CS 225
Advanced Distributed Systems
Introduction

Instructor: Jingzhu He

Spring, 2022

Logistics

- Instructor: Jingzhu He (hejzh1@shanghaitech.edu.cn)
  - Office: 1C-503B, SIST
  - Office Hours: 3:00-3:45pm Tu/Th
- TA: Dehong Chen (chendh1@shanghaitech.edu.cn)
  - Office: TBD
  - Office hours: TBD
- More information
  - https://jhe16.github.io/teaching/2014-spring-teaching-1

Course Goal

- This class is about learning fundamental concepts and the state of the art research results in distributed systems
- Focus on the intersection between distributed systems and machine learning
- Require real-world distributed system implementation
- Target audience: MS thesis students!

Course Requirements

- **Prerequisites**
  - **CS 130 (operating systems)**
  - **CS 120 (computer networks)**
  - **Programming skills in C++/Java and Linux**
- **What to expect**
  - Programming-intensive projects (a real distributed system that you can be proud of)
  - Research paper reviews for 2/3 semester

How will you Learn?

- Lectures
  - Canonical distributed system problems
  - Recent advances in distributed system research
- Read papers
  - Learn how to read and write by reading the state-of-the-art papers
- Projects
  - The best way to learn distributed systems to design and implement a cool distributed system by yourself
- Discussion
  - Ask questions (important)
  - Give feedback to your peers on their projects

How will you get an A?

- Write good paper reviews
  - One paper each week for 2/3 semester
  - 20%
- Class participation
  - Make one paper presentation (15%)
  - Attend the class, pass the quizzes, and participate discussions (15%)
- Projects
  - Project Proposal (5%)
  - Proposal Presentation (5%)
  - Project Mid-review (10%)
  - Project Demo (10%)
  - Final Presentation (10%)
  - Final Writeup (10%) – a research paper!

Get Project Started **NOW**

- Form your team
  - one or two members per group
  - Learn to conduct team work!
- Choose project ideas
  - See course homepage
  - Recent conference papers: SOSP/OSDI, NSDI, FAST, ICDCS, Middleware, DSN
  - Talk to the instructor
    - Make appointments by email
- Demo environments
  - Amazon AWS, Google Cloud, Microsoft Azure, HPC cluster

About Me

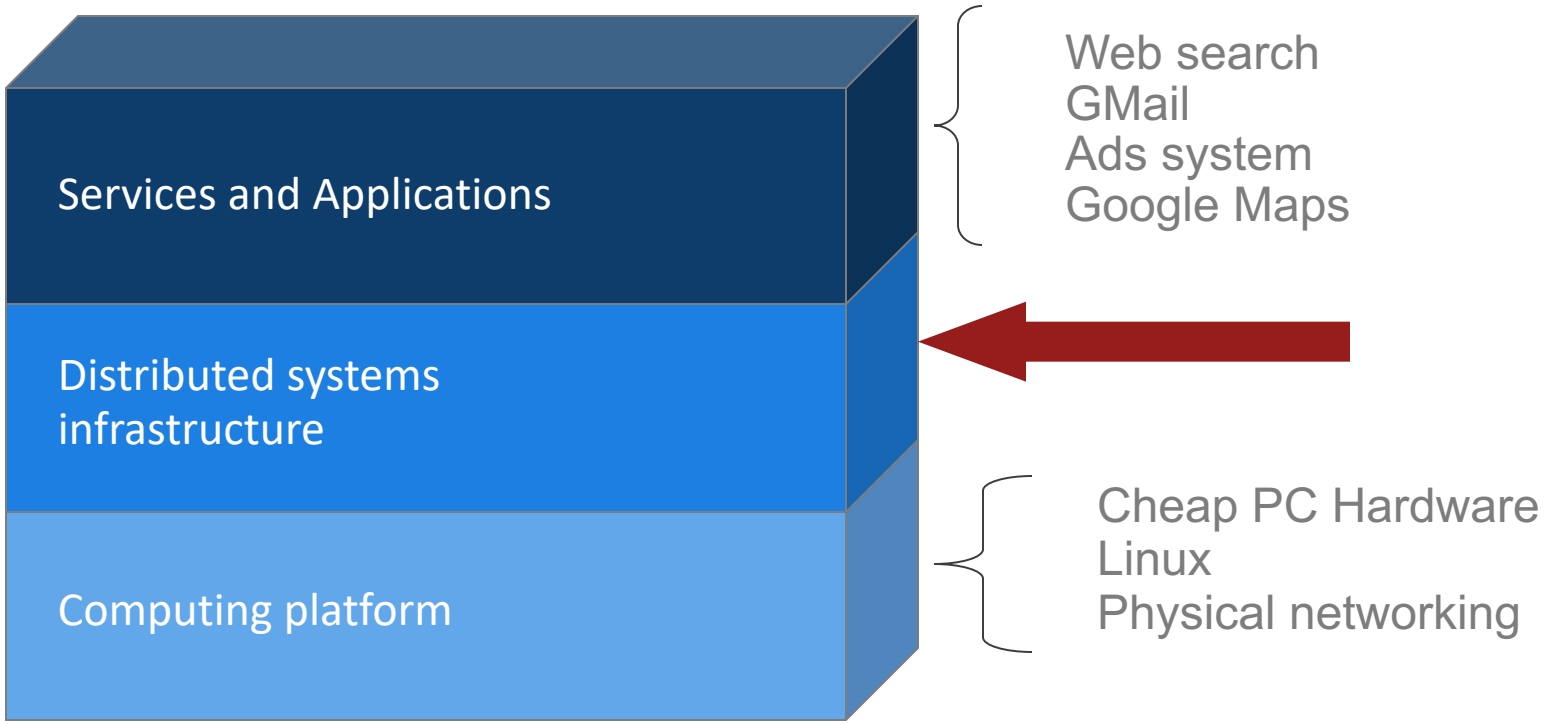
- Since 2021
  - Tenure-track Assistant Professor in ShanghaiTech University
  - lead the Cloud Intelligence lab
- Research
  - Improving reliability, availability, and performance of large-scale cloud systems
  - Automatic system debugging

What is a Distributed System?

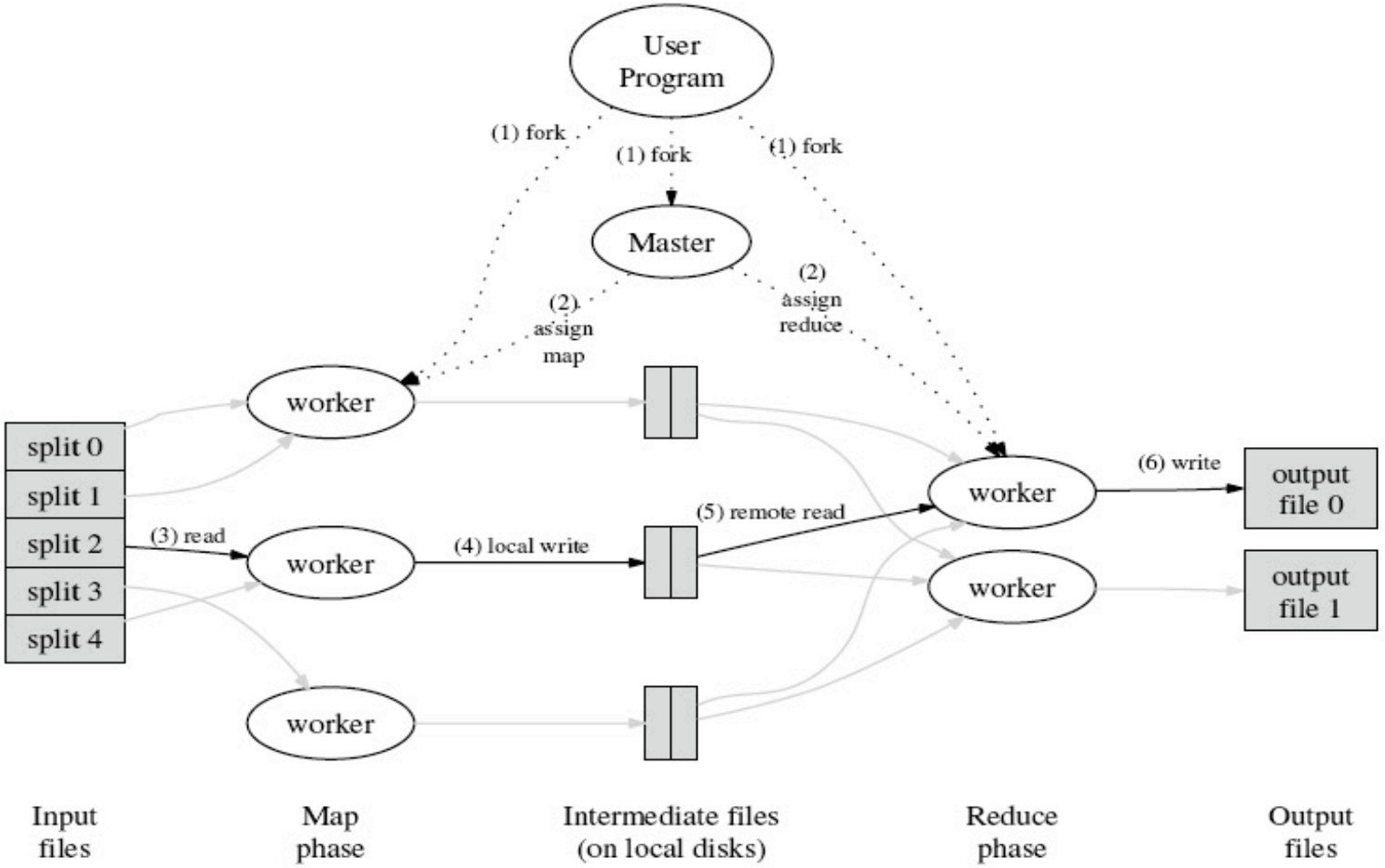
Some Examples

- Client-Server
- The Web
- The Internet
- A sensor network
- DNS
- Kazaa (peer to peer overlays)
- Data Center
- Stock Trading System
- Cluster
- Grid

Google Technology Layers



Google's Map/Reduce Execution



Web Hosting Systems

- Application software distributed among three types of machines
  - User machine
    - thin client
  - Middle-tier server
    - Gateway
    - Convert protocols
    - Merge/integrate results from different data source
  - Backend server

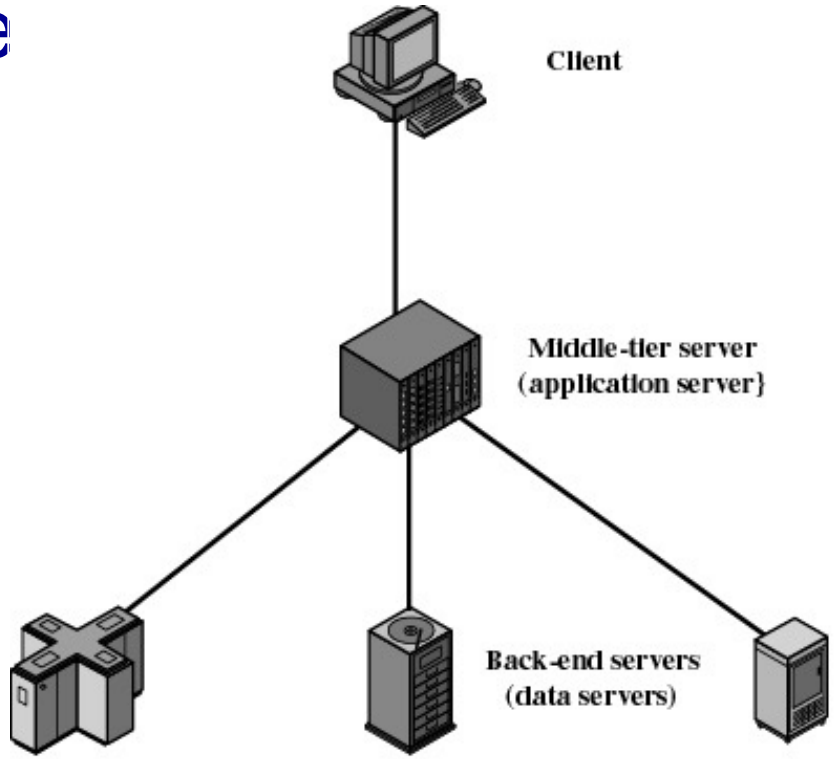


Figure 13.6 Three-tier Client/Server Architecture

Online Dictionary Definition

A collection of (probably heterogeneous) automata ***whose distribution is transparent*** to the user so that the system appears as one local machine. This is in contrast to a network, where the user is aware that there are several machines, and their location, storage replication, load balancing and functionality is not transparent. Distributed systems usually use some kind of client-server organization.

Textbook Definitions

- A distributed system is a collection of independent computers that appear to the users of the system as a single computer.

  [Andrew Tanenbaum]

- A distributed system is several computers doing something together. Thus, a distributed system has three primary characteristics: multiple computers, interconnections, and shared state.

  [Michael Schroeder]

Unsatisfactory

- Why are these definitions short?
- Why do these definitions look inadequate to us?
- Because we are interested in the insides of a distributed system
  - Design/Algorithms/Protocols
  - Implementation
  - Maintenance
  - Management

A working definition for us

A distributed system is a collection of entities, each of which is autonomous, programmable, asynchronous and failure-prone, and which communicate through an unreliable communication medium.

- Entity=a process on a host
- Communication Medium=Wired or wireless network

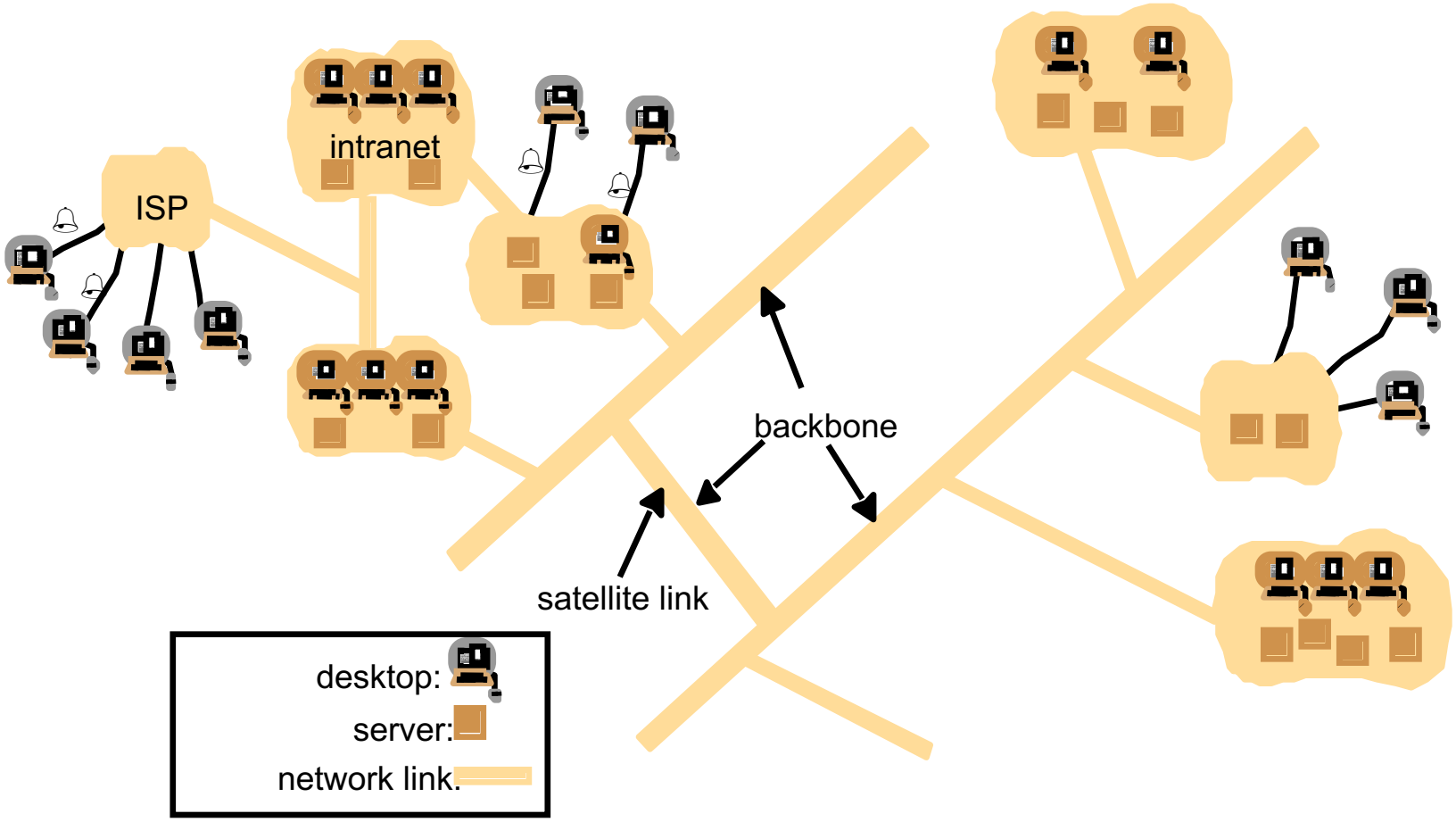
Distributed System Design Goals

- Robustness – is the system resilient to host crashes and failures, and to the network dropping messages?
- Availability – are data, services always there for clients?
- Transparency – can the system hide its internal workings from the users? i.e., Operating in such a way as to not be perceived by users.
- Heterogeneity – can the system handle different types of devices?

Distributed Systems Design Goals

- Concurrency – can the server handle multiple clients simultaneously?
- Efficiency – is it fast enough?
- Scalability – can it handle 100 million nodes? (nodes=clients and/or servers)
- Security – can the system withstand hacker attacks?
- Openness – is the system extensible?

Distributed System Example -- the Internet



The Internet

- A vast interconnected collection of computer networks of many types.
- Intranets – subnetworks operated by companies and organizations.
- ISPs – companies that provide modem links and other types of connections to users.
- Intranets are linked by backbones – network links of large bandwidth, such as satellite connections, fiber optic cables, and other high-bandwidth circuits.

Internet Apps: Their Protocols and Transport Protocols

Application	Application layer protocol	Underlying transport protocol
e-mail	smtp [RFC 821]	TCP
remote terminal access	telnet [RFC 854]	TCP
Web	http [RFC 2068]	TCP
file transfer	ftp [RFC 959]	TCP
streaming multimedia	proprietary (e.g. RealNetworks)	TCP or UDP
remote file server	NFS	TCP or UDP
Internet telephony	proprietary (e.g., Skype)	typically UDP

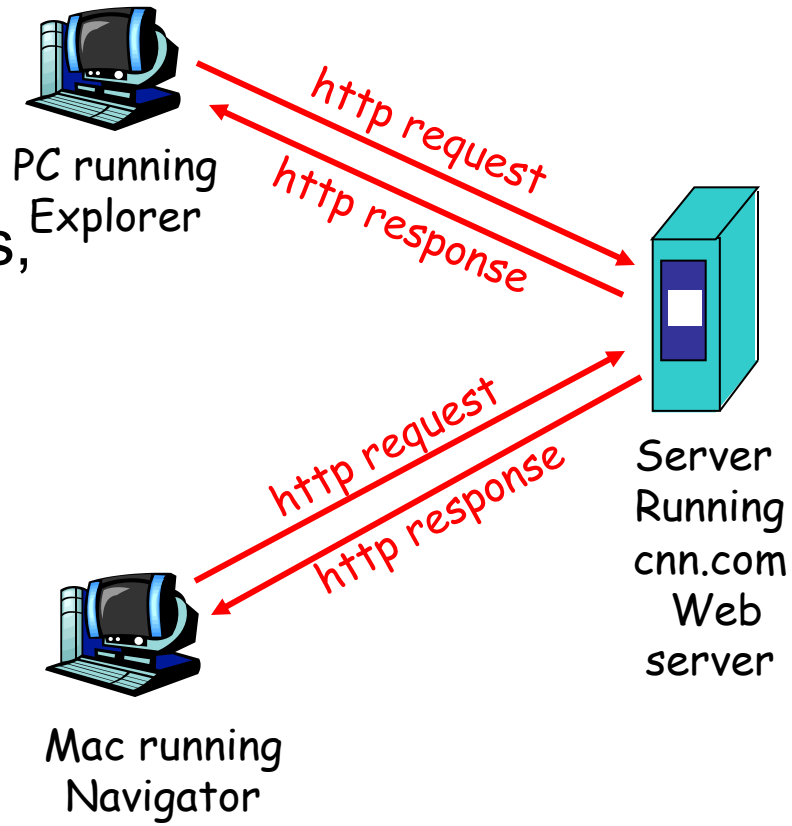
TCP=Transmission Control Protocol
UDP=User Datagram Protocol

Implemented via network "sockets". Basic primitive that allows machines to send messages to each other

WWW: the HTTP Protocol

HTTP: hypertext transfer protocol

- WWW's application layer protocol
- client/server model
  - *client:* browser that requests, receives, and "displays" WWW objects
  - *server:* WWW server stores the website, and sends objects in response to requests
- http1.0: RFC 1945
- http1.1: RFC 2068



The HTTP Protocol: More

http: TCP transport service:

- client initiates a TCP connection (creates socket) to server, port 80
- server accepts the TCP connection from client
- http messages (application-layer protocol messages) exchanged between browser (http client) and WWW server (http server)
- TCP connection closed

http is "stateless"

- server maintains no information about past client requests

aside

Protocols that maintain "state" are complex!

- past history (state) must be maintained
- if server/client crashes, their views of "state" may be inconsistent, and hence must be reconciled.

Distributed Software Systems

- Approaches:
  - Client/server model
  - Centralized control
  - Decentralized control
  - Peer-to-peer
  - Transactions

Client/Server Computing

- Client machines: single-user PCs/workstations
  - user-friendly interface
- Each server provides
  - shared user services
- Server enables many clients
  - to share access to same database
  - to use high-performance computer system (manage database)

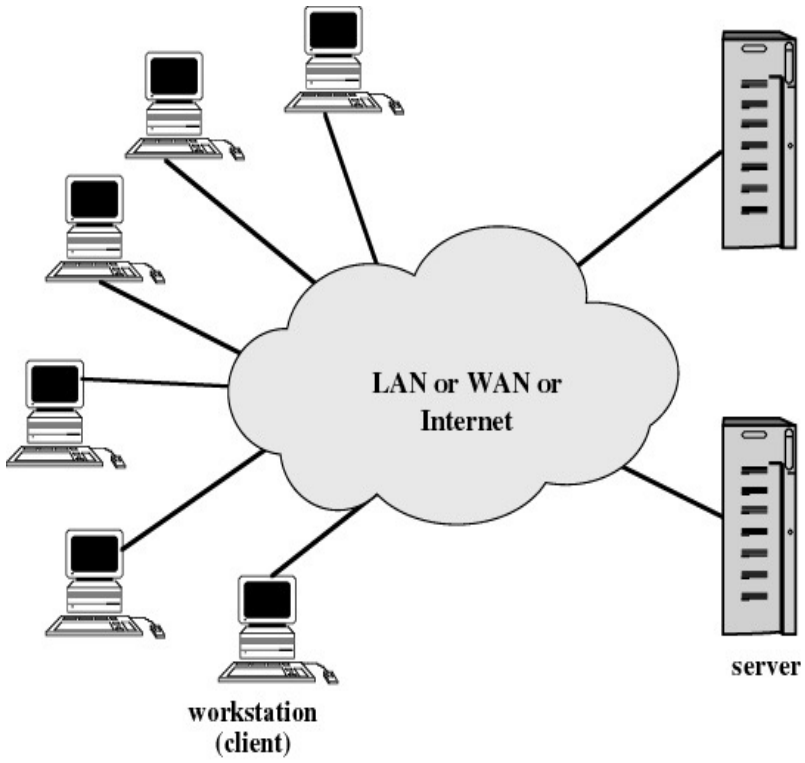
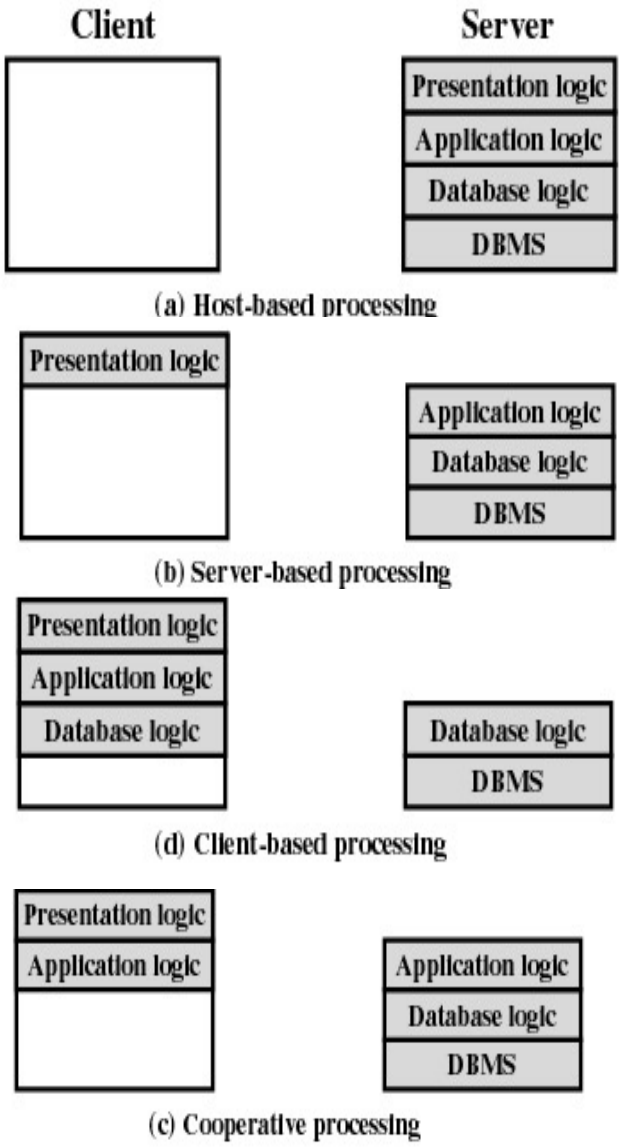


Figure 13.1 Generic Client/Server Environment

Classes of Client/Server Applications

1. Host-based processing
   - not true client/server computing
   - traditional mainframe environment
2. Server-based processing
   - Server: all processing
   - User: provides graphical interface
3. Client-based processing
   - Client: all processing data
   - Server: validation /database logic
4. Cooperative processing
   - processing optimized: client/server
   - complex to set up and maintain



Major Class Topics

- Basic Concepts
  - Clock Synchronization/Consensus, Replication
- Large-scale decentralized systems
- Big data
- System management using machine learning
- Cloud computing, Containers, Kubernetes
- Virtualization
- Research methodology & Presentation skills