# CLIENT / SERVER && PEER-TO-PEER MODELS

- Communication network - a system of interconnected end systems, intermediate systems, and other equipment allowing information to be exchanged.

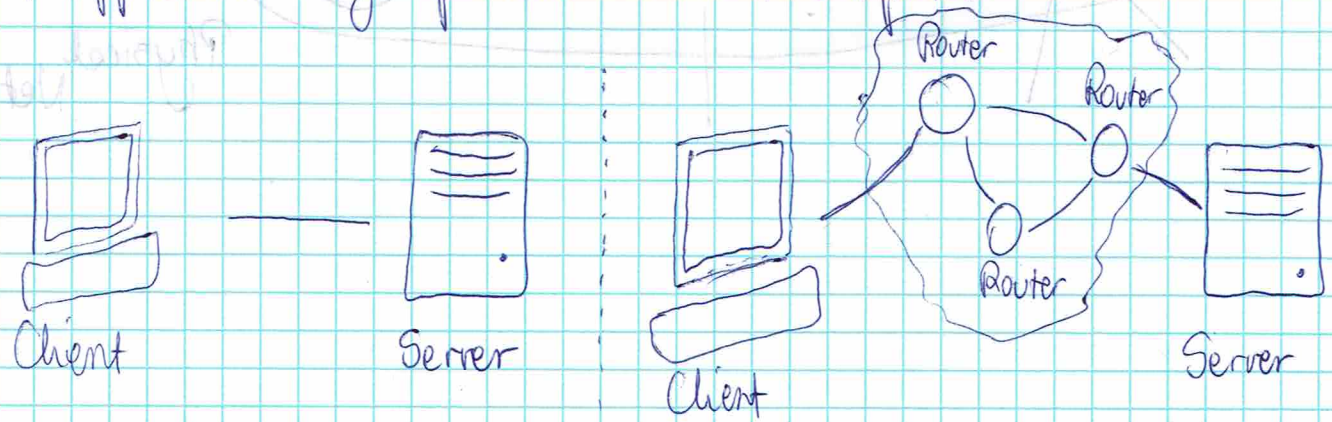
- Intermediate systems - a devices that operates as a relay element between 2 or more end systems (networks)
  (router, switch, hub, bridge ...)

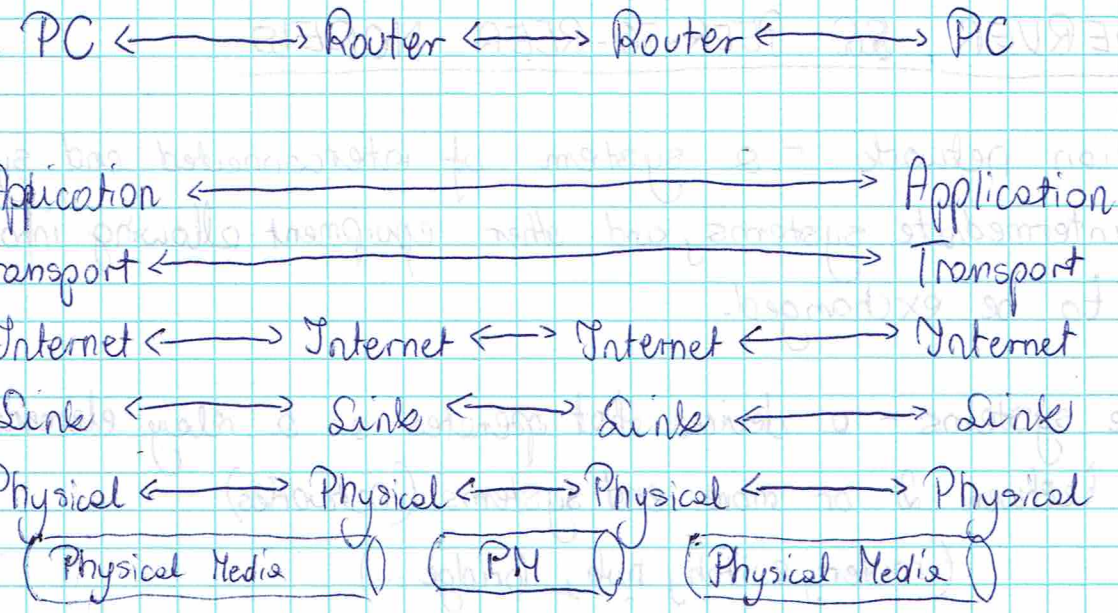
- End system - a device that uses or provides end-user applications or network services (PC, web server, DNS server)
  * they are labeled "end systems" because they sit at the edge of a network
  * End systems that are connected to the Internet are also reffered to as "hosts", because they host (run) Internet applications

- 2 fondamental interaction models
  - Client / Server
  - P2P

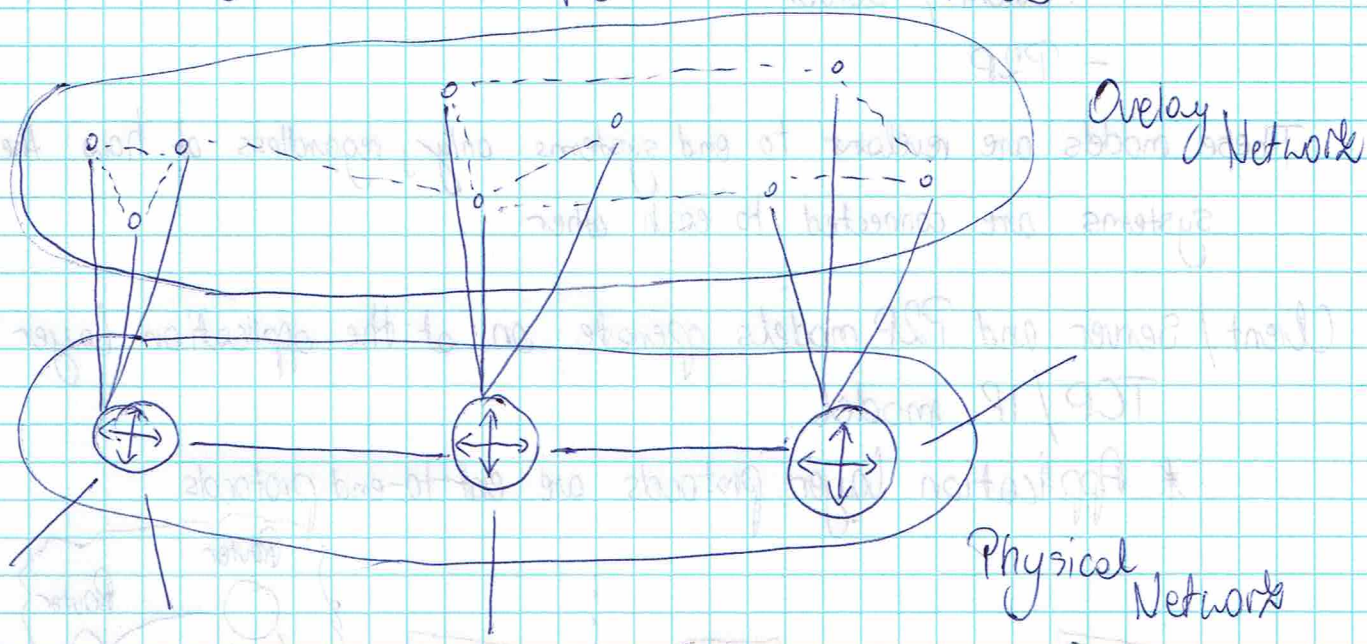
These models are revelant to end systems only, regardless of how the end systems are connected to each other.

- Client / Server and P2P models operate on at the application layer of the TCP / IP model.
  * Application layer protocols are ent-to-end protocols.






- Client/Server and P2P systems are implemented as virtual networks of nodes and logical links built on top of existing (aka underlay) network, typically the Internet.
  - * They are called overlay networks.
  - * The overlay ~~network~~ is a logical view that might not directly mirror the physical network topology.



- 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.
    - they must have knowledge of the available servers and the services they provide
    - they can communicate with servers only
  * SERVERS have a passive role and respond to their clients by acting on each request and returning results.
    - one server generally supports numerous clients.

- Software roles:
  * TCP/IP uses different pieces of software for many protocols to implement "client" and "server" roles.
  * Some devices may run both client and server software.

- Transactional roles:
  * In any exchange of information, the client is the entity that initiates communication or sends a query; the server responds usually by sending information.

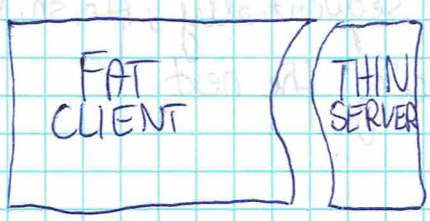
- 2 types of servers
  * Iterative servers which iterate through following steps:
    1. Wait for a client request to arrive
    2. Process the request and send the ~~process~~ response back
    3. Go back to step 1
    - They handle clients sequentially, finishing with one client before servicing the next.

* Concurrent servers perform the following steps
  1. Wait for a client request to arrive
  2. Use a new process / task / thread to handle the request
  3. Go back to step 1
  - They handle clients requests in parallel

- 2 approaches to implement concurrent server
  * Thread-per-client - a new thread is spawned to handle each request
  * Thread pool - a prespawned set of reusable threads which ~~turns~~ take turn (round-robin) handling incoming requests.

- Which one should we use?
  * Iterative design is quite simple and is most suitable for short duration services that exhibit relatively little variation in their execution time.
  * Concurrent design is more complex but yields better performance
    - It allow 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.
  * As a rule; TCP-based servers are concurrent; UDP-based on the other hand are iterative.

- 2 types of clients
  * Fat (also thick or full)
    - Fat clients are devices / programs that are powerful enough and operate with limited dependence on their server counterparts.




Functionality && Processing Load

* Thin (aka slim or lean)
  - Thin clients are devices / programs that have very limited functionality and depend heavily on their server counterparts.

- Benefits / drawbacks of thin clients

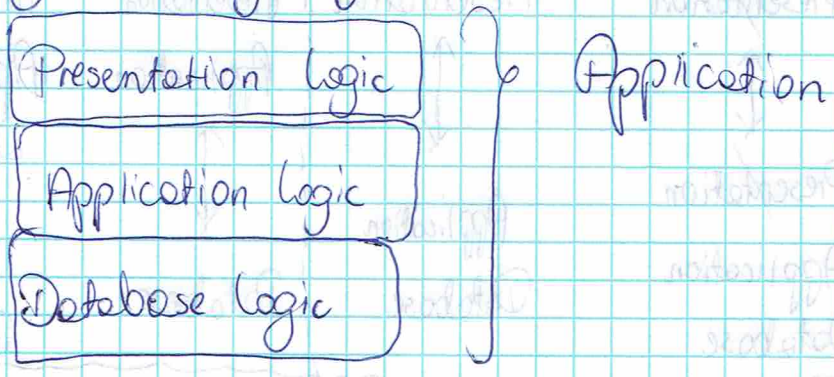
+ no viruses, spyware, thefts
+ easy to keep the software
+ lower TCO
+ fewer points of failure

- servers are the central point of failure
- systems tend to be ~~proprietary~~ proprietary
- Multimedia-rich applications require a significant amount of bandwith to function to their maximum potential

- In general application software can be divided into 3 logical tiers
  * Presentation logic
  * Application logic
  * Database logic

- Each tier in the software is responsible for a specific task in the application

- This logical layering of application software does not need to be the same as its physical layering



- Presentation logic is responsible for displaying the information and interfacing with the user.
- Application logic processes commands, makes logical decisions, performs calculations, and coordinates the application
- Database logic refers to the management of underlying databases
- Physical Tiers
  - * 1-tier architecture is used to describe systems in which all of the processing is done on a single post.
  - * 2-tier architecture (aka Net) is used to describe client/server systems, where clients requests resources and servers respond directly to these requests, using there own resources