**Conceptual Question Scenario Question Multiple choice Question**

**Client/Server**: A client connected through a server and the server provides service to the client

**Peer-to-peer**: Hosts are directly connected

Group of connected hosts → Single Network Group of single network → Internet

**Pointer**:

* Stored memory as 0 and 1
* Store a variable in consecutive addresses if variable contains more than 8 bytes of information
* Recast variable if you not sure about receiving the type

**Network Layers:**

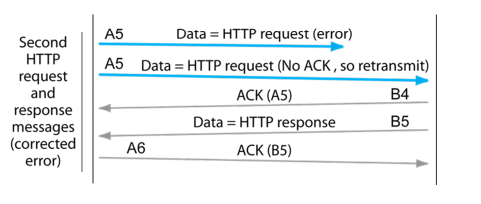
* **Application**:
* **Transport**:
* **Networking** (**Routing**):
* **Datalink** (or **Physical**): specify sources and destination MAC address, automatically find a switch to optimized routes going
* The process of **Data Transfer**: **Application** to **Transport** and **Networking** then **Datalink** and when it gets to the destination, the path is opposite

**Networking Devices**:

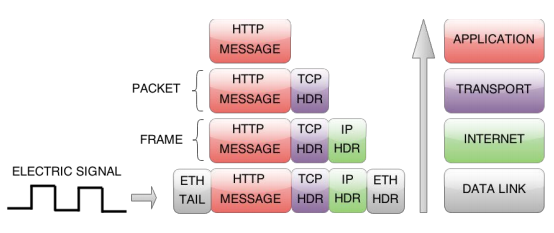
* **Hub** (Not good way): only interact with Physical. Single point of connection between hosts
* Examples: only Host A connected to Hub and transfer data to Host B & C
* **Switch** (Normal method): interact with Physical and Datalink. Allow multiple hosts to connect
* Examples: Host A, B, C, D connected to a Hub and transfer data to each other
* **Router** (Best choice): interact with Physical, Datalink and Networking. Connect multiple networks
* Examples: Multiple switches connected
* **PC**: interact with Physical, Datalink, Networking and Application

**TCP/IP Protocol**

* **Application**: Specify the format of data to be transmitted (Browser, Email)
* **Transport**: Link app with different hosts
* **TCP**: ensure data has been received, perform handshakes and acknowledgements but limited in the amount → Reliable

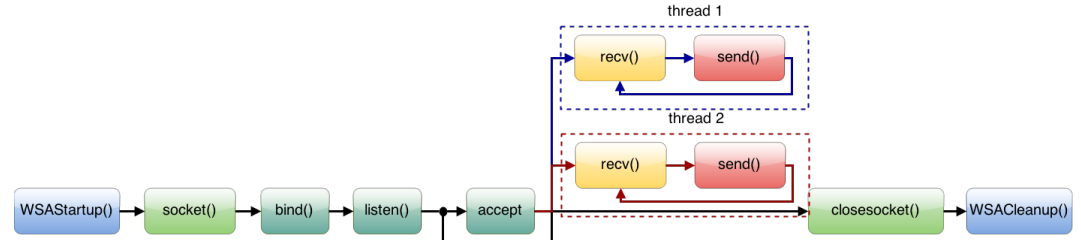


* **UDP**: no limited in amount, but lack of accuracy → Unreliable
* **Network**: Link Hosts to Networks
* **Datalink**: Link physically connected hosts

****

**Threads**: sequences of program instruction, exist with a process (A Program that is in running), run concurrently and shares same address space, each thread usually performing the same algorithm. It requires a synchronization point to ensure all child thread has finished its task before moving to the next task

* **Software**: associated with a specific task
* **Hardware**: executes software threads
* **Potential Errors**: since threads share the same address, high chance of threads trying to update memory value at the same time leads to a different result for each running time
* Threads usually use when having more than one socket between client and server to do multiple tasking

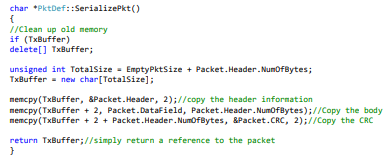


**send():** 1st: transfer the file to specified socket/port, 2nd: start at the address determined, 3rd: transmits the number of bytes data to be transferred, 4th: any possible flag while transferring the data

* example: send(ClientSocket, TxBuffer, sizeof(TxBuffer), 0);
* example2: send(ClientSocket, (char \*)&Pkt, sizeof(Pkt), 0); where (char\*) is to convert the type

**recv()**: receive the incoming data/information from send(). The holder must have from enough to bigger space to hold the message or it will be lost

* example: recv(ServerSocket, RxBuffer, sizeof(RxBuffer), 0);



**CAN Bus**:

* **CRC**: Use to check if the file has successfully arrived at the destination without any missing files or problem