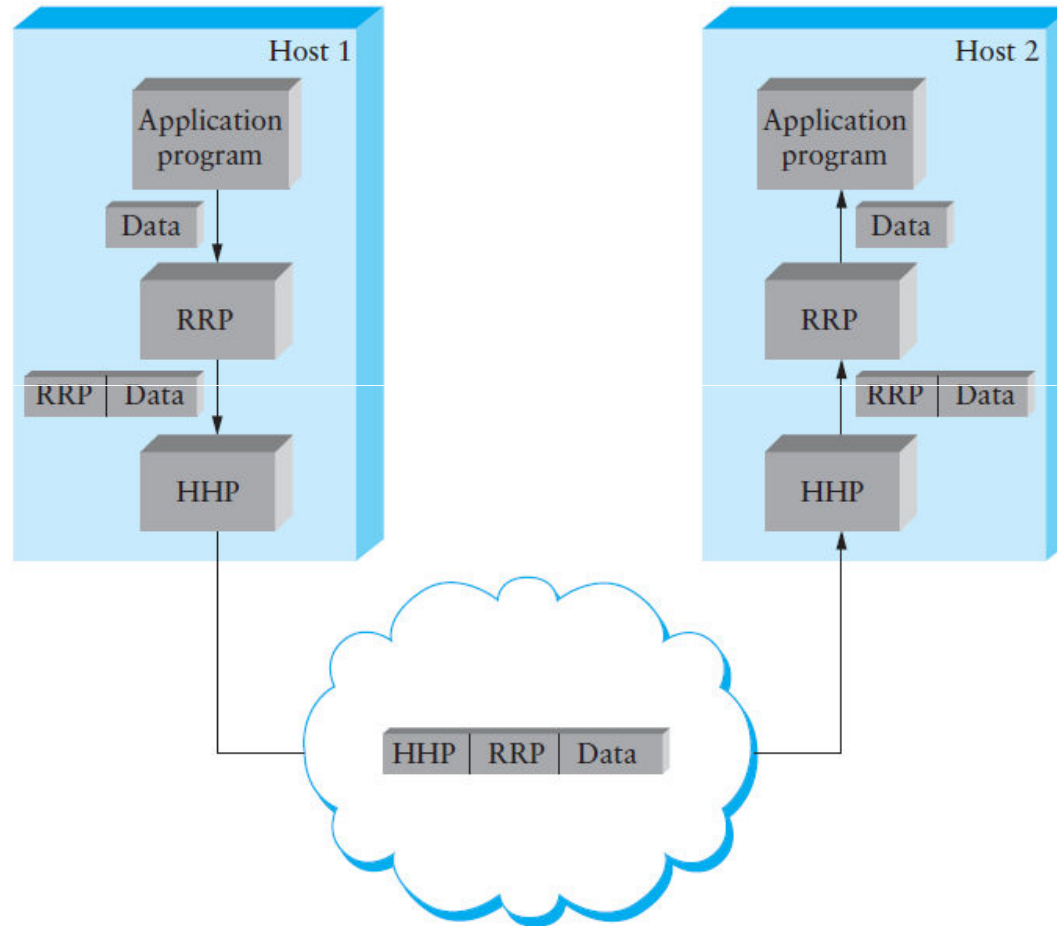
QUERY

SALE 1
SALE 2
SALE 3
SALE 4

Database

Storage

http://en.wikipedia.org/wiki/Multitier_architecture

Layering

Application programs	
Request/reply channel	Message stream channel
Host-to-host connectivity	
Hardware	

Service and peer interface

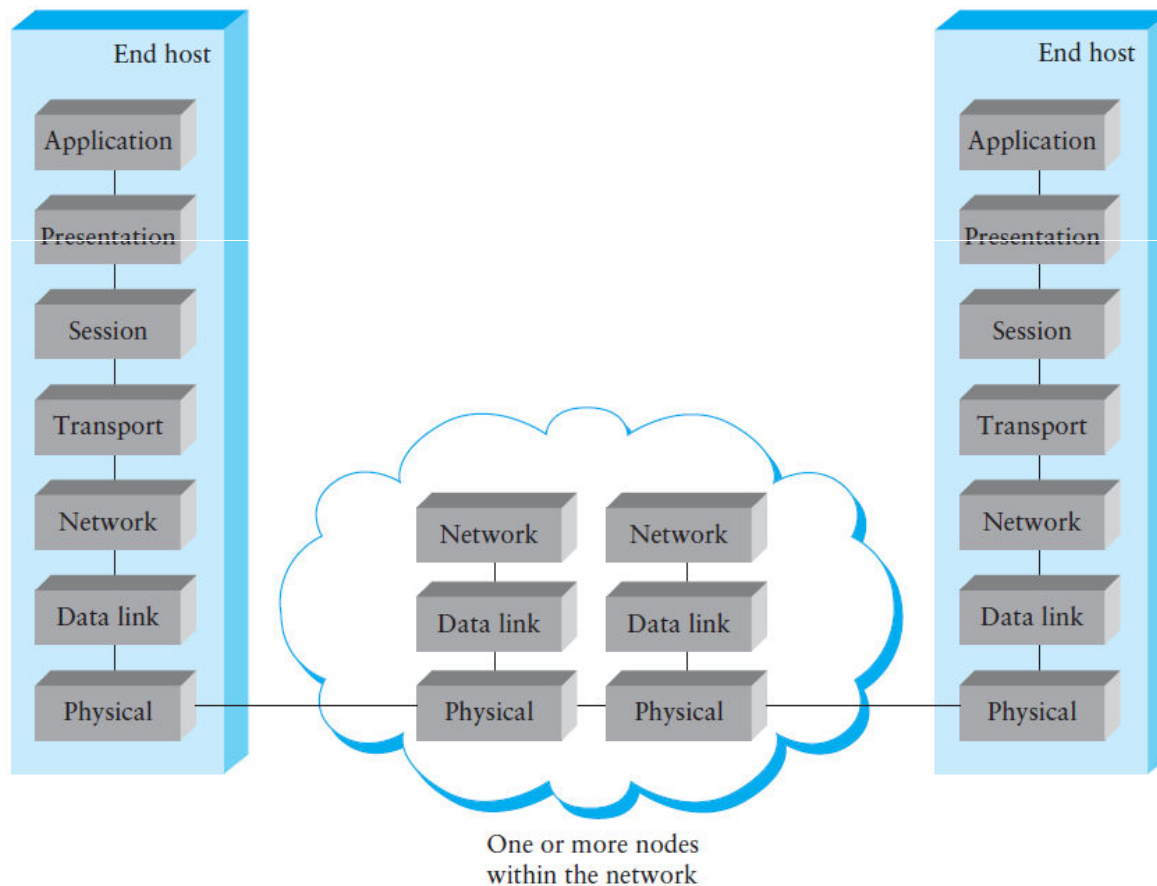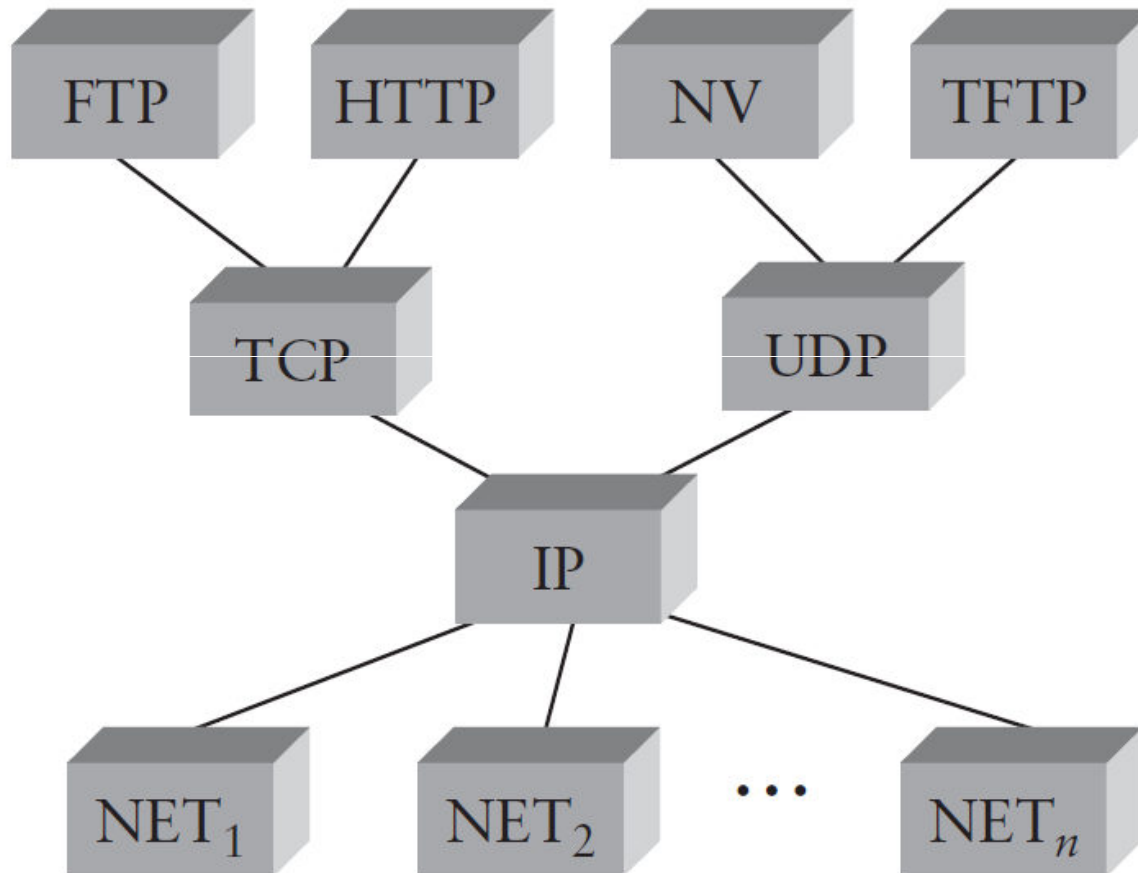# Network architecture as a protocol graph

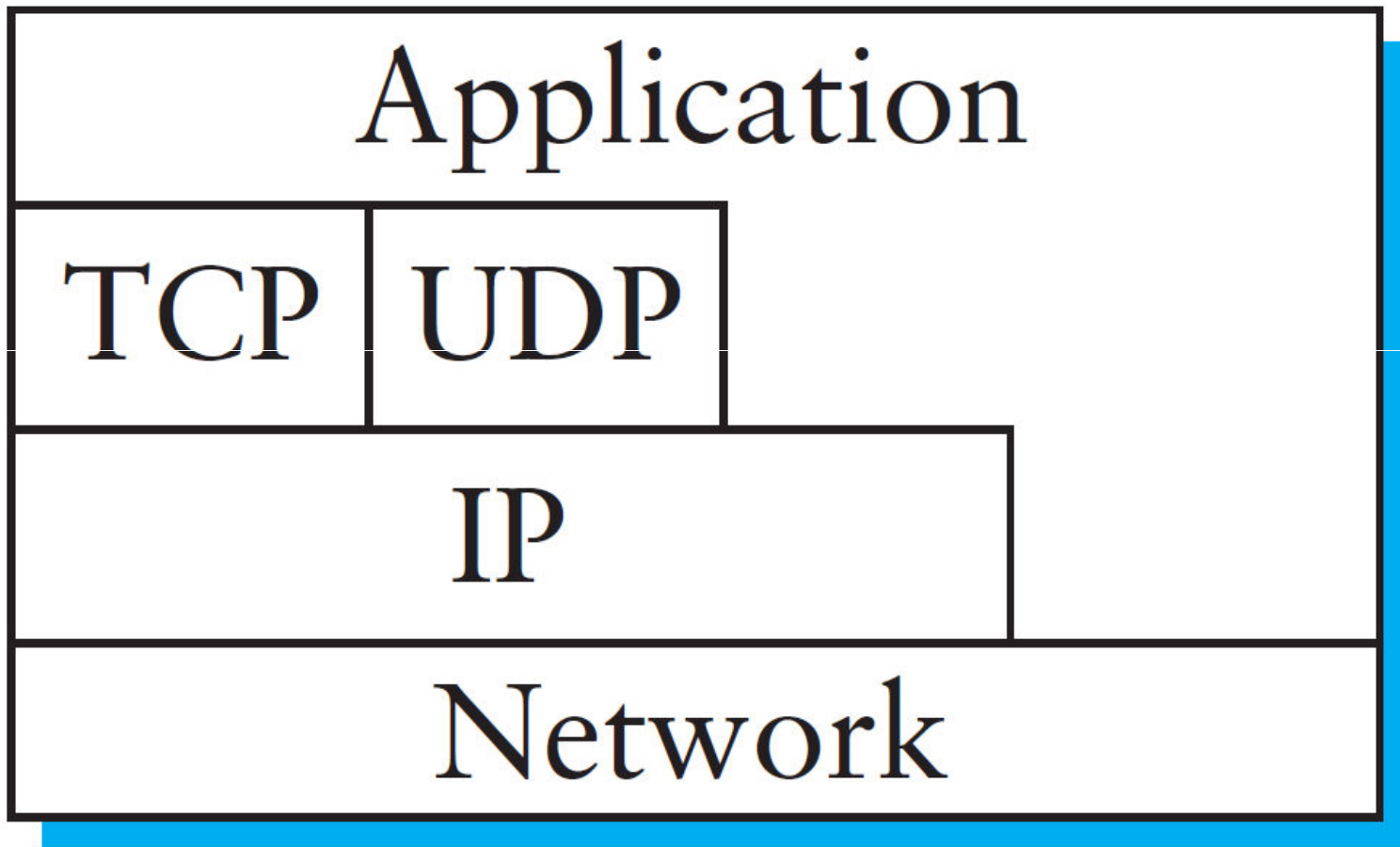# Encapsulation

Reference model

- OSI reference model

Internet Architecture protocol graph

Alternative view of Internet architecture

Implementing network software

- Protocol to protocol interface