Chapter 16

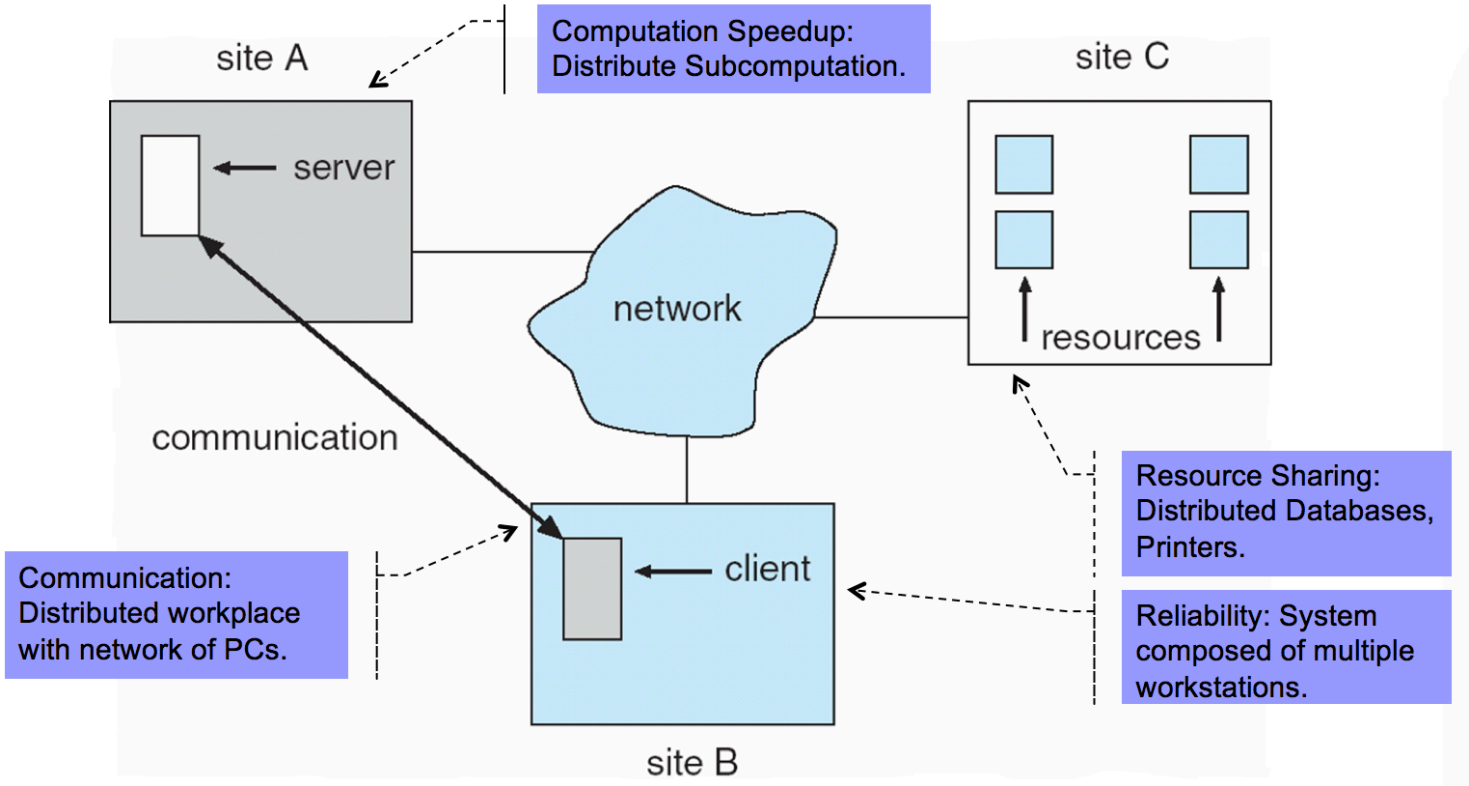
1. Distributed System Structures

1.1. Distributed System:

Definition: Collection of Processors that do not share memory or a clock, communicating through networks.

- **Processors**: varies in size, functions; including all types of computers.
- **Site**: location of a computer system.
- **Host**: specific system at a site (e.g., server, client)

Four **reasons** for Distributed Systems: **resource sharing**, **computation speedup**, **reliability**, and **communication**.



1.2. Network-Based Operating Systems

There are 2 types of network based OSs.

1.2.1. Network Operating Systems

User must know the techniques, or which resources to query, where to obtain the resources, i.e. command sets before hand. **Example**: *Remote logging* (telnet, ssh), *Remote desktop* (RDP), *Data Transfer* (s/FTP, SCP)

1.2.2. Distributed Operating Systems

User does not need to have special knowledge about remote system, just **access them as local resources**, all *data migration*, *computation migration* and *process migration* is taken care by the distributed OS.

Two techniques to move processes in network: by OS (transparent to users), or by users' specific inputs.

Characteristics of Client / Server Computing: ely on user-friendly app, share services on server side, common DB server, high priority on network management and secury.

1.2.3. Client / Server Computing

> Key feature is allocation of tasks between Client & Server.

> - Client / Server must share same protocol, support same app.
> - Easy to use GUI
> - Possible to have different OSes

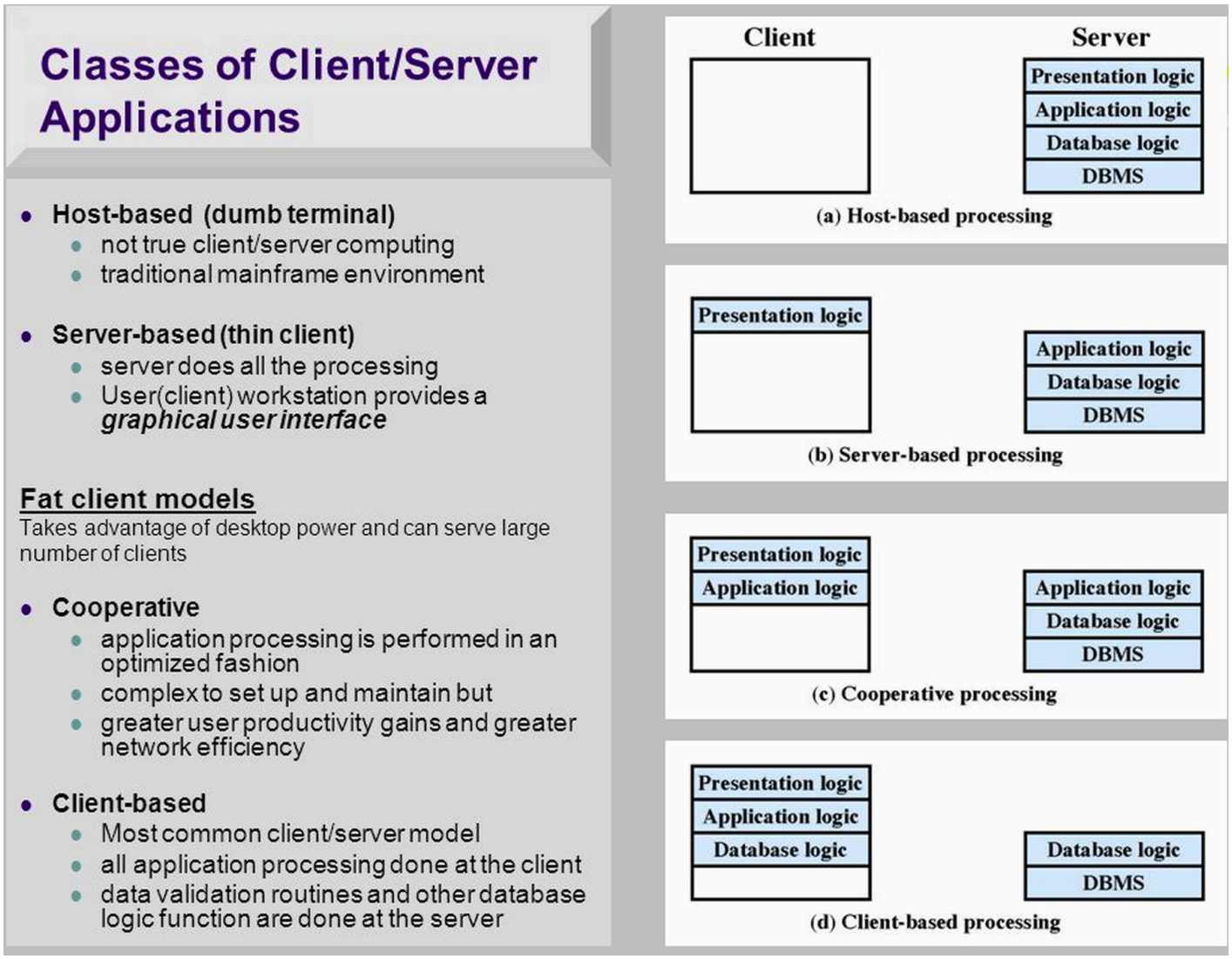
1. **Client/Server Classes**

   There are 4 classes of Client/Server Computing:

   - **Host-based**: dumb terminal, traditional mainframe
   - **Server-based – aka Thin Client**: server does all processing, client only does presentation.
   - **Cooperative (Fat Client)**: application logic is shared between client & server, complex to setup but greater user productivity & network effeciency.
   - **Client-based (Fat Client)**: **Most common model**, all processing done at client, only DB logic at server side. Represented by Relational Database, example is SQL database provides db service to Client side.

   Fat Client v/s Thin Client

	Fat Client	**Thin Client**
Advantage	less bottle neck at server side	migration path from mainframe to distributed computing network
Disadvantage	difficult to maintain, upgrade (involving hundress of desktops)	bottleneck at server side due to high processing load



2. **Three-Tier Architecture**: Thin client, middle tier server, and database.
3. **Middleware**: standard programming interface between client app, server service and OS.
4. **Distributed Message Passing**: rely on message passing to communicate between client & server. Example: RPC (Remote Procedure Calls): CALL P(X,Y), X is parameter, Y is return value.
    - There are **2 types of RPC binding**: Non-persistent (connection is closed after completion), and persistent (connection stays idle for a while).
    - There are 2 types of RPCs: synchronous (blocking) and asynchronous (nonblocking, parallelism).

2. Network Structure

> LAN, WAN

2.1. LAN

Network communication can be implemented by:

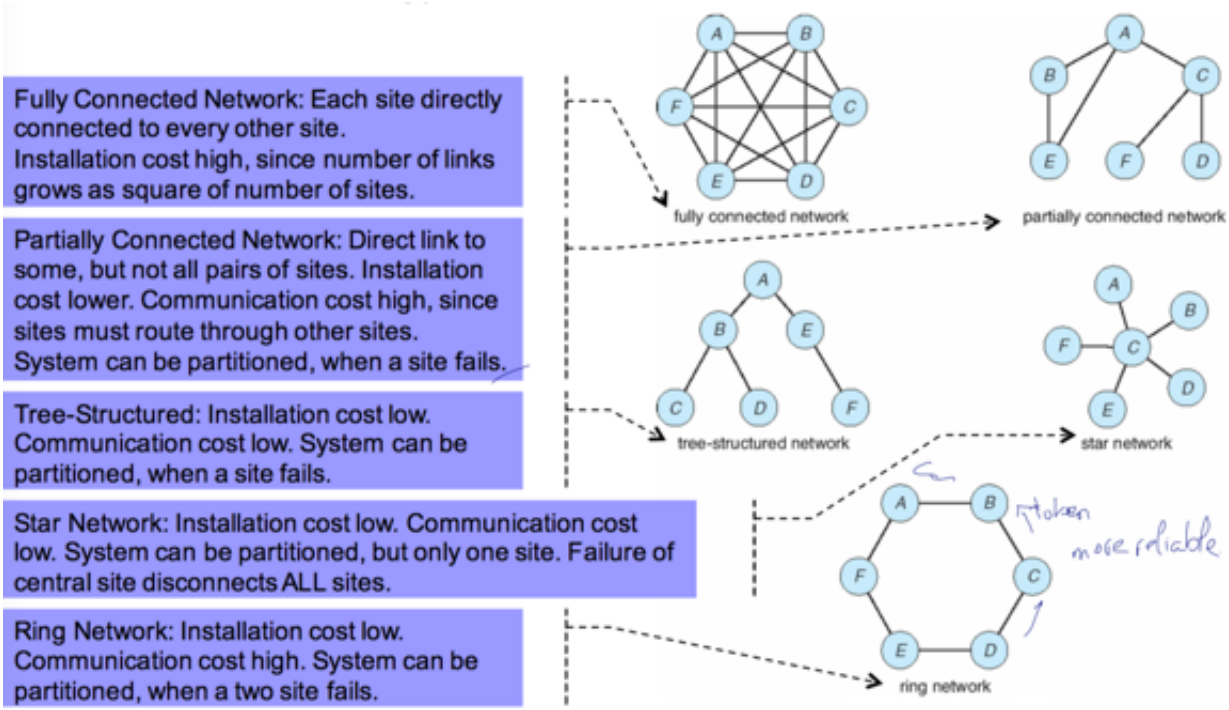
- **Ethernet** (multiaccess bus): simple, reliable, cost effective; or
- **Token Ring**: deterministic, far distance, great throughput under heavy load.
    - Cons: Require special HW thus increase cost, and risk of losing token.

2.2. WAN

> - Originated from Arpanet 1968, a packet switching network.
> - Or PPP (Point-to-Point) connection over modems.

3. Network Topology

> Has different topologies, such as Fully Connected Network, Partially Connected Network, Tree-Structured, Star Network, or Ring Network.



4. Communication Structure

5 basic problems to solve: **Naming and name resolution**, **Routing strategies**, **connection strategies**, and **Contention**.

4.1. Naming and Name Resolution

- Identify Process by `<Hostname, IP Address>`
- FQDN (Fully Qualified Domain Name): www.google.com, sau@svuca.edu, etc. The resolving is reversed order, right to left.

4.2. Routing Strategies

- **Fixed Routing**: direct path between A-B is pre-defined.
  - Pros: shortest path can be chosen to minimize cost, ensure ordering of messages.
  - Cons: unable to adapt load changes.
- **Virtual Circuit**: a path from A to B is defined for the session duration. Same pros as Fixed Routing, and improve the handling of load changes by specifying the path avoiding congestion.
- **Dynamic Routing**: path between A to B is chosen when a message is sent, by the router or hops.
  - Pros: adapt to load changes
  - Cons: message may arrive out of order.

4.3. Packet Strategies

- **UDP**: unreliable message, no guarantee it reach the destination.
- **TCP**: reliable message, guarantee reaching destination.

5. Communication Structure

3 Connection strategies (scheme):

1. **Circuit Switching** - a permanent link is established throughout the communication session (telephone/modem).
2. **Message Switching** - a temporary link is established during one message tranfer.
3. **Packet Switching** - split the message into smaller fixed-length packets and dispatching to the destination, regardless of orders, routes.

Pros/Cons:

- Circuit Switching requires more setup time, but less overhead for each message, may waste bandwidth if idle.
- Message and Packet switching less setup time, but more overhead per message.

5.1. Contention

There are 2 methods to solve contention in communication.

1. **Token Passing**: circulates a token in the system.
2. **Message Slots** (Ring structure): reserve slots for communication.

6. Communication Protocols

- ISO (from bottom up): Transport layer, Session layer, Presentation layer, Application layer.
- TCP/IP (from bottom up): Physical, Data Link, IP, TCP-UDP, (HTTP, DNS) etc.

[]()

7. Robustness

> Able to detect link failure, site failure or message lost.

8. Reconfiguration

> Ability to reconfigure & recover on failure and resume operation.

9. Design Issues

To be considered when designing a distributed system structures:

- Transparency
- Fault Tolerance
- Scalability
- Clusters

10. Questions

Qn: What are the reasons of using Distributed Systems?
Ans:

Qn: What is Network Operating System, give an example.

Qn: What is Distributed Operating System, give an example.
Example: **WWW** has many aspec of a distributed computing environment: data migration (client/server), computation migration (web client triggers db operation on server), process migration (java applet is used to take care of some execution on client side)

Qn: What are 4 classes of Client / Server Computing? Which one is the most common? What are fat client, thin client, compare.

Qn: What is the pros, cons of Token Ring network?

Qn: which routing strategy is more adaptable to load changes?

Qn: which switching method with var length can take different path in the network?
Ans: Packet Switching.