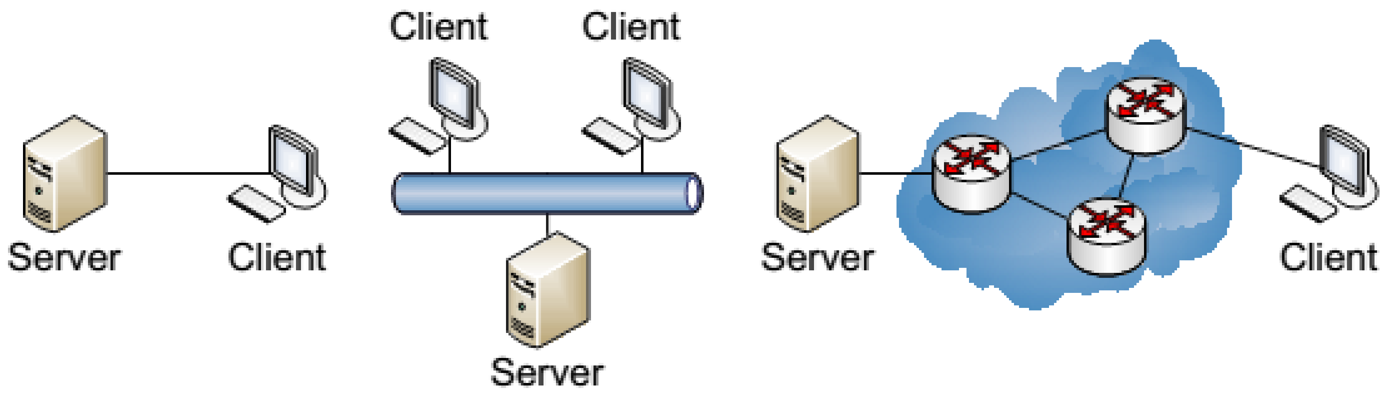
CSC 402 – Internet Technology

Recap

- Mail Access Protocol
- POP3
- POP3 Drawbacks
- IMAP
- Telnet
- SSH
- PGP

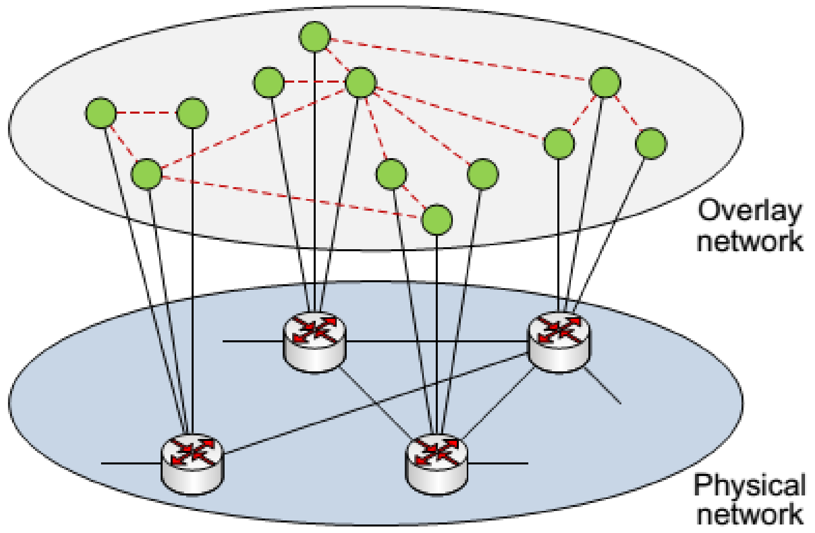
Interaction Model

- End systems can be positioned on a network in different ways relative to each other i.e., they can be made to communicate and share resources according to different interaction models.
- 2 fundamental interaction models :
  - Client/server
  - Peer-to-peer (aka P2P)
- These models are relevant to end systems only, regardless of how the end systems are connected to each other.



Interaction Model

- Client/server and P2P protocols operate at the application layer of the TCP/IP model.
  - Application layer protocols are end-to-end protocols.
- Client/server and P2P systems are implemented as virtual networks of nodes and logical links built on top of an existing (aka underlay) network, typically the Internet.
  - These virtual networks are called overlay networks.
  - The overlay is a logical view that might not directly mirror the physical network topology.
- Overlay networks means:
  - Tunnels between host computers
  - Hosts implement new protocols and services
  - Effective way to build networks on top of the Internet



Client/Server Model

- In the client/server model, all end systems are divided into **clients** and **servers** each designed for specific purposes
- Clients have an active role and initiate a communication session by sending requests to servers
  - Clients must have knowledge of the available servers and the services they provide
  - Clients can communicate with servers only; they cannot see each other
- Servers have a passive role and respond to their clients by acting on each request and returning results
  - One server generally supports numerous clients

Client/Server Model

- Hardware roles :
  - The terms ”client” and ”server” usually refer to the primary roles played by networked hardware
  - A ”client” is usually something like a PC used by an individual, and primarily initiates conversations by sending requests
  - A ”server” is usually a powerful machine dedicated to responding to client requests, sitting in a server room somewhere that nobody but its administrator ever sees
- Software roles :
  - TCP/IP uses different pieces of software for many protocols to implement ”client” and ”server” roles
  - Client software is usually found on client hardware and server software on server hardware, but not always
  - Some devices may run both client and server software
- Web clients: Mozilla Firefox, Internet Explorer, Google Chrome, . . .
  - See “Web Statistics” by W3Schools www.w3schools.com/browsers/browsers stats.asp
- Web servers: Apache, Microsoft IIS, GWS, . . .
  - See “Web Server Survey” by Netcraft Ltd. http://news.netcraft.com/archives/category/web-server-survey/

Client/Server Model

- Transactional roles :
  - In any exchange of information, the client is the entity that initiates communication or sends a query; the server responds, usually providing information.
  - Again, usually client software on a client device initiates a transaction, but this is not always the case.
  - E.g., when 2 SMTP servers communicate to exchange electronic mail, even though they are both server programs running on server hardware, during any transaction one device acts as client, while the other acts as server.

Client/Server Model – Server Types

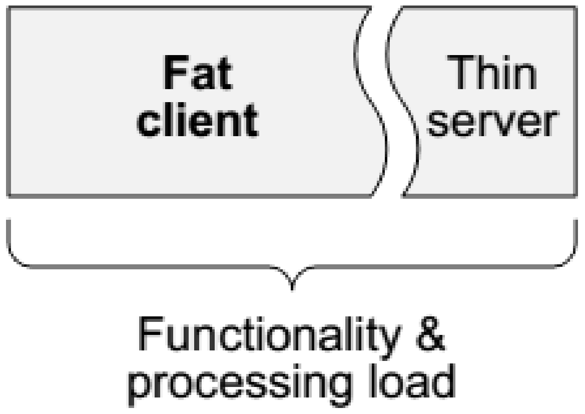
- The purpose of servers is to provide some predefined services for clients. 2 types of servers :
  - Iterative
  - Concurrent
- **Iterative servers** iterate through the following steps:
  - Wait for a client request to arrive
  - Process the request and send the response back to the client
  - Go back to Step 1
- Thus, iterative servers handle clients sequentially, finishing with one client before servicing the next
- **Concurrent servers** perform the following steps:
  - Wait for a client request to arrive
  - Use a new process/task/thread to handle the request
  - Go back to Step 1
- Thus, concurrent servers handle client requests in parallel
- 2 approaches to implement concurrent servers:
  - **Thread-per-client** – a new thread is spawned to handle each request
  - **Thread pool** – a prespawned set of reusable threads which take turns (round-robin) handling incoming requests

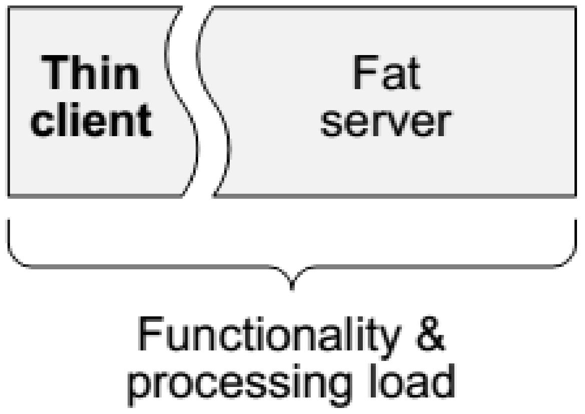
Client/Server Model – Server Types

- **Iterative or concurrent?**
- Iterative design is quite simple and is most suitable for short-duration services that exhibit relatively little variation in their execution time.
  - I.e., if the time to handle a client can be long, the waiting time experienced by subsequent clients may be unacceptable.
  - Internet services like echo (RFC 862) and daytime (RFC 867) are commonly implemented as iterative servers.
- Concurrent design is more complex but yields better performance.
  - It allows to improve responsiveness and reduce latency when the rate at which requests are processed is less than the rate at which requests arrive at the server.
  - Internet services like HTTP, telnet, and FTP are commonly implemented as concurrent servers
- As a rule, TCP-based servers are concurrent and UDP-based servers are iterative

Client/Server Model – Client Types

- Clients are devices/programs that request services from servers
- Clients (and, hence, servers) can be distinguished according to the functionality they provide and the amount of processing load they carry
- 2 types of clients :
  - Fat (aka thick or full)
  - Thin (aka slim or lean)





Client/Server Model – Client Types

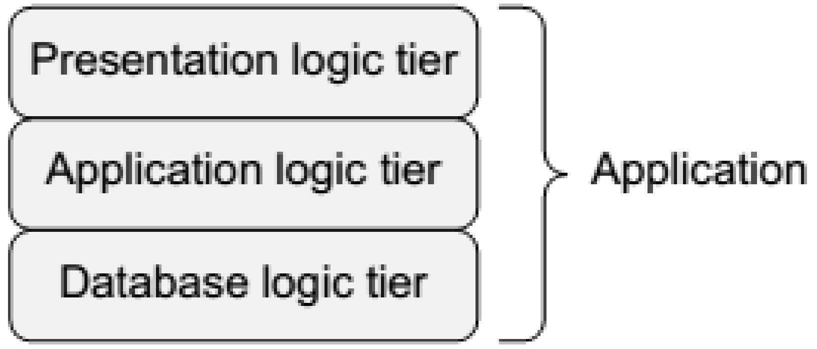
- Fat clients are devices/programs that are powerful enough and operate with limited dependence on their server counterparts
- Fat clients as devices – a user workstation that is powerful and fully-featured in its own right
  - E.g., a desktop PC, a laptop, a netbook
- Fat clients as programs – a client carries a relatively large proportion of the processing load
  - E.g., the Lineage II gaming client (more than 2 GB in size)
- Thin clients are devices/programs that have very limited functionality and depend heavily on their server counterparts
- Thin clients as devices – a user workstation that contains a minimal operating system and little or no data storage
  - E.g., Computers in labs that uses a central server to process commands.
- Thin clients as programs – a client mainly provides a user interface, while the bulk of processing occurs in the server
  - E.g., the OnLive gaming client (about 10 MB in size)

Client/Server Model – Client Types

- Fat or thin? Both technologies have their positive and negative aspects
- Benefits of thin-client systems:
    - No viruses, spyware, spam, thefts, etc.
    - Easy to keep the software properly configured and patched.
    - Lower TCO (Total Cost of Ownership).
    - Fewer points of failure.
- Shortcomings of thin-client systems:
    - Servers are the central point of failure.
    - Systems tend to be proprietary.
    - Multimedia-rich applications require a significant amount of bandwidth to function to their maximum potential (e.g., OnLive recommends a 5 Mbit/s connection speed or higher).
- Today, there are a lot of factors driving IT towards thin clients:
    - Desktop virtualization and increase in Web-based applications.
    - High-bandwidth LAN and WAN connections.
    - Data security concerns.
    - Energy savings.
    - Advances in low-cost computers (i.e., netbooks and, possibly, nettops).

Client Server Model – Logical Tiers

- In general, application software can be divided into 3 logical tiers :
  - Presentation logic
  - Application logic (aka business rules)
  - Database logic
- Each tier in the software is responsible for a specific task in the application
- This layering is a reference model; in practice, the boundaries may be not so sharp
- The logical layering of application software does not need to be the same as its physical layering

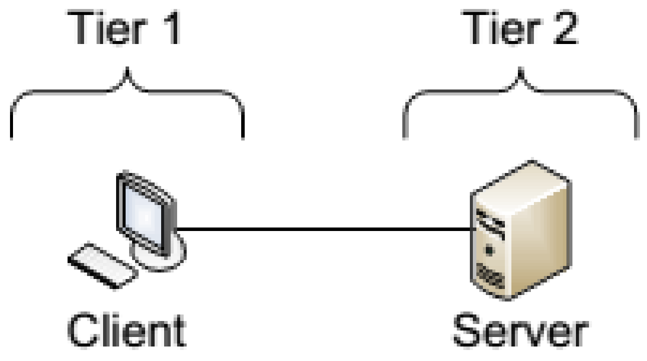


Client Server Model – Logical Tiers

- **Presentation logic** is responsible for displaying the information and interacting with the user (i.e., user interface)
  - It must make the information available in a suitable form to the user and must respond appropriately to input from the user
- **Application logic** processes commands, makes logical decisions, performs calculations, and coordinates the application
  - It also moves and processes data between the presentation logic and database logic tiers
- **Database logic** refers to the management of underlying databases
  - It is responsible for storing and retrieving the data according to the requirements of the application logic tier
  - A static content is typically stored in a file system

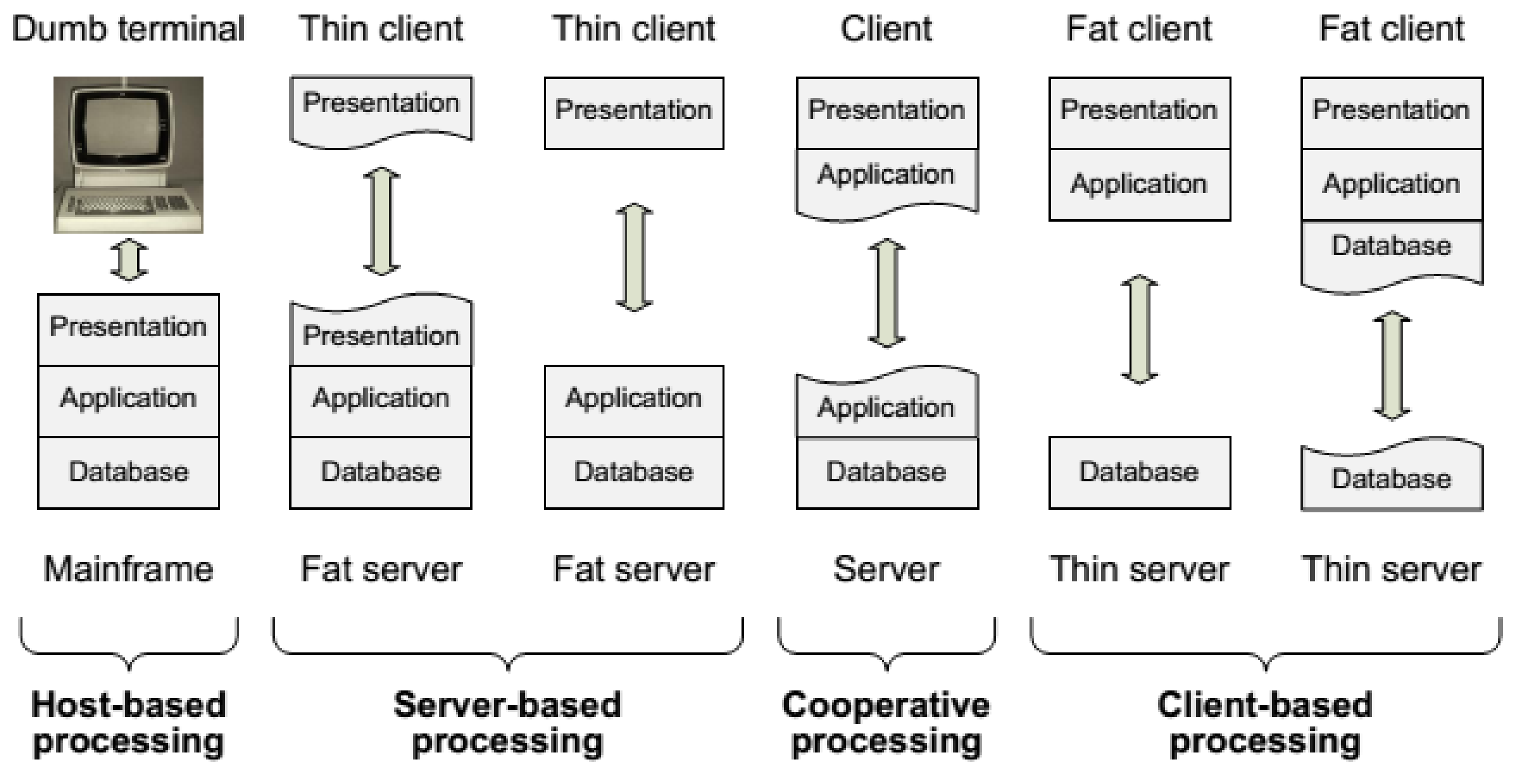
Client Server Model – Physical Tiers

- **1-tier architecture** is used to describe systems in which all of the processing is done on a single host
- Users can access such systems (aka mainframes) through display terminals (aka dumb terminals) but what is displayed and how it appears is controlled by the mainframe
- **2-tier architecture** (aka **flat**) is used to describe client/server systems, where clients request resources and servers respond directly to these requests, using there own resources



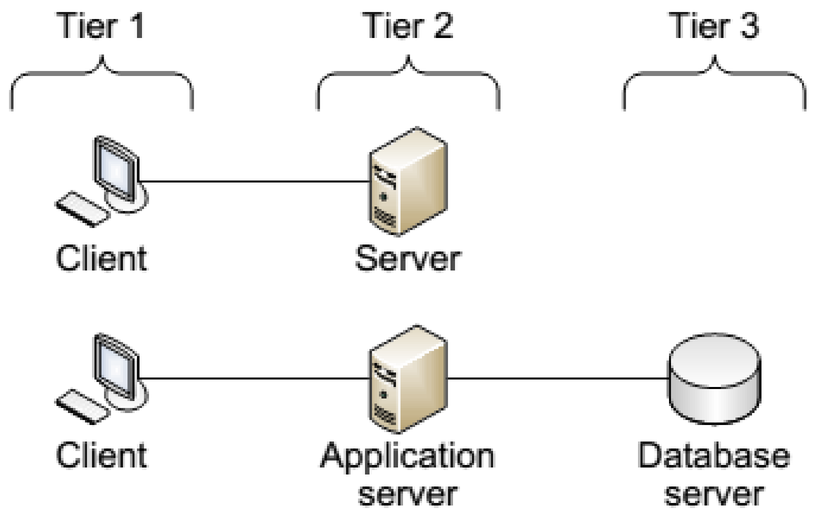
Client Server Model – Physical Tiers

- Distributing logical tiers between 2 physical tiers



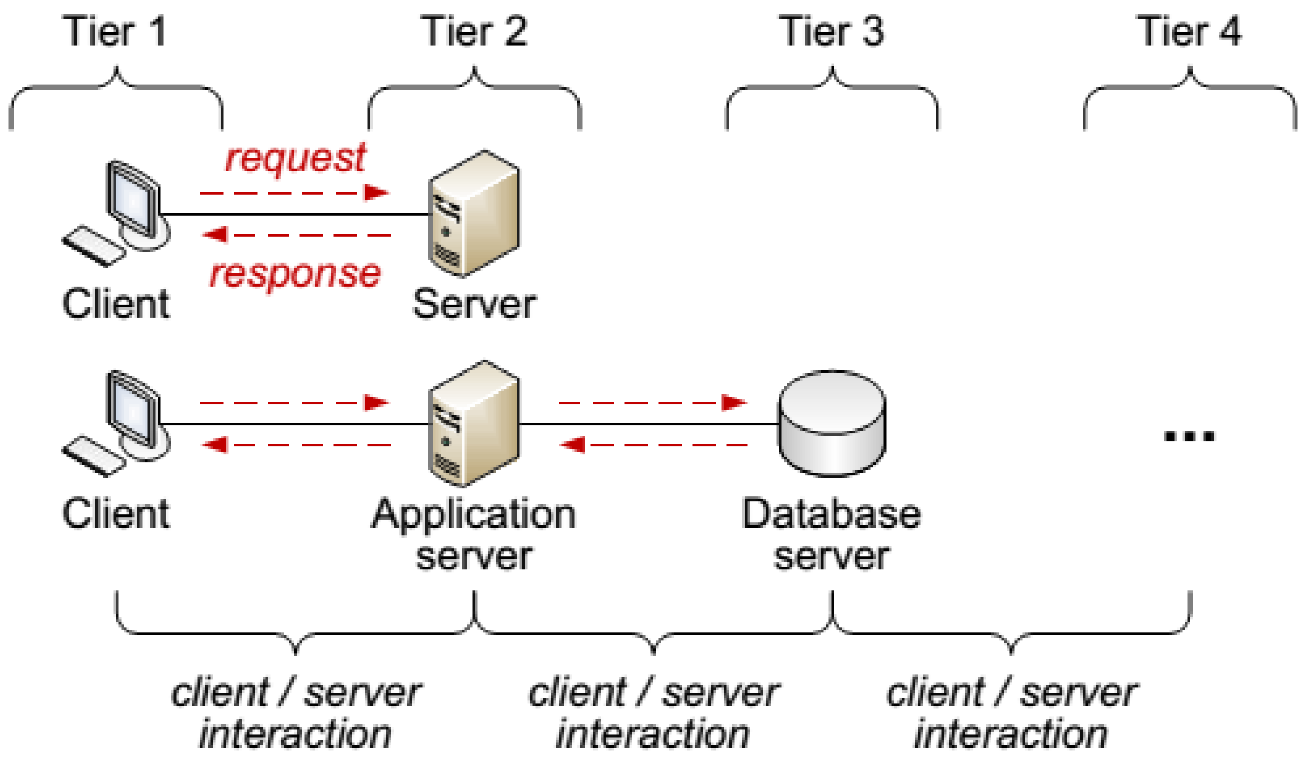
Client Server Model – Physical Tiers

- 3-tier architecture is used to describe client/server systems consisting of:
  - Clients which request services
  - Application servers whose task is to provide the requested resources, but by calling on database servers
  - Database servers which provide the application servers with the data they require



Client Server Model – Physical Tiers

- N-tier architecture is used to describe client/server systekms consisting of more than 3 tiers



Client Server Model – Pros and Cons

- **Benefits** of 3- and N-tier (aka multi-tier or hierarchical) architectures:
  - Increased flexibility, as changes to the presentation/application/database logic are largely independent from one another.
  - Increased security, as security can be defined for each service and at each tier.
  - Increased performance, as tasks are shared between tiers.
  - Distribution of data is straightforward.
  - Makes effective use of networked systems.
  - Easy to add new servers or upgrade existing servers.
- **Shortcomings** of multi-tier architectures:
  - Increased complexity of development and testing.
  - Increased need for load balancing and fault tolerance.
  - No shared data model so data interchange may be inefficient.
  - Redundant management in each server
  - No central register of names and services - it may be hard to find out what servers and services are available.