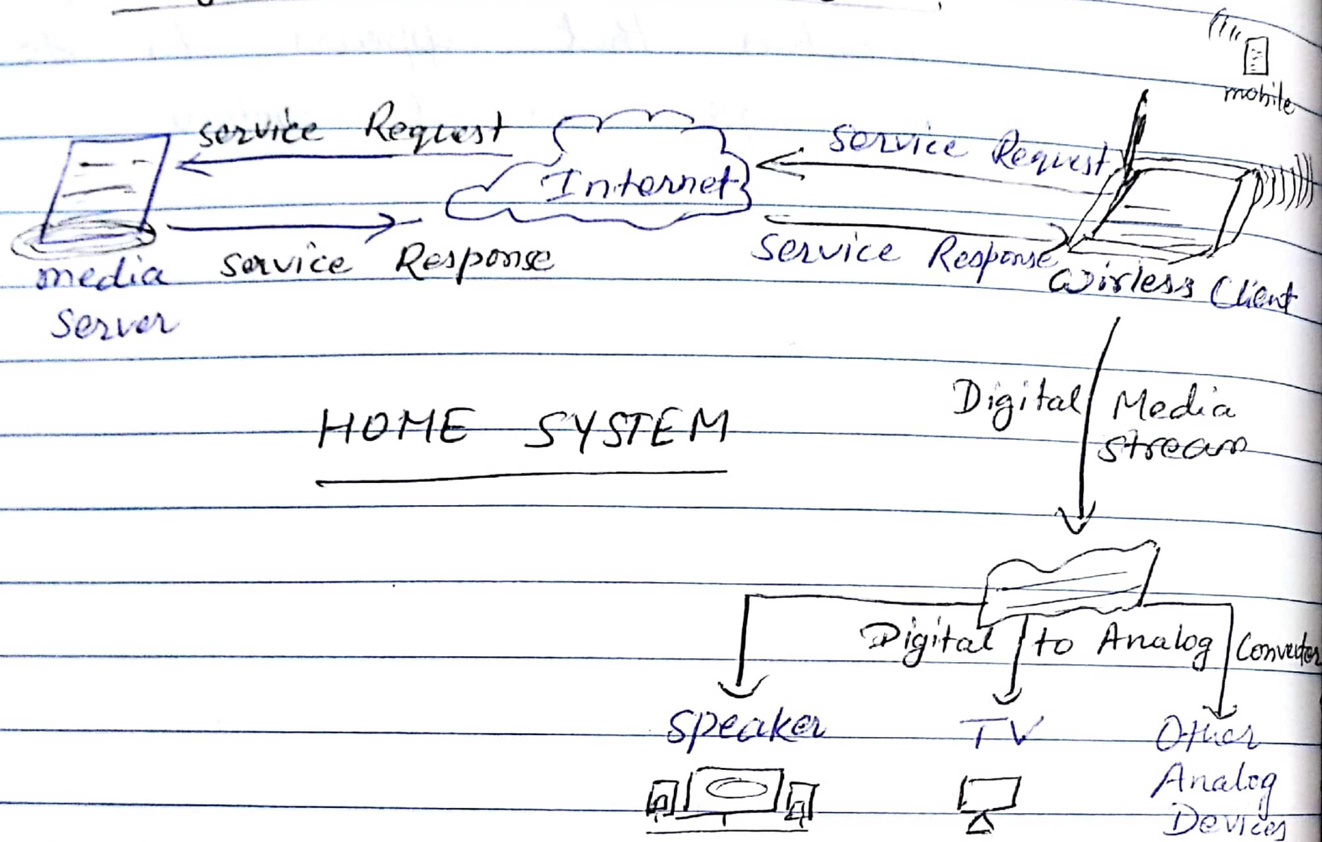
Assignment #1

Chapter : 1

Q1. Design for a home system



Q2. Describe precisely what is meant by a scalable system.

Ans: Scalable system can expand to support increasing workloads. This capability allows Computer equipment and software programs to grow over time without an unacceptable loss of performance rather than needing to be replaced. Scalability indicates the Capability of a system to increase performance under an increased load when resources (typically hardwares) are added.

Q3. Difference b/w multiprocessor and multicomputer.
Ans

Multiprocessor	MultiComputer
1. A system with two or more CPU's that allow simultaneous processing of programs is called Multiprocessor.	1. MultiComputer means it is a single processor (CPU) with multiple cores. It is a set of processors connected by the communication network that works jointly to solve computation problem.
2. It has a single ~~phy~~ physical address memory shared by all the CPU's.	2. MultiComputer have one physical address memory per CPU.
3. It run slower, because it would be in one computer.	3. MultiComputer run faster.
4. Multiprocessor size is limited.	4. It can grow to a very large numbers of processors.
5. Easier to process.	5. less easy to program
6. More difficult & costlier to build.	6. Easier & cost effective to build.
7. It supports parallel computing	7. It supports distributed computing.

Chapter 2:

Q4. If a client and a server are placed far apart, we may see network latency dominating overall performance. How can we ~~deal~~ tackle this problem?

Ans: Latency is one of the basic fundamentals of measuring network performance. Network latency means that there is delay of transmission of data. The speed of a network is measured by the time taken ~~by~~ for a data packet to be sent from one point to another. In client-server architecture, performance of clients depends ~~on~~ mainly on latency. When the latency is high, clients would experience delay and this may be improved by following methods:~

1. Instead on large request-response ~~[illegible]~~, client-side codes, it could be broken into smaller parts or data so that when small amount of data is received, client can start work instead of waiting for a bigger chunk of data. In the meanwhile, next piece of code can be scheduled for work.

2. Clients can run multiple sessions with server & data obtained can be added up at client ~~end~~ end.

3. The Client can utilize the delay time b/w sending request and receiving ~~response~~ & response for other processes.

Q5. Consider a chain of processes $P_1, P_2, ---, P_n$ implementing a multitiered client-server architecture. Process P_i is client of process $P(i+1)$ and P_i will ~~return~~ return a reply to $P(i-1)$ only after receiving a reply from P_{i+1}. What are the main problems with this organisation when taking a look at the request-reply ~~performace~~ performance at process P1?

Ans. Performance for large number of process which implementing client-server architecture is low or shows bad performance for large n.

Below are the problems with this organisation:

1. The performance b/w P_1 and P_2 may also be determined by $n-2$ request-reply interactions b/w the Other layers.

2. Another proplem 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.

Q 6. Consider a bitTorrent system in which node has an outgoing link with a bandwidth capacity Bout and an incoming link with bandwidth capacity Bin. Some of these nodes (called seeders) 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?

Ans: We need to consider that the outgoing capacity of the seeding nodes required to be shared b/w the clients.

let's assume that Total no. of Seeders = S

Total no. of Clients = N where each client randomly picks one of the seeders.

∴ The joint outgoing capacity of the seeders = S * Bout

∴ Immediate download capacity of each client = S * Bout / N

If the Clients help each other, each of them will be able to download the chunk at the rate of B_{out} considering that $B_{in} > B_{out}$

∴ Total download capacity will be:

$$((S * B_{out})/N) + B_{out}.$$

Chapter 4

Q7. 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 seperate headers. Why is this not done?

Ans:- Every layer must be independent from the other layers. The ~~data~~ message passed from layer l+1 down to layer l contains both the header and the data, but layer l can not inform that either its header or its data. But if we will think to have a single big header at the front of each message, it will destroy the transparency and make changes in the protocol of one layer visible to other layers. For example, there are network layer & transport layer and ~~both~~ both layers have different functionality to distinguish their purpose distinguished by their headers.

Q8. 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?

Ans. If call by reference is used, a pointer to i is passed to incr and i will be incremented two times, so the final output will be 2.

However, if copy/restore is used, i will be passed by value twice, each value initially 0. Both will be incremented, so both will become 1. Now both will be copied back with the second copy overwriting the first one. Therefore the final result will be 1.

Q9. One way to handle parameter conversion in RPC system is to have each machine send parameters in its native representation, with the other one doing the translation, if need be. The native system could be indicated by a code in the first byte. However, since locating the first byte in the first word is precisely the problem, can this actually work?

Ans- When a computer sends byte, it always receives same byte. For example, if a computer sends byte 0, it always arrives in bytes 0. So, the ~~destin~~ receiver computer can simply access byte 0 and the code will be in it. The order of byte does not matter if it is low or high. The alternative idea is to put the code in all the bytes of the first word. In this case the code will be there, no matter which byte is examined.

Q10. What trade-off should be made when we decide b/w 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?

Ans:- Between shared memory model and a message passing model is all about personal choice and which one is available. There is no "best" model, although there are better implementations of some models over there.

	Shared Memory	Message Passing
Interface to Communication	Communication b/w CPU's is implicit & transparent. Processors access memory through shared bus.	Processors must explicitly communicate with each other through messages.
Complexity	Supports conventional architecture better since existing processors can be added to the shared bus system easily	Since there are fewer assumptions on the model, it leads to a simpler multiprocessing architecture
Overall		The requires code to be rewritten for new platforms due to explicit interface to communication
Convenience	Serial code runs without modification	Message passing libraries ex: mpi, pvm, ... are available for a wide variety of platforms.

	Shared Memory	Message Passing
Protocols	Processors do not explicitly communicate with each other, so communication protocols are hidden within the system.	Communication protocols are fully under user ~~pro~~ control.
	Since communication occurs as part of the memory system, a smart shared memory ~~ast~~ architecture can make communication faster by taking advantage of the memory hierarchy	These protocols are complex to the programmer ~~cau~~ causing communication to be treated as an I/O call for portability reasons. This can be expensive and slow.

Distributed systems 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~~ to its local memory have no effect on the memory of other processors.

Hence, the concept of cache ~~coh~~ coherency does not apply to distributed systems across the internet.