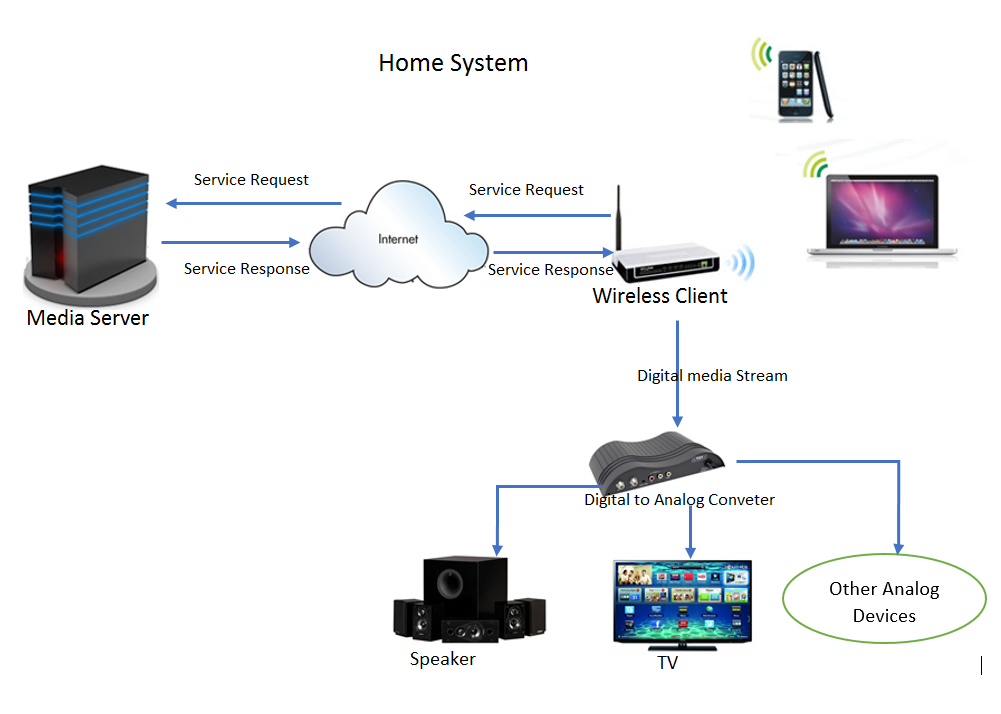


NAME: MOHD SADIQUE

CWID: A20380442

**Chapter 1**

1. **Q:** Sketch a design for a home system consisting of a separate media server that will allow for the attachment of a wireless client. The latter is connected to (analog) audio/video equipment and transforms the digital media streams to analog output. The server runs on a separate machine, possibly connected to the Internet, but has no keyboard and/or monitor connected.



1. **Q:** Describe precisely what is meant by a scalable system.

**A:** A system is scalable if it handles growing amount of work or have a potential to accommodate that growth in terms of network or process is known as scalability of a system. Scalability, as a property of systems which can grow in terms components, geo-graphical size or number and size without an unacceptable loss of performance.

Examples:-

1. A routing protocol is considered scalable with respect to network size
2. A given piece of equipment may have a capacity for 1–1000 users

1. **Q:** Scalability can be achieved by applying different techniques. What are these techniques?

**A:** Scaling can be achieved through distribution, replication, and caching

**Chapter 2**

1. **Q:** If a client and a server are placed far apart, we may see network latency dominating overall performance. How can we tackle this problem?

**A:** It depends on how the client is organized but Latency is problem of any communication. In client server architecture performance of clients depend greatly on the latency. Latency can be tackled by following methods: -

1. Instead on large request-replies client side messages or codes it could be broken into smaller parts or data. So that a when small amount of data is received the client can start work instead of waiting for a bigger chunk of data. In the same time next piece of code can be scheduled for work.
2. Clients can run multiple sessions with the server and data obtained can be added up at client end.
3. The client can utilize the delay time between sending of request and receiving reply for other processes.
4. A client and a server effectively replaces the synchronous client-server communication with asynchronous one-way communication.
5. **Q:** Consider a chain of processes P1,P2,…,Pn implementing a multitiered client-server architecture. Process Pi is client of process P(i+1), and Pi will return a replay to P(i-1) only after receiving a reply from Pi+1. What are the main problems with this organization when taking a look at the request-reply performance at process P1?

**A:** Performance for large number of processs which implementing a multitiered client-server architecture is low or shows bad performance for large n. Following are the problems with this organization :-

* The performance between *P*1 and *P*2 may also be determined by *n* -2 request-reply interactions between the other layers. The problem is that each communication between two successive layers between two different machines.
* Another problem is that if one machine in the chain performs badly or is even temporarily unreachable, then this will immediately degrade the performance at the highest level.

1. **Q:** Consider a BitTorrent system in which each node has an outgoing link with a bandwidth capacity *Bout* and an incoming link with bandwidth capacity *Bin*. Some of these nodes (called seeds) voluntarily offer files to be downloaded by others. What is the maximum download capacity of a BitTorrent client if we assume that it can contact at most one seed at a time?

**A:**. For BitTorrent system in which each node has an outgoing link with bandwidth capacity Bout, let us assume that there are S seeders and N clients, and that each client randomly picks one of the seeders. So total outgoing capacity of the seeders is S × Bout and each of the clients (S × Bout) /N immediate download capacity. In addition, if clients help each other, each one of them will be able to download chunks at a rate of Bout, by assuming that Bin >Bout . Then the maximum download capacity of a BitTorrent client is mainly dictated by its outgoing capacity. Therefore, the maximum download capacity will be

**(S × Bout )/N + Bout .**

**Chapter 4**

1. **Q:** In many layered protocols, each layer has its own header. Surely it would be more efficient to have a single header at the front of each message with all the control in it than all these separate headers. Why is this not done?

**Ans .** Each layer header contains information of that layer so it must be independent of the other ones. The data passed from layer n+1 down to layer n contains both header and data, but layer n cannot tell which is which. Having a single big header that all the layers could read and write would destroy this transparency and make changes in the protocol of one layer visible to other layers. Suppose there are network layer and transport layer and both layer have different functionality to distinguish there purpose distinguished by their header.

1. **Q:** Consider a procedure *incr* with two integer parameters. The procedure adds one to each parameter. Now suppose that it is called with the same variable twice, for example, as *incr*(*i*, *i*). If *i* is initially 0, what value will it have afterward if call-by-reference is used? How about if copy/restore is used?

**A:** If call by reference is used, a pointer to is passed to incr variable i. It will be incremented two times, so the ﬁnal result will be 2. However, with copy/restore, I will be passed by value twice, each value initially 0. Both will be incremented, so both will be 1. Now both will be copied back, with the second copy overwriting the ﬁrst one. The ﬁnal value will be 1, not 2

So in Call-by-reference variable value is 2 and in copy/restore variable value is 1

1. **Q:** Describe how connectionless communication between a client and a server proceeds when using sockets.

**A:** A network socket is an internal endpoint for sending or receiving data at a single node in a computer network. Both the client and the server create a socket, but only the server binds the socket to a local endpoint. The server can then subsequently do a blocking read call in which it waits for incoming data from any client. Likewise, after creating the socket, the client simply does a blocking call to write data to the server. Sockets are *local*, they are local resources and cannot be referred to directly by other nodes, unlike ports. There is no need to close a connection.

1. **Q:** What trade-off should be made when we decide between a shared memory model and a message passing model? Why does this make shared memory a bad match for a system distributed across the Internet?

**A:** Between shared memory model and a message passing model is all about personal choice and what is available. There is no "best" model, although there certainly are better implementations of some models over others.

In the shared-memory programming model, tasks share a common address space, which they read and write asynchronously. Various mechanisms such as locks / semaphores may be used to control access to the shared memory.

**Advantage**:-

* There is no need to specify explicitly the communication of data between tasks.
* All processes see and have equal access to shared memory
* Global address space provides a user-friendly programming perspective to memory
* Data sharing between tasks is both fast and uniform

**Disadvantage**:-

* The main disadvantage in terms of performance is that it becomes more difficult to understand and manage data locality
* lack of scalability between memory and CPUs because of cache coherence.
* Its programmer responsibility for synchronization constructs that ensure "correct" access of shared memory

In Message Passing Model tasks exchange data through communications by sending and receiving messages and Data transfer usually requires cooperative operations to be performed by each process.

**Advantages:**

* Memory is scalable with the number of processors. Increase the number of processors and the size of memory increases proportionately.
* Each processor can rapidly access its own memory without interference and without the overhead.
* Cost effectiveness: can use commodity, off-the-shelf processors and networking.

**Disadvantages:**

* The programmer is responsible for many of the details associated with data communication between processors.
* It may be difficult to map existing data structures, based on global memory, to this memory organization.
* Non-uniform memory access times - data residing on a remote node takes longer to access than node local data

**A:**  Distributed system across the internet requires communication network to connect inter-processor memory. All processors have their own local memory and memory addresses in one processor do not map to another processor over internet, so there is no concept of global address space or shared memory across all processors so that’s why shared memory a bad match for a system distributed across the Internet.

Another Reason each processor has its own local memory, it operates independently. Changes it makes to its local memory have no effect on the memory of other processors. Hence, the concept of cache coherency does not apply to distributed systems across the internet.