Swami Keshvanand Institute of Technology, Management &Gramothan,
Ramnagaria, Jagatpura, Jaipur-302017

Question Paper Solution

Branch : CS (I & II Shift) **Semester: VIII** **Subject: DS(8CS3A)** **Mid Term: I**
Submitted By: Ms. Ankita Tripathi

Q.1 What is Distributed System? Explain its Characteristics, Examples & Challenges.
[5 Marks]

Ans:- A distributed system is a model in which components located on networked computers communicate and coordinate their actions by passing messages. The components interact with each other in order to achieve a common goal. A distributed system is a collection of independent computers that appears to its users as a single coherent system or as a single system.

Characteristics of distributed systems are: concurrency of components, lack of a global clock, and independent failure of components.

Examples of Distributed Systems:

(i) The Internet: net of nets global access to "everybody" (data, service, other actor; open ended). enormous size (open ended).

- no single authority
- communication types - interrogation, announcement,
- stream - data, audio, video

(ii) Intranets:

- a single authority
- protected access - a firewall
  - total isolation
- may be worldwide
- typical services: - infrastructure services: file service, name service
  -application services

(iii) Mobile and ubiquitous computing:

- Portable devices –laptops, handheld devices, wearable devices , devices embedded in appliances Mobile computing
- Location-aware computing
- Ubiquitous computing, pervasive computing

The challenges in distributed systems are:

- Heterogeneity
- Transparency
- Openness
- Concurrency
- Security
- Scalability
- Resilience to failure

Heterogeneity

This term means the diversity of the distributed systems in terms of hardware, software, platform, etc. Modern distributed systems will likely span different:

- Hardware devices: computers, tablets, mobile phones, embedded devices, etc.
- Operating System: Ms Windows, Linux, Mac, Unix, etc.
- Network: Local network, the Internet, wireless network, satellite links, etc.
- Programming languages: Java, C/C++, Python, PHP, etc.
- Different roles of software developers, designers, system managers

Transparency

Distributed systems designers must hide the complexity of the systems as much as they can. Adding abstraction layer is particularly useful in distributed systems. While users hit search in google.com, they never notice that their query goes through a complex process before google shows them a result. Some terms of transparency in distributed systems are:

Transparency	Description
Access	Hide differences in data representation and how a resource is accessed
Location	Hide where a resource is located
Migration	Hide that a resource may move to another location
Relocation	Hide that a resource may be moved to another location while in use
Replication	Hide that a resource may be copied in several places
Concurrency	Hide that a resource may be shared by several competitive users
Failure	Hide the failure and recovery of a resource
Persistence	Hide whether a (software) resource is in memory or a disk

Openness

If the well-defined interfaces for a system are published, it is easier for developers to add new features or replace sub-systems in the future. Example: Twitter and Facebook have API that allows developers to develop theirs own software interactively.

Concurrency

Distributed Systems usually is multi-users environment. In order to maximize concurrency, resource handling components should be anticipate as they will be accessed by competing users. Concurrency is a tricky challenges, then we must avoid the system's state from becoming unstable when users compete to view or update data.

Security

Every system must consider strong security measurement. Distributed Systems somehow deals with sensitive information; so secure mechanism must be in place.

Scalability

Distributed systems must be scalable as the number of user increases.

A system is said to be scalable if it can handle the addition of users and resources without suffering a noticeable loss of performance or increase in administrative complexity
Scalability has 3 dimensions:

- Size
- Geography
- Administration

Resilience to Failure

Distributed Systems involves a lot of collaborating components (hardware, software, communication). So there is a huge possibility of partial or total failure.

OR

Q.1 Explain the Lamport's Logical Clock Algorithm.

Ans:-Lamport's logical clocks

- the "time" concept in distributed systems -- used to order events in a distributed system.
- assumption:
  - the execution of a process is characterized by a sequence of events. An event can be the execution of one instruction or of one procedure.
  - sending a message is one event, receiving a message is one event.
- The events in a distributed system are not total chaos. Under some conditions, it is possible to ascertain the order of the events. Lamport's logical clocks try to catch this.

Lamport's "happened before" relation

The "happened before" relation (®) is defined as follows:

- *A* ® *B* if *A* and *B* are within the same process (same sequential thread of control) and *A* occurred before *B*.
- *A* ® *B* if *A* is the event of sending a message *M* in one process and *B* is the event of receiving *M* by another process
- if *A* ® *B* and *B* ® *C* then *A* ® *C*

Event *A causally affects* event *B* iff *A* ® *B*.

Distinct events *A* and *B* are *concurrent* (*A* | | *B*) if we do not have *A* ® *B* or *B* ® *A*.

Lamport Logical Clocks

- are local to each process (processor)
- do not measure real time
- only measure ``events"
- are consistent with the happened-before relation
- are useful for totally ordering transactions, by using logical clock values as timestamps

Logical Clock Conditions

C_i is the local clock for process P_i

- if a and b are two successive events in P_i, then
  $C_i(b) = C_i(a) + d_1$, where $d_1 > 0$
- if a is the sending of message m by P_i, then m is assigned timestamp $t_m = C_i(a)$
- if b is the receipt of m by P_j, then
  $C_j(b) = \max\{C_j(b), t_m + d_2\}$, where $d_2 > 0$

Logical Clock Conditions

The value of d could be 1, or it could be an approximation to the elapsed real time. For example, we could take d_1 to be the elapsed local time, and d_2 to be the estimated message transmission time. The latter solves the problem of waiting forever for a virtual time instant to pass.

Implementing Lamport's Logical Clocks

- When a message is transmitted from P1 to P2, P1 will encode the send time into the message.
- When P2 receives the message, it will record the time of receipt
- If P2 discovers that the time of receipt is before the send time, P2 will update its software clock to be one greater than the send time (1 milli second at least)
- If the time at P2 is already greater than the send time, then no action is required for P2
- With these actions the "happens-before" relationship of the message being sent and received is preserved.

Lamport's Algorithm

Lamport's algorithm is based on two implementation rules that define how each process's local clock is incremented.

Notation:

- the processes are named Pi ,
- each process has a local clock, Ci
- the clock time for an event a on process Pi is denoted by Ci (a).

Rule 1:

If a and b are two successive events in Pi and a --> b then Ci (b) = Ci (a) + d where d > 0.

Rule 2:

If a is a message send event on Pi and b is the message receive event on Pj then:

- the message is assigned the timestamp tm = Ci (a)
- Cj (b) = max (Cj , tm +d)

Q.2 What is RPC in Distributed Systems? Explain the client/server architecture.

[5 Marks]

Ans:- In distributed computing a remote procedure call (RPC) is when a computer program causes a procedure (subroutine) to execute in another address space (commonly on another computer on a shared network), which is coded as if it were a normal (local) procedure call, without the programmer explicitly coding the details for the remote interaction. That is, the programmer writes essentially the same code whether the subroutine is local to the executing program, or remote. This is a form of client–server interaction (caller is client, executer is server), typically implemented via a request–response message-passing system.

The object-oriented programming analog is remote method invocation (RMI). The RPC model implies a level of location transparency, namely that calling procedures is largely the same whether it is local or remote, but usually they are not identical, so local calls can be distinguished from remote calls. Remote calls are usually orders of magnitude slower and less reliable than local calls, so distinguishing them is important.

RPCs are a form of inter-process communication (IPC), in that different processes have different address spaces: if on the same host machine, they have distinct virtual address spaces, even though the physical address space is the same; while if they are on different hosts, the physical address space is different. Many different (often incompatible) technologies have been used to implement the concept.

RPC uses the client-server model. The requesting program is a client and the service providing program is the server. Like a regular or local procedure call, an RPC is a synchronous operation requiring the requesting program to be suspended until the results of the remote procedure are returned. However, the use of lightweight processes or threads that share the same address space allows multiple RPCs to be performed concurrently.

RPC message procedure
When program statements that use RPC framework are compiled into an executable program, a stub is included in the compiled code that acts as the representative of the remote procedure code. When the program is run and the procedure call is issued, the stub receives the request and forwards it to a client runtime program in the local computer.

The client runtime program has the knowledge of how to address the remote computer and server application and sends the message across the network that requests the remote procedure. Similarly, the server includes a runtime program and stub that interface with the remote procedure itself. Response-request protocols are returned the same way.

RPC models and alternative methods for client-server communication
There are several RPC models and distributed computing implementations. A popular model and implementation is the Open Software Foundation's Distributed Computing Environment (DCE).

Client/Server Architecture
Client/server is a program relationship in which one program (the client) requests a service or resource from another program (the server).

Although the client/server model can be used by programs within a single computer, it is a more important concept for networking. In this case, the client establishes a connection to the server over a local area network (LAN) or wide-area network (WAN), such as the Internet. Once the server has fulfilled the client's request, the connection is terminated. Your Web browser is a client program that has requested a service from a server; in fact, the service and resource the server provided is the delivery of this Webpage.

Computer transactions in which the server fulfills a request made by a client are very common and the client/server model has become one of the central ideas of network computing. Most business applications use the client/server model as does the Internet's main program, TCP/IP. For example, when you check your bank account from your computer, a client program in your computer forwards a request to a server program at the bank. That program may in turn forward a request to its own client program, which then sends a request to a database server at another bank computer. Once your account balance has been retrieved from the database, it is returned back to the bank data client, which in turn serves it back to the client in your personal computer, which then displays the information to you.

Both client programs and server programs are often part of a larger program or application. Because multiple client programs share the services of the same server program, a special server called a daemon may be activated just to await client requests. In marketing, the client/server was once used to distinguish distributed computing by personal computers (PCs) from the monolithic, centralized computing model used by main frames. This distinction has largely disappeared, however, as mainframes and their applications have also turned to the client/server model and become part of network computing.

Other program relationship models included master/slave and peer-to-peer (P2P). In the P2P model, each node in the network can function as both a server and a client. In the master/slave model, one device or process (known as the master) controls one or more other devices or processes (known as slaves). Once the master/slave relationship is established, the direction of control is always one way, from the master to the slave.

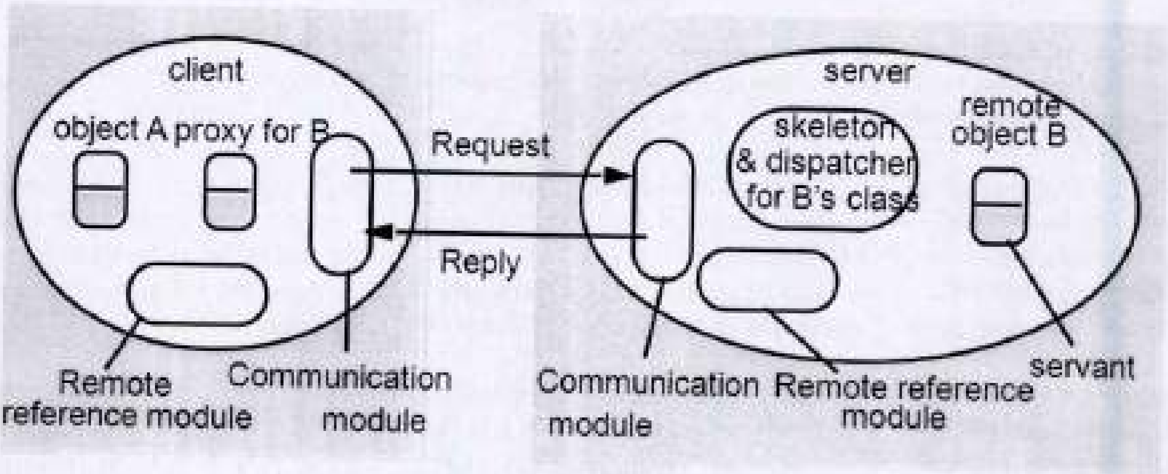
OR

Q.2 Explain the Design Issues and Implementation of RMI.
Ans:-
Design Issues for RMI

(i) **RMI Invocation Semantics:** Invocation semantics depend upon implementation of Request-Reply protocol used by RMI and Maybe, At-least-once, At-most-once invocation semantics.

(ii) **Transparency**: There Should be remote invocations transparent to the programmer. Partial failure, higher latency m Current consensus: remote invocations should be made transparent in the sense that syntax of a remote invocation is the same as the syntax of local invocation (access transparency) but programmers should be able to distinguish between remote and local objects by looking at their interfaces, e.g. in Java RMI, remote objects implement the Remote interface

RMI Implementation:



1. Communication module
2. Remote reference module
3. RMI software
4. Server initialization
5. activation of remote objects
6. persistent object stores
7. Object location
8. Distributed garbage collection

Q.3 Describe the Interfaces in the Distributed System. **[5 Marks]**
Ans:-
An interface provides a definition of the signatures of a set of methods without specifying their implementation.

(i). Service interfaces (RPC) – The specification of the procedures offered by a server. It defines the types of input and output arguments for each procedure.

(ii). Remote interfaces (RMI) – The specification of the methods of an object that are available for invocation. It defines the types of input and output arguments.

Programs organized as a set of modules that communicate with one another via procedure calls/method invocations:

- Explicit interfaces defined for each module in order to control interactions between modules
- In distributed systems, modules can be in different processes
- A remote interface specifies the methods of an object that are available for invocation by objects in other processes defining the types of the input and output arguments of each of them

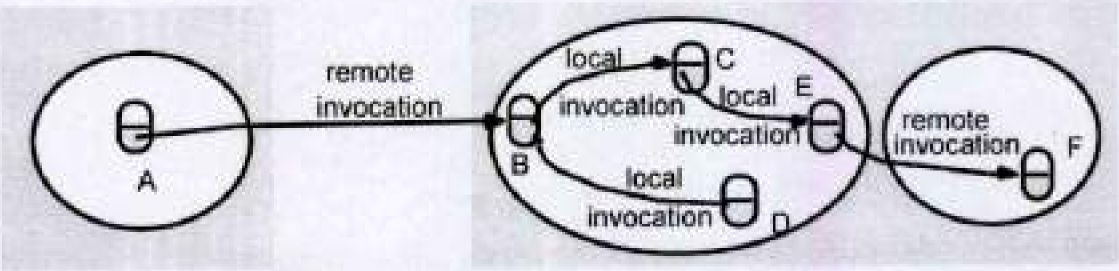


Fig: Remote and local method invocations

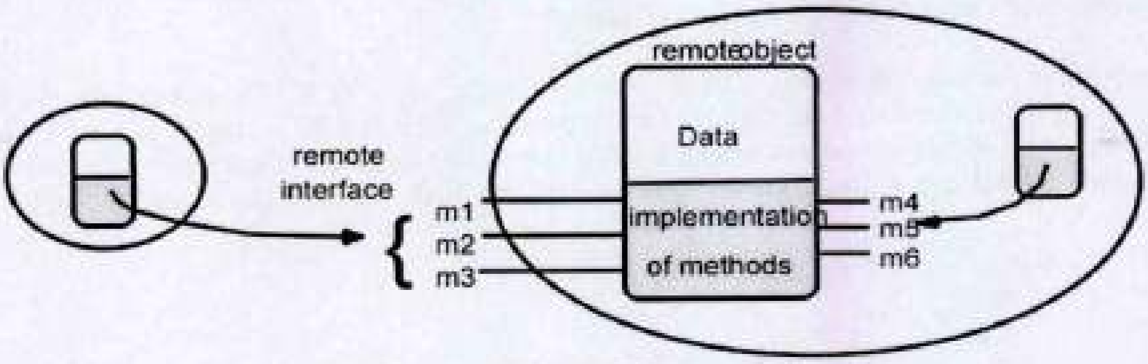


Fig: A remote object and its remote interface

OR

Q.3 Write Short Notes on the following: **[2.5x2=5 Marks]**

(a) Naming and Directory Services

Ans:-

Naming Service

• A naming system provides a naming service to its customers for performing naming-related operations. – For example, associate a domain name with an IP address.

• A namespace is the set of names in a naming system.

Directory Service

• A Directory Service is an extension of a naming service that allows one to lookup objects based on names or based on attributes.

• Attributes have attribute identifiers and a set of attribute values.

• Therefore – Directory Services have a lot in common with RDBMS
– OpenLDAP is a Directory Service, and you can plug a RDBMS in the back-end

Directory Enabled Applications

- A directory-enabled application is an application that uses a naming or directory service.
- Applications can share the common infrastructure provided by the directory. – Example: A mail client, scheduling systems and mail forwarding program might all use the same address book stored in a common directory.
- The directory may also be used as an object store for programs needing the same object.

(b) Events and States in Distributed Systems

Ans:-

Event: The occurrence of a single action that a process carries out as it executes e.g. Send, Receive, change state.

The concept of one **event** happening before another in a **distributed system** is examined, and is shown to define a partial ordering of the events. A **distributed** algorithm is given for synchronizing a **system** of logical clocks which can be used to totally order the events.

Events at a single process pi, can be placed in a total ordering denoted by the relation → between the events. i.e.
e → e' if and only if the event e occurs before e' at pi

State: The global state of a distributed system is the set of local states of each individual processes involved in the system plus the state of the communication channels.

At any point in computation there is at most one event that can happen next. The global state of a distributed system is the union of the states of the individual processes. Given that the processes of a distributed system do not share memory but instead communicate solely through the exchange of messages, a process that wishes to construct a global state must infer the remote components of that state through message exchanges.

Swami Keshvanand Institute of Technology, Management &Gramothan,
Ramnagaria, Jagatpura, Jaipur-302017

Question Paper Solution

Branch: CS Semester: VIII **Subject: Real Time Systems** **Mid Term: I**

Submitted By : Sunil Dhankhar and Priyanka Trikha

1. **What is Real time system? Explain characteristics of real time systems.**

Ans: Definition 1: RT-systems are systems in which the correctness of the system behavior depends on the logical results of the computations and on the physical time when these results are produced **Definition 2:** RT-systems are systems that have to be designed according to the dynamics of a physical process.

Characteristics of real time systems:

1. Speed: The rate of progress a job makes toward completion.

2. Predictability: Predictability is defined as the ability of the controlling system to determine a task's completion time with certainty and ensuring that all hard deadlines will be met, **taking system state** and **task's resource need** into account. The importance of this requirement of a real-time system can be demonstrated by considering the example of **an air-defense system monitoring** the sky for incoming enemy missiles. The nature of this application demands that the incoming enemy missile must be detected and destroyed within a predefined deadline without failing. ***Static Guarantee*** **:-** In small real-time systems, where characteristics of all critical tasks are known prior, guarantee of satisfying deadlines of all critical tasks can be given at design time.

Run-time **or** ***Dynamic Guarantee*** **:-**The same guarantee cannot be given for more complex real-time systems working in a dynamic environment as characteristics of all tasks are not necessarily known a priori and the necessary deadline guarantee can be given only at run time, using an on-line schedulability analysis.

3. **Reliability:** Reliability in real time systems can be defined as the ability of the system to deliver a **certain volume of computation in a given period of time**, so as to meet task deadlines. Reliability in real-time systems combines the **hardware and software** reliability with **temporal correctness**. Reliability is an essential requirement of real-time systems.

4. Adaptability: If a system is adaptive, one does not have to **redefine** the system or **recomputed resource** and task allocation for every small change. The adaptability reduces **development and maintenance costs** and allows system to expand without redefining the whole system. A real time system should be adaptive to change in system state including **overloads and failures, system configuration, input specification, and task specifications.**

2. Explain the following?

1. Period: Minimum inter release time between two jobs. This is denoted by p.

2. Release time – the instant in time when a job becomes available for execution.

- May not be exact: Release time jitter so r_i is in the interval $[r_i^-, r_i^+]$.
- A job can be scheduled and executed at any time at, or after, its release time, provided its resource dependency conditions are met.

3. Execution Time

- A job J_i will execute for time ei
    - This is the amount of time required to complete the execution of J_i when it executes alone and has all the resources it needs.
    - Execution times fall into an interval $[e_i^-, e_i^+]$; assume that we know this interval for every hard real-time job, but not necessarily the actual e_i
        - Terminology: (x, y] is an interval starting immediately after x, continuing up to and including y
    - Value of e_i depends upon complexity of the job and speed of the processor on which it is scheduled; may change for a variety of reasons:
        - Conditional branches, Cache memories and/or pipelines

- Compression (e.g. MPEG video frames)
- Often, we can validate a system using e_i^+ for each job; we assume $e_i = e_i^+$ and ignore the interval lower bound

4. **Deadline:** There are two types of deadline:

Relative deadline Di – the maximum allowable job response time.

Absolute deadline d_i– the instant of time by which a job is required to be completed (often called simply the deadline)

- Absolute deadline = release time + relative deadline

5. **Tardiness:** Lateness is completion time minus deadline; positive lateness is tardiness; negative lateness is earliness.
6. **Phase:** It denote the release time of first job.

5. There are 4 independent periodic tasks in the system:

T1= (0, 4, 1, 4)
T2= (0, 5, 1.8, 5)
T3= (0, 20, 1, 20)
T4= (1, 20, 2, 20)

Calculate the following:

1. Hyperperiod= LCM (4, 5, 20, 20)= 20
2. Total no. of Jobs in hyperperiod= 11 jobs
3. System utilization= 1/5+1.8/5+1/20+2/20= 0.2+0.36+0.05+0.1=0.71

4. Write the short note on

1. **Periodic Task:** Periodic tasks are time-driven and are invoked or activated at regular time intervals called period and have deadlines by which they must complete their execution. The deadline of a periodic task in general may be less than, equal to or greater than the period of the task. Periodic tasks generally ensure the supervision of the controlled system and typically have strict timing constraints dictated by the physical characteristics of the environment. Some of the periodic tasks may exist from the point of system initialization, while other may come into existence dynamically. Deadline of the periodic tasks must be met regardless of the other conditions in the system, failing which the system may lead to serious accidents including loss of human lives.
2. **Aperiodic Task:** Aperiodic tasks are event driven and invoked only when certain events occur. The aperiodic tasks have irregular arrival times and are characterized by their ready time, execution time, and deadline. These values are known only when the task arrives. Most of the dynamic processing requirement in real-time applications is aperiodic. For example, aperiodic tasks are activated to treat errors, faults, alarms and all situations indicating the occurrence of abnormal events in the controlled system or dynamic events such as object falling in front of a moving robot or a human operator pushing a button on a console.
3. **Sporadic Task:** Sporadic tasks are special aperiodic tasks with known minimum interarrival times and having hard deadlines. These are also activated in response to some external events that occur at arbitrary point of time, but with defined maximum frequency. Each sporadic task will be invoked repeatedly with a (non-zero) lower bound on the duration between consecutive occurrences. For example, a sporadic task with a minimum interarrival time of 100 ms will not invoke its instances more often than each 100 ms. In fact, time between two consecutive invocations may be much larger than 100 ms.

Swami Keshvanand Institute of Technology, Management &Gramothan,
Ramnagaria, Jagatpura, Jaipur-302017

Question Paper Solution

Branch CSE/ IT **Semester: VIII** **Subject: Mobile Computing** **Mid Term: I**
Submitted By: Ankit Kumar, Vipin jain , Anjana Sagwan

Q1. What is mobile computing? Explain adaptability issues in detail.

Ans: - Mobile computing is a generic term used to refer to a variety of devices that allow people to access data and information from where ever they are. Mobile computing is about providing information anytime anywhere or, more generally, computing anytime and anywhere. Mobile computing is also about dealing with limitations of mobile computing devices. For example, *personal digital assistants* (PDAs and laptops have small interfaces and are powered by batteries. One of the major issues is how to do computation in an energy-efficient manner. Battery technology is not advancing at the same pace as processor technology, so it is not expected that battery capacity will double at a fixed rate, as is true about processor speeds, according to Moore's law. One does not have to deal with these issues when designing systems for stand-alone or distributed systems. One may have to deal with issues of fault tolerance in distributed systems, such as server crashes or network link failures. However, energy usually is not an issue. In mobile systems, energy becomes a resource like processing time or memory space. Therefore, one now has to design resource management techniques for energy, just as traditional operating systems deal with process and memory management.

1) Transparency

Transparency is the ability of a system to hide some characteristics of its underlying implementation from users. Much of the research effort in distributed computing has been devoted to developing mechanisms for providing various forms of transparency. Examples of these include the following:

_ ***Access transparency*** is the ability of a system to hide the differences in data representation on various machines and the mode of access of a particular resource.

_ ***Location transparency*** is the ability of a system to conceal the location of a resource. Related to location transparency are *name transparency user mobility* (which ensures that no matter which machine a user is logged onto, she should be able to access resources with the same name).

_ ***Failure transparency*** is the ability of the system to hide failure and recovery of a system component. Mobile computing systems can be viewed as a form of distributed system, and attempts can be made to provide "mobility transparency," which would encompass the transparencies just mentioned. This would, in essence, support application-transparent adaptation. But is this an achievable, or even desirable, goal for building mobile computing systems and applications? Let us look closely at the characteristics of the mobile computing environment and their implications in these regards.

2) Constraints of mobile computing Environments

Mobile computing has many constraints that distinguish a *mobile computing environment* (MCE) from the traditional desktop workstation/ PC–based distributed computing environment. Notable among these are the following

Question Paper Solution

Branch CSE/ IT **Semester: VIII** **Subject: Mobile Computing** **Mid Term: I**

Submitted By: Ankit Kumar, Vipin jain , Anjana Sagwan

1. ***Mobile computers can be expected to be more resource-poor than their static counterparts.***

With the continued rapid improvement of hardware technology, in accordance with Moore's law, it is almost certain that a laptop computer purchased today is more powerful that the desktop computer purchased just a year or even a few months ago. However, mobile computers require a source of electrical energy, which is usually provided by battery packs. Since batteries store a finite amount of energy, they need to be replaced or recharged. The first option costs money, and the second option, although cheaper in terms of money expended, requires plugging in the computer for recharging, restricting mobility. This has an impact on the design of mobile computers—all the hardware and software components in mobile computers are designed to reduce energy consumption and to increase the lifetime of the batteries. For example, processors on mobile computers are designed to consume less energy and, consequently, achieve a lower computation performance.

2. ***Mobile computers are less secure and reliable.***

Since mobile computers accompany their users everywhere, they are much more likely to be lost Or stolen. Furthermore, they are more likely to be subjected to hostile or unfriendly use, e.g., a child throwing his daddy's PDA in a tantrum.

3. ***Mobile connectivity can be highly variable in terms of its performance (bandwidth and latency) and reliability.***

Disconnections, both voluntary and involuntary, are common. The link bandwidth can vary by orders of magnitude over time and space. Thus, in general, resource availability and quality vary dynamically. The preceding characteristics of a mobile computing environment require rethinking about how mobile applications and systems should be designed. Resource paucity and the lower reliability of mobile devices point toward designing systems in such a manner that more reliance is placed on the static infrastructure. On the other hand, the possibility Of disconnections and poor connectivity point toward making systems less reliant on the static infrastructure. Further, as mobile devices are moved around (or even if they are not), their situations keep changing over time. Mobile devices should change their behavior to be either more or less reliant on static infrastructure depending on current conditions.

3 Application-aware adaptation

Who should be responsible for adaptation—the application, the system, or both? There are two extreme approaches to designing adaptive systems: application-transparent (the system is fully responsible for adaptation) and laissez-faire (the system provides no support at all) obviously, the laissez-faire approach is not desirable because it puts too much burden on the application developer. Further, no support from the underlying system restricts the types of adaptations that can be performed. However, as the following example points out, the application-transparent approach is not sufficient either. Consider two different multimedia applications. In one you are videoconferencing using a mobile device, and in the other you are watching live video stream from a remote server on your mobile device. Now consider the following scenarios. In one, you move from an area with sufficient bandwidth for your application to an area where the amount of bandwidth is less than that needed by your application. In the other, your laptop's battery power level drops considerably. Both scenarios deal with changes in availability of resources. How would you like your system/application to behave under each scenario? In the application (user)–transparent approach, the system/application may behave the same irrespective of the application running. However, different responses may be suitable depending on the type of Application that is running. For example, in the first scenario, a nonadoptive system may just do nothing and let the audio/video quality drop. In the second scenario, the system may just give a warning to the user without any assistance on how to deal with the situation. In an adaptive system, various behaviors can be envisioned. For example, the system may try to do its best in both situations. However, the system's adaptation does not take into account the kind of application that is running. For example, in the first scenario, the system may try to adapt by

Requesting that the server or other peers start to send lower-quality video, in effect requiring lower bandwidth. In the second scenario, the system may try to conserve energy by reducing the intensity of the backlight of the display (besides warning the user of the lower battery power level). A still more adaptive approach is possible in which the system interacts with the user/application in deciding how to adapt. In the application-transparent approach, the responsibility of adaptation lies solely with the underlying system. On the other hand, in the Application-aware approach, the application collaborates with the underlying system software. Here the system provides status information about the available resources. The application uses this information to make decisions on how to adapt to changes in the resource availability.

Each application can adapt in its own way

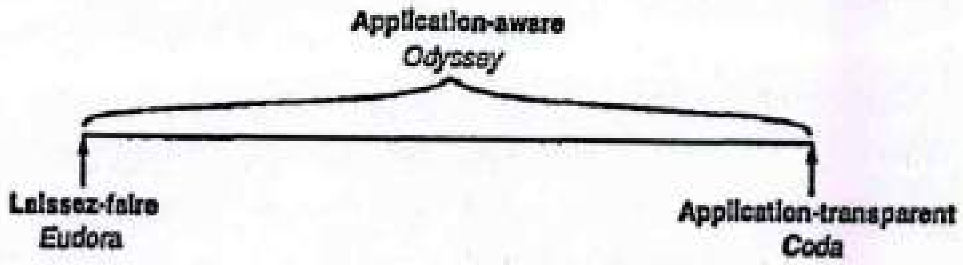


Figure 1: Models of Adaptation

OR

Q1. What is mobility management? Briefly explain PCS location management scheme.

Ans- Mobility management is a functionality that facilitates mobile device operations in Universal Mobile Telecommunications System (UMTS) or Global System for Mobile Communications (GSM) networks. Mobility management is used to trace physical user and subscriber locations to provide mobile phone services, like calls and Short Message Service (SMS).

PCS location management scheme:-

Two types of location registrars are used: home location registrars (HLRs) and visitor location registrars (VLRs). The HLR keeps the location and profile information for all the mobile nodes to which the PCS network is supposed to provide service.Each RA has a VLR associated with it that records the location (cell ID) of all the mobile nodes that currently are in that RA. The HLR for a mobile node records the RA, which is the ID of the VLR associated with the RA in which the mobile node currently is located. When a call needs to be established to mobile node m that is currently located in cell c of RAc, first the HLR of mobile node m [HLR (m)] is consulted to obtain the ID of the VLR [VLR (n)] that may have information about m; next, VLR (m) is contacted to obtain the current cell in which mobile node m is located. When a mobile node is switched on, it registers with one of the available access points (base stations). This registration operation also involves updating the VLR and the HLR. While the mobile node remains active, a reregistration is performed (1) when a handoff occurs and (2)

Periodically. When a mobile node m moves from cell c to cell d, the following two scenarios are possible:

_ Both cells c and d belong to the same RA. In this case, only the VLR is updated to indicate in which cell mobile node mis currently located. This helps when mobile node m needs to be located. In this case, there is no need to contact the HLR (m).

_ Cells c and d belong to different registration areas, RAc and RA d , respectively. In this case, the following two actions need to be taken:

_ Mobile mode m needs to register with RAd and deregister with RAc. HLR (m) needs to be notified that mobile node m is now in RAd.

Now let's examine in more detail what actions need to be performed when mobile node mneeds to be located, for example, to establish a connection between mobile nodes n and m. Assume that mobile node n is in cell c and that mobile node m is in cell d, where cell c belongs to RAc and cell d belongs to RAd. _ First, VLR(RAd) is consulted to see whether mobile node m is in RAd. If so, a search is performed in the vicinity of last reported cell of mobile node m. If m is not RAd, then the HLR (m) is contacted to get the current RAm. VLR (RAm) is contacted and performs a local search in the last reported cell of mobile node m, and if successful, it returns the current location of mobile node m.

Q2. Explain mechanisms for adaptation and incorporating adaptation.

Adaptation in mobile computing means the ability an application or algorithm has to output different valid results, depending on the characteristics of the environment where the mobile device is located. This is not common in fixed systems, where the conditions are practically stable

It helps applications to deal intelligently with limited or fluctuating resource levels.

– Adapt behavior and expectations to conserve scare resources

– Adjust quality of service (QoS) – guarantee performance

• Mobile audio application - It provides lower quality audio in case of disruption in bandwidth instead of stop delivering any audio.

• Mobile Video application – It provides lower quality video in case of disruption in bandwidth.

• Mobile Video game application – When the battery level is very low during the running of gaming application, it adjusts to decreased battery level by modifying resolution to conserve power.

Types of adaptation

• Application-aware strategies:

• OS aware strategies:

The different types of devices, network connection, and execution context of mobile systems when compared to fixed ones insert many constraints in the mobile environment not present in the fixed one. Due to those constraints, when defining an adaptation architecture in mobile computing, various

Aspects should be observed: data, security, and quality of service, available resources, costs, performance, and broadcast and multicast issues.

It is also important to consider whether or not the adaptation architecture is addressed to a specific application. Architectures addressed to specific applications often achieve better results, since the target application is known, and the designer may focus on its main characteristics and propose the best adaptation techniques for that application. However, those adaptation architectures can only be used for that target application. Other applications running upon this architecture will not benefit from it.

Generic architectures, on the other hand, do not address a target application. Instead, some general adaptation techniques are implemented. When an application is running upon a generic architecture, it will benefit only from the adaptation techniques that can be applied to it. Since these techniques were not designed specifically for this application, they often will not achieve the best possible adaptation level. However, they will be able to achieve good adaptation levels with many applications.

OR

Q2 .Explain data dissemination and its models. What are the challenges faced in data dissemination in mobile environment.

Ans: - Data dissemination on the internet is possible through many different kinds of communications protocols. The internet protocols are the most popular non-proprietary open system protocol suite in the world today. They are used in data dissemination through various communication infrastructures across any set of interconnected networks. Despite the name internet protocol, they are also well suited for local area networks (LAN) and wide area network (WAN) communication.

Using the internet, there are several ways data can be disseminated. The World Wide Web is an interlinked system where documents, images and other multimedia content can be accessed via the internet using web browsers. It uses a markup language called hypertext markup language (HMTL) to format disparate data into the web browser. The Email (electronic mail) is also one of the most widely used systems for data dissemination using the internet and electronic medium to store and forward messages. The email is based on the Simple Mail Transfer Protocol (SMTP) and can also be used by companies within an intranet system so that staff could communicate with other.

The more traditional ways for data dissemination which are still in wide use today are the telephone systems which include fax systems as well. They provide fast and efficient ways to communicate in real time. Some telephone systems have been simulated in internet applications by using the voice over internet protocol (VoIP). Through this protocol, hundreds of free or minimally charge international phone calls are already available. This simulated phone call is possible using the computer with microphone and speaker system or headphones. When a video camera is used, it could be possible to have video conferencing.

Challenges:-

Architecture-Based Cellular Mobile Networks

1. Weak Connectivity
2. Severe Resource Constraints
3. Asymmetric Communication Links
4. Location and Time (context) Dependent

Architecture-less Mobile Ad Hoc Network (MANET)

1. Weak Connectivity
2. Severe Resource Constraints

Q3. Explain broadcast disk scheduling in detail with example.

Ans: - Broadcast disk scheduling

Broadcast scheduling deals with determining how often to publish a certain data item – once the decision regarding which data items to publish has been made. A novel way to view a broadcast channel is to regard it as an extension to the memory hierarchy of the mobile node—a sort of memory disk in the air. The broadcast channel itself can be structured as multiple virtual disks, each spinning at different rates. The hottest set of data items is allocated to the fastest-spinning disk, the next hottest data item to next-fastest disk, and so on. There are nine data items, with item *D*1 assigned to disk 1, items *D*2 through *D*5 assigned to disk 2, and items *D*6 through *D*9 assigned to disk 3. As can be seen from the broadcast schedule, the items on disk 1 appear four times as frequently as those on disk 3, and the items on disk 2 appear two times as frequently as those on disk 3. Ineffect, disk 1 is spinning fastest, followed by disk 2 and then disk 3.

Frequency of Broadcast

Items on Disk 1 appear 4-times as frequently as those on disk 3

Items on Disk 2 appear 2-times as frequently as those on disk 3

Speed of Spinning

Disk 1 – Fastest

Disk 2 - Middle

Disk 3 - Slowest

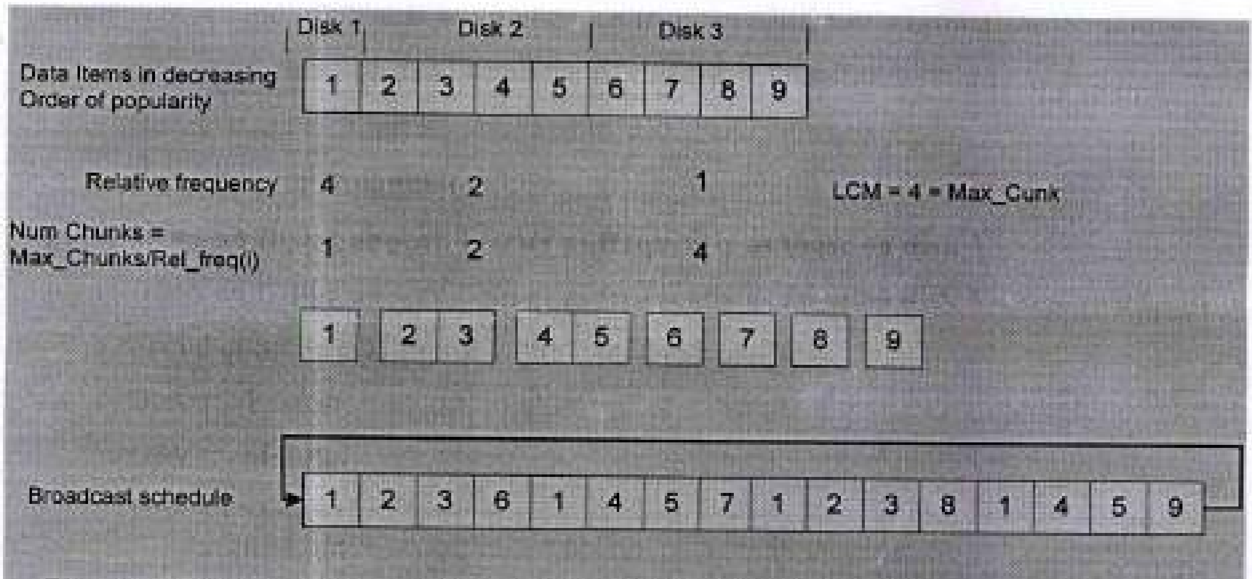


Figure 3.5 Example derivation of broadcast schedule for broadcast disks
Adapted from the book "Fundamentals of Mobile and Pervasive Computing, 2005, by Frank Adelstein, et. al, McGraw-Hill

- Developed by Acharaya, Alonso, Franklin, and Zodnik
- Scheduling of Broadcast Channel that achieves a selected relative spinning of the disk
- Inputs
    - Number of disks
    - The assignment of data items to the disk
    - Relative spinning frequencies of the disks
- Output
    - Broadcast schedule
    - Rel_freq(disk_1) = 4
    - Rel_freq(disk_2) = 2
    - Rel_freq(disk_3) = 1
- Order the data items from the hottest to coldest
- Partition the list into multiple ranges, called disks. Each disk consists of data items which nearly same popularity ratio. Let the number of disks chosen be num_disks.
- Choose the relative frequency of broadcast for each disk
- Cluster the items in each disk into smaller units called chunks;

    number_chunk(i) = max_chunks/rel_freq(i),

    - where max_chunks is the least common multiple of relative frequencies
- Create broadcast schedule as follows: (see references)

```
• For i = 0 to max_chunks -1
    ■ For j = 1 to num_disks
        ■ K = i mod num_chunks(j)
        ■ Broadcast chunk Cj,k
    ■ End_for
• End_for
```

OR

Q3. What is mobile data caching? Explain cache information maintenance schemes in detail.

Ans: - Mobile Data caching is an appropriate technique for reducing wireless data transmissions in mobile information systems. The literature describes numerous caching approaches on a theoretical level. Semantic, preemptive, or context aware caches are discussed but not implemented for mobile devices even evaluated in real applications. The problem here is the complexity of data management tools.

A cache Information maintenance schemes

Before going into details of some important mobile cache maintenance scheme, let us first understand the main characteristics of these schemes through the following classification.

As we saw earlier, there are two different cache consistency requirements: *strong cache consistency* and *weak cache consistency*. Strong cache consistency specifies that the cached data always be up-to-date. Polling every time and invalidating data on modification are two approaches to achieve strong cache consistency. On the other hand, weak cache consistency allows some degree of data inconsistency. TTL based consistency strategies are used when it is sufficient to guarantee weak consistency for a data item. In general, there are three basic strategies for maintaining cache consistency: polling every time, TTL-based, and invalidation-based with *polling-every-time* and *TTL-based* caching strategies, the client initiates the consistency verification; i.e., the client is responsible for verifying the data consistency before using them. In a TTL based caching strategy, every cached data item is assigned a TTL value, which can be estimated based on the data item's update history. For example, the adaptive TTL approach in Cate (1992) estimates TTL based on the age of a data item. When the user request arrives for a data item *x*, if data item *x*'s residence time has exceeded its TTL value, the client sends a message to the server to ask if *x* has changed. Based on the server's response, the client may get a new copy of *x* from the server (if data item *x* has changed since the last time the client cached *x*) or just use the cached copy to answer the user's request (if data item *x* has not been modified since the last time the client received a copy of *x*). For the polling-every-time approach, every time a data item is requested, the clients need to poll the server to verify if the cached data item has changed. The polling-every-time caching strategy can be thought of as a special type of TTL-based scheme, with the TTL field equal to zero for every data item. Polling-every-time and TTL-based approaches are used in many existing Web caches. On the other hand, with *invalidation-based strategies*, the server initiates the cache consistency verification. Invalidation-based cache strategies are further classified into *stateless* and *stateful* approaches

In a *stateless* approach, the server does not maintain information about the cache contents of the clients; i.e., the server does not know what data are cached or how long they have been cached
by a particular client.

In *asynchronous* approaches,

Invalidation reports are sent out on data modification. In *synchronous* approaches, the server sends out invalidation reports periodically. For a detailed comparison and evaluation of different approaches in this category, interested readers should refer to Tan, Cai, and Ooi (2001). In *stateful* approaches, a server keeps track of the cache contents of its clients. Although stateful approaches also can be categorized into synchronous and asynchronous approaches, there are hardly any schemes in the *stateful synchronous* category. One example of a *stateful asynchronous* approach has been the Asynchronous Stateful (AS) scheme proposed by Kahol et al., 2001 Such push-based architectures present a new challenge for cache management.

Traditionally, caches are used to store the most frequently used data, but in push-based environments, the cost of obtaining the data also should be considered. For example, consider the following scenario from Acharya and colleagues (1995): Assume that data item *x* is accessed 1 percent of the time at a client *C* and is also broadcasted 1 percent of the time and that another data item *y* is accessed 0.5 percent of the time at *C* but is broadcast only 0.1 percent of the time. This implies that the time period between two occurrence of data item *y* on the broadcast channel *ty* is 10 times the broadcast period *tx* of data item *x*. If we choose to cache *x* instead of *y*, then the client will experience longer delays when a cache miss happens for *y*. This will adversely affect the average data access delay. Intuitively, the penalty for not caching a data item on the average data access latency is proportional to the product of access frequency and broadcast period. For this example, the product of access frequency and broadcast period for data item *y* is higher than that for data item *x*. Therefore, new cache management algorithms have been developed for pushed-based information systems that take into account the cost of a cache miss. Note that in traditional cache systems all cache misses are assumed to have the same cost. This is not so in push-based information access architectures. In general, an important metric to evaluate these cache management algorithms is the achievable *hit ratio*, the fraction of total data requests satisfied from the cache. This metric depends not only on cache management algorithms but also on the cache size and the particular request pattern. In mobile computing environments, the hit ratio should not be the only metric to evaluate cache management algorithms (because its underlying assumption is that all cache misses have equal cost. This assumption does not necessarily hold in weakly connected environments, where cache miss cost also depends on data size and timing. New metrics representing different cache costs in mobile computing environments therefore are needed. Let us consider one such metric called PIX. Acharya and colleagues (1995) proposed a cost-based page-replacement algorithm using PIX. Suppose that the access probability of data item *d* is *P* and the broadcast frequency is *X*, then the PIX value of *d* is *P/X*. For the preceding example, the algorithm replaces item *x* with *y* because *y* has a higher PIX value. Note that, in general, the broadcast pattern of the server also should be considered for the client's cache management algorithms.

Question Paper Solution

Branch : CS/IT Semester: 8th Subject: DIP Mid Term: I/[✓ II]/Extra/Imp.
Submitted By : Tapan Badal , Dr. Sanwta Dogiwal , Rashmi Dadhich

Sol^n 1 Image may be defined as a two dimensional signal that can be observed by human visual system. It is a two dimensional function, f(x,y), where x and y are spatial coordinates and the amplitude of f at any pair of coordinates (x,y) is called intensity or gray level of image at that point.

A digital image is composed of a finite number of elements each of which has a particular location & values. These elements are referred to as picture elements, image elements. pels and pixels.

For representation of an image, we assume that an image f(x,y) is sampled so that the resulting image has M rows & N columns. We say that the image is of size M×N The values of coordinates are discrete quantities & integer values are used for these discrete coordinates.

The image can be represented using a coordinate system as a matrix using f(x,y) as:-

$$f(x,y) = \begin{bmatrix} f(0,0) & f(0,1) & \cdots & f(0,N-1) \\ f(1,0) & f(1,1) & \cdots & f(1,N-1) \\ \vdots & \vdots & & \vdots \\ f(M-1,0) & f(M-1,1) & \cdots & f(M-1,N-1) \end{bmatrix}$$

The right side of this equation is a digital image by definition. Each element of this array is called an image element, picture element, pixel.

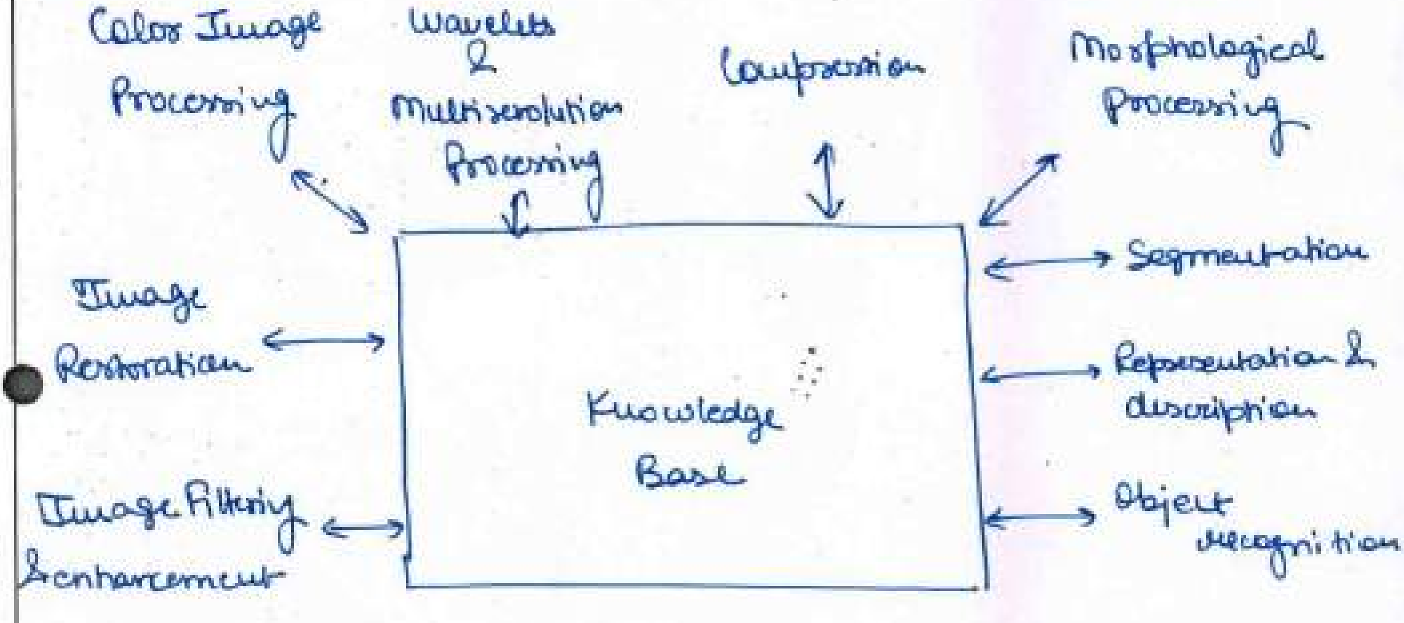
OR

Sol^n 1 Image Sampling & Quantization :-

→ Images can be acquired by numerous ways but the output of most sensors is a continuous voltage waveform whose amplitude & spatial behaviour are related to the physical phenomenon being sensed. To create a digital image, we need to convert the continuous sensed data into digital form. This involves two processes: sampling & quantization.

→ Digitizing the coordinate values is called sampling & digitizing the amplitude values is called quantization. Sampling means taking samples & quantization means diving in quanta (partitions). Sampling is done on an independent variable whereas quantization is done on a dependent variable. Eg:- in case of equation $y = \sin(x)$ sampling is done on variable x & quantization on variable Y.

Sol^{n}2



OR

Sol^{n}2 Color models: To describe the quality of a Chromatic light source, we have 3 basic quantities: Radiance, Luminance & brightness.

Radiance is the total amount of energy that flows from the light source & it is usually measured in watts (W). Luminance measured in lumens (lm) gives a measure of the amount of energy an observer perceives from a light source.

Brightness is a subjective descriptor that is practically impossible to measure. It embodies the achromatic notion of intensity & is one of the key factors in the describing color sensation.

① RGB color model:

→ The primary colors for vision are considered to be red green & blue.

→ We can represent RGB model with unit cube defined on R, G & B axes, origin represents black & vertex with coordinates (1, 1, 1) is white.

→ RGB color scheme is an additive model i.e. intensities of the primary colors are added to produce other colors. Thus, a color C_λ is expressed in RGB components as :

$$C_\lambda = RR + GG + BB$$

② CMY color model:

→ A color model defined with the primary colors cyan, magenta & yellow (CMY) is useful for describing color output to hard copy devices.

The Colors on the hard copy output are seen by reflected light, a subtractive process.
Eg: Cyan can be formed by adding green and blue light. Therefore, when white light is reflected light must have no red component. This is, red light is absorbed or subtracted by the ink.

3) HSV Color Model :

Color parameters in this model are Hue (H) saturation (S) & Value (V).

→ Hue is an attribute associated with the dominant wavelength is a mixture of light waves. Hue represents dominant color as perceived by an observer. Thus, when we call an object red, orange or yellow we are referring to call an object as its Hue.

• Saturation refers to the relative purity or the amount of white light mixed with a Hue. The pure Spectrum colors are fully saturated. Colors seen as Pink (Red & white) and lavender (violet & white) are less saturated, with the degree of saturation being inversely proportional to the amount of white light added.

→ Hue & Saturation taken together are called Chromaticity & therefore color may be characterized by Chromaticity & brightness.

Question Paper Solution

Branch : CS/IT Semester: 8 Subject: DIP Mid Term: I/II/Extra/Imp.
Submitted By : Tapan Badal, Dr. Sangita Dogiwal, Rashmi Dadhich

Solution 3: The discrete form of fourier transform in 2D space can be expressed as

$$F\{f(x,y)\} = F(u,v) = \sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x,y)\exp\left[-j2\pi\left(\frac{ux}{M}+\frac{vy}{N}\right)\right]$$

for $u=0,1...,M-1$ and $v=0,1,---,N-1$

Inverse Transform:

$$F^{-1}\{F(u,v)\} = f(x,y) = \frac{1}{MN}\sum_{u=0}^{M-1}\sum_{v=0}^{N-1} F(u,v)\exp\left[j2\pi\left(\frac{ux}{M}+\frac{vy}{N}\right)\right]$$

for $x=0,1,--,M-1$ and $y=0,1,--,N-1$

$$F(u) = \sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x)\, W_M^{ux} \cdot W_N^{vy}$$

where $$W_N = \exp\left[\frac{-j2\pi}{M}\right]$$

$$F = W_N f W_N^T$$

$$f = W_N^{-1} F (W^{-1})^T$$

W_N is a NxN matrix having its elements

$$W_N(u,x) = W_N^{ux} = \exp[-j2\pi ux/N]$$

for $N=4$: $$W_4 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & 1 & j \\ 1 & -1 & 1 & -1 \\ 1 & j & -1 & -j \end{bmatrix}$$

$$F(u,v) = W_N\, f(x,y)\, W_N^T$$

$$= \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & 1 & j \\ 1 & -1 & 1 & -1 \\ 1 & j & -1 & -j \end{bmatrix} \begin{bmatrix} 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & 1 & j \\ 1 & -1 & 1 & -1 \\ 1 & j & -1 & -j \end{bmatrix}$$

$$F(u,v) = \begin{bmatrix} 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 2 \end{bmatrix}$$

Backward fourier transformation:

$$f(x,y) = \frac{1}{MN}\left[W_4^* \, F(u,v) \, (W_4^*)^T\right]$$

$$= \frac{1}{16}\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & j & 1 & -j \\ 1 & -1 & 1 & -1 \\ 1 & -j & -1 & j \end{bmatrix} \begin{bmatrix} 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 2 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & j & 1 & -j \\ 1 & -1 & 1 & -1 \\ 1 & -j & -1 & j \end{bmatrix}$$

$$= \frac{1}{16}\begin{bmatrix} 32 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 6 \end{bmatrix}$$

$$= \begin{bmatrix} 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

While, applying inverse fourier transform we get original image back. Hence DFT works.

Question Paper Solution

Branch : CS| IT .... Semester: VIII Subject: D.I.P .... Mid Term: I/II/Extra/Imp.
Submitted By : Tapan Bachal/Dr. Sanmita Dagiwal, Rashmi Dadhich

Solⁿ 4 $MN = 4096$

r_K	n_K	p_r	s_K	Round off s_K
$r_0=0$	790	0.19	$s_0 = 7 \times 0.19 = 1.33$	1
$r_1=1$	1023	0.25	$s_1 = 7 \times (0.19+0.25) = 7 \times 0.44 = 3.08$	3
$r_2=2$	850	0.21	$s_2 = 7 \times (0.19+.25+.21) = 7 \times .65 = 4.55$	5
$r_3=3$	656	0.16	$s_3 = 7 \times (.19+.25+.21+.16) = 7 \times .81 = 5.67 = 6$	6
$r_4=4$	329	0.08	$s_4 = 7 \times (.19+.25+.21+.16+.08) = 7 \times .89 = 6.23$	6
$r_5=5$	245	0.06	$s_5 = 7 \times (.19+.25+.21+.16+.08+.06) = 7 \times 0.95 = 6.65$	7
$r_6=6$	122	0.03	$s_6 = 7 \times (.19+.25+.21+.16+.08+.06+.03) = 7 \times 0.98 = 6.86$	7
$r_7=7$	81	0.02	$s_7 = 7(.19+.25+.21+.16+.08+.06+.03+.02) = 7 \times 1 = 7$	7

$p_r = n_K / MN$; $s_K = T(r_K) = (L-1) \sum_{j=0}^{K} p_r(r_j)$

Now plot these values on the graph

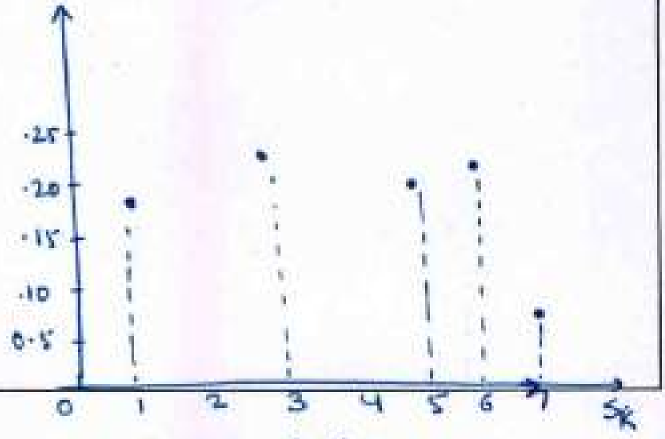
$1 \rightarrow 0.19$

$3 \rightarrow 0.25$

$5 \rightarrow 0.21$

$6 \rightarrow$ ~~0.16+0.08~~ $(656+329)/4096 = 0.24$

$7 \rightarrow (245+122+81)/4096 = 0.10$



Equalized histogram

Sol^n 4

OR

Histogram is a graph that shows frequency of anything. In DIP histogram shows the freq. of pixels intensity values. In an image histogram, the x axis shows the gray level intensities & the y axis shows the frequency of these intensities. The histogram of a a digital image with intensity levels in the range [0, L-1] is a discrete fun^n $h(r_k) = n_k$, where r_k is the kth intensity value & n_k is the number of pixels in the image with intensity r_k.

→ A method used to generate a processed image that has a specified histogram is called histogram matching or histogram specification.

→ It involves more computations then histogram equalization.

Sol^n 5 1) Spatial filtering :

→ Filtering refers to accepting (passing) or rejecting certain frequency components. For example; a filter that passes low frequencies is called lowpass filter. The net effect produced by a lowpass filter is to blur (smooth) an image

→ We can accomplish a similar smoothing directly on the image itself by using spatial filters (also called spatial masks, kernels, templates and windows).

→ Spatial filter consists of :

① a neighborhood (typically a small rectangle)

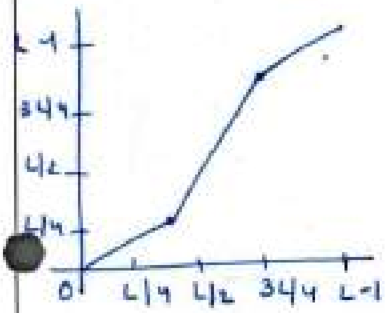
② a predefined operation that is performed on the image pixels encompassed by the neighborhood.

→ Correlation & Convolution are used as spatial filters. Correlation is the process of moving a filter mask over the image & computing the sum of products at each location. The mechanics of convolution are the same, except that the filter is first rotated by 180°.

Question Paper Solution

Branch : CS/IT Semester: VIII Subject: DIP Mid Term: I/II/Extra/Imp.
Submitted By : Tapas Badal, Dr. Sanwta Dagiwal, Rashmi Dadhich

2) Contrast Stretching : It is a process that expands the range of intensity levels in an image so that it spans the full intensity range of the recording medium or display device.



The diagram shows a typical transformation used for contrast stretching. The location of points $(r_1 \& s_1)$ and (r_2, s_2) control the shape of the transformation fun^n.

3) Properties of DFT : (a) The Separability of Property
(b) Translation
(c) Periodicity and conjugate symmetry
(d) Rotational Property
(e) Distributivity & scaling
(f) Average value Property.

5) Image Acquisition:

It means capturing the real scene in order to store it so that it can be processed. For this we have various sensors that work on energy incident on them like single imaging sensor, line sensor and array sensor etc.

Question Paper Solution

Branch : CS/IT Semester: 8 Subject: DIP Mid Term: [✓ I]/II/Extra/Imp.

Submitted By : Tapas Badal, Dr. Sanwita Dogiwal, Rashmi Dadhich

4> Bit Plane Slicing :

→ Pixels are digital numbers composed of bits. For Example: the intensity of each pixel in a 256 level gray scale image is composed of 8 bits (i.e. one byte)

→ Instead of highligting intensity level ranges, we could highlight the contribution made to total image appearance by specific bits.

→ It is important to note that the higher order bit planes, especially the last two, contain the significant amount of the visually significant data. The lower order planes contibute to more subtle intensity details in the image.

→ Decomposing an image into its bit planes is useful for analyzing the relative importance of each bit in the image, a process that aids in determining the adequacy of the number of bits used to quantize the image.

→ This type of decomposition is also useful for image compression in which fewer than all planes are used in reconstructing an image.

Swami Keshvanand Institute of Technology, Management &Gramothan,
Ramnagaria, Jagatpura, Jaipur-302017

Question Paper Solution

Branch : ......IT...................... Semester: ......VIII... Subject: ......STV........................
Mid Term: I/II/Extra/Imp.I
Submitted By :.........Richa Rawal...

Solutions:Q1.

V-model is also known as Verification and Validation (V&V) model. In this each phase of SDLC must be completed before the next phase starts. It follows a sequential design process same like waterfall model.Don't you think that why do we use this V Model, if it is same as Waterfall Model.

Let me mention the next point on why do we need this Verification and Validation Model.

It overcomes the disadvantages of waterfall model. In the waterfall model, we have seen that testers involve in the project only at the last phase of the development process.

In this, test team involves in the early phases of SDLC. Testing starts in early stages of product development which avoids downward flow of defects which in turns reduces lot of rework. Both teams (test and development) work in parallel. The test team works on various activities like preparing test strategy, test plan and test cases/scripts while the development team works on SRS, Design and Coding.

Once the requirements were received, both the development and test team start their activities.

Deliverables are parallel in this model. Whilst, developers are working on SRS (System Requirement Specification), testers work on BRS (Business Requirement Specification) and prepare ATP(Acceptance Test Plan) and ATC (Acceptance Test Cases) and so on.

Testers will be ready with all the required artifacts (such as Test Plan, Test Cases) by the time developers release the finished product. It saves lots of time.

Let's see the how the development team and test team involves in each phase of SDLC in V Model.

1. Once client sends BRS, both the teams (test and development) start their activities. The developers translate the BRS to SRS. The test team involves in reviewing the BRS to find the missing or wrong requirements and writes acceptance test plan and acceptance test cases.
2. In the next stage, the development team sends the SRS the testing team for review and the developers start building the HLD (High Level Design Document) of the product. The test team involves in reviewing the SRS against the BRS and writes system test plan and test cases.
3. In the next stage, the development team starts building the LLD (Low Level Design) of the product. The test team involves in reviewing the HLD (High Level Design) and writes Integration test plan and integration test cases.
4. In the next stage, the development team starts with the coding of the product. The test team involves in reviewing the LLD and writes functional test plan and functional test cases.
5. In the next stage, the development team releases the build to the test team once the unit testing was done. The test team carries out functional testing, integration testing, system testing and acceptance testing on the release build step by step.

Advantages:

Testing starts in early stages of product development which avoids downward flow of defects and helps to find the defects in the early stages

Test team will be ready with the test cases by the time developers release the software which in turns saves a lot of time

Testing is involved in every stage of product development. It gives a quality product.

Total investment is less due to less or no rework.

Disadvantages:

Initial investment is more because test team involves right from the early stage.

Whenever there is change in requirement, the same procedure continues. It leads more documentation work.

Applications:

Long term projects, complex applications, When customer is expecting a very high quality product with in stipulated time frame because every stage is tested and developers & testers are working in parallel

"OR'

Introduction to Functional testing

As the name goes, a functional test is a kind of black box testing that is performed to confirm that the functionality of an application or system is behaving as expected.

Therefore, there must be something that defines what is acceptable behavior and what is not. This is specified in a determine the conformance of the application or system to it. Additionally, sometimes this could also entail actual business side scenarios to be validated.

Therefore, functionality testing can be carried out via **two popular techniques**:

- Testing based on Requirements: Contains all the functional specific...ions which form a basis for all the tests to be conducted.
- Testing based on Business scenarios: Contains the information about how the system will be perceived from a business process perspective.

Functional testing Types: It has many categories and these can be used based on the scenario. The most prominent types are discussed in brief below:

- **Unit Testing:** Unit testing is usually performed by the developer who writes different code units that could be related or unrelated to achieve a particular functionality.

Therefore, this usually entails writing unit tests which would call the methods in each unit and validate that when the needed parameters are passed, its return value is as expected. Code coverage is an important part of unit testing where test cases need to exist to cover the below three:

i)Linecoverage

ii)Codepathcoverage

iii) Method coverage

- **Sanity Testing:** Testing that is done to ensure that all major and vital functionalities of the application/system are working correctly. This is generally done after a smoke test.
- **Smoke testing:** Testing that is done after each build is released to test to ensure build stability. It's also called build verification testing.
- **Regression tests:** Testing performed to ensure that the adding new code, enhancements, fixing of bugs is not breaking existing functionality or cause instability and still works according to the specifications. Regression tests need not be as extensive as the actual functional tests but should ensure just the amount of coverage to certify that the functionality is stable.
- **Integration tests:** When the system relies on multiple functional modules that might individually work perfectly, but have to work coherently when clubbed together to achieve an end to end scenario, validation of such scenarios is called integration testing.
- **Beta/Usability testing:** Product is exposed to the actual customer in a production like an environment and they test the product. The user's comfort is derived from this and feedback is taken. This is similar to User Acceptance testing.

Functional Testing Process

Approach, Techniques, and Examples

Functional or behavioral testing generates an output based on the given inputs and determines if the System is functioning correctly as per the specifications. So a pictorial representation looks like something below:

Approach:

Different kind of scenarios can be thought of and authored in the form of "test cases". As QA folks, we all know how the skeleton of a test case looks.

It has mostly four parts to it:

- Test summary
- Pre-requisites
- Test Steps and
- Expected results.

The basic approach to testing this scenario can be classified into two broad categories:

1. Positive testing and
2. negative testing

each of these cateeories has their own subsection of tests that will be carried out.

vital to customer usage.
Negative scenarios ensure that the product behaves properly even subjected to unexpected data.

Testing Techniques:

#1) End user based/System tests: The system under test may have many components that when coupled together achieve the user scenario. In the example, a customer scenario would include HRMS application loading, entering the correct credentials, going to the home page, performing some actions and logging out of the system. This particular flow has to work without any errors for a basic business scenario.
#2) Equivalence tests: In Equivalence partitioning, the test data are segregated into various partitions called equivalence data classes. Data in each partition must behave the same way therefore only one condition needs to be tested. Similarly, if one condition in a partition doesn't work, then none of the others will work.
#3) Boundary Value tests: Boundary tests imply data limits to the application and validate how it behaves. Therefore, if the inputs are supplied beyond the boundary values, it is considered to be neg.:tive testing
#4) Decision-based tests: Decision-based tests are centered around the ideology of the possible outcomes of the system when a particular condition is met. In the scenario given above, the following decision-based tests can be immediately derived:

1. If wrong credentials are entered, it should indicate that to the user and reload the login page.
2. If the user enters the correct credentials, it should take the user to the next UI.
3. If the user enters the correct credentials but wishes to cancel login, it should not take the user to the next UI and reload the login page.

The following items are the knowledge source of functional(**Black Box**) testing:
1). Requirement document
2). Specifications.
3) Domain Knowledge.
4). Defect Analysis data

Ans2. **Integration testing** tests integration or interfaces between components, interactions to different parts of the system such as an operating system, file system and hardware or interfaces between systems.

Approaches/Methodologies/Strategies of Integration Testing:
The Software Industry uses variety of strategies to execute Integration testing , viz.
Big Bang Approach :
Incremental Approach: which is further divided into following
Top Down Approach
Bottom Up Approach
Sandwich Approach - Combination of Top Down and Bottom Up
Below are the different strategies, the way they are executed and their limitations as well advantages
Big Bang Approach:
Here all component are integrated together at **once**, and then tested.
Advantages:
Convenient for small systems.
Disadvantages:
Fault Localization is difficult.
Given the sheer number of interfaces that need to be tested in this approach, some interfaces links to be tested could be missed easily.
Since the integration testing can commence only after "all" the modules are designed, testing team will have less time for execution in the testing phase.
Since all modules are tested at once, high risk critical modules are not isolated and tested on priority. Peripheral modules which deal with user interfaces are also not isolated and tested on priority.

modules are added and tested for the proper functioning. Process continues until all of the modules are joined and tested successfully.

This process is carried out by using dummy programs called **Stubs and Drivers**. Stubs and Drivers do not implement the entire programming logic of the software module but just simulate data communication with the calling module.

Stub: Is called by the Module under Test.

Driver: Calls the Module to be tested.

Incremental Approach in turn is carried out by two different Methods:

Bottom Up

Top Down

Bottom up Integration

In the bottom up strategy, each module at lower levels is tested with higher modules until all modules are tested. It takes help of Drivers for testing

Advantages:

Fault localization is easier.

No time is wasted waiting for all modules to be developed unlike Big-bang approach

Disadvantages:

Critical modules (at the top level of software architecture) which control the flow of application are tested last and may be prone to defects.

Early prototype is not possible

Top down Integration:

In Top to down approach, testing takes place from top to down following the control flow of the software system.

Takes help of stubs for testing.

Advantages:

Fault Localization is easier.

Possibility to obtain an early prototype.

Critical Modules are tested on priority; major design flaws could be found and fixed first.

Disadvantages:

Needs many Stubs.

Modules at lower level are tested inadequately.

"OR"

What is performance testing ? Write the tool and process for performance testing with suitable examples.

Software Performance testing is type of testing perform to determine the performance of system to major the measure, validate or verify quality attributes of the system like responsiveness, Speed, Scalability, Stability under variety of load conditions. The system is tested under a mixture of load conditions and check the time required responding by the system under varying workloads. Software performance testing involves the testing of application under test to ensure that application is working as expected under variety of load conditions. The goal of performance testing is not only find the bugs in the system but also eliminate the performance bottlenecks from the system.

Types of Performance Testing:

1) Load Testing:

2) Stress Testing:

5) Scalability Testing:

6) Volume testing:

Top Performance Testing Tools:

- WebLOAD
- LoadRunner
- Apache JMeter
- NeoLoad
- LoadUI
- OpenSTA
- WAPT
- LoadImpact
- Loadster
- Httperf
- Rational Performance Tester
- QEngine (ManageEngine)
- Testing Anywhere
- CloudTest
- Loadstorm

Performance Testing Process:

The following seven activities that most commonly occur across successful performance-testing projects.

1) Identify your testing environment –

Do proper requirement study & analyzing test goals and its objectives. Also determine the testing scope along with test Initiation Checklist. Identify the logical and physical production architecture for performance testing, identify the software, hardware and networks configurations required for kick off the performance testing. Compare the both test and production environments while identifying the testing environment. Get resolve the environment-related concerns if any, analyze that whether additional tools are required for performance testing. This step also helps to identify the probable challenges tester may face while performance testing.

2) Identify the performance acceptance criteria –

Identify the desired performance characteristics of the application like Response time, Throughput and Resource utilization.

3) Plan & design performance tests –

Planning and designing performance tests involves identifying key usage scenarios, determining appropriate variability across users, identifying and generating test data, and specifying the metrics to be collected. Ultimately, these items will provide the foundation for workloads and workload profiles. The output of this stage is prerequisites for Test execution are ready, all required resources, tools & test data are ready.

4) Configuring the test environment –

Prepare with conceptual strategy, available tools, designed tests along with testing environment before execution. The output of this stage is configured load-generation environment and resource monitoring tools.

According to test planning and design create your performance tests.

6) Execute the tests –

- Collect and analyze the data.
- Problem Investigation like bottlenecks (memory, disk, processor, process, cache, network, etc.) resource usage like (memory, CPU, network, etc.,)
- Generate the Performance analysis reports containing all performance attributes of the application.
- Based on the analysis prepare recommendation report.
- Repeat the above test for the new build received from client after fixing the bugs and implementing the recommendations

7) Analyze Results, Report, and Retest

Consolidate, analyze and share test results.

Based on the test report re-prioritize the test & re-execute the same. If any specific test result within the specified metric limit & all results are between the thresholds limits then testing of same scenario on particular configuration is completed.

Q3. What is regression testing? List the steps involved in doing regession testing.
SOL: Regression testing a black box testing technique that consists of re-executing those tests that are impacted by the code changes. These tests should.be executed as often as possible throughout the software development life cycle.

Types of Regression Tests:

- **Final Regression Tests: -** A "final regression testing" is performed to validate the build that hasn't changed for a period of time. This build is deployed or shipped to customers.
- **Regression Tests: -** A normal regression testing is performed to verify if the build has NOT broken any other parts of the application by the recent code changes for defect fixing or for enhancement.

Selecting Regression Tests:

- Requires knowledge about the system and how it affects by the existing functionalities.
- Tests are selected based on the area of frequent defects.
- Tests are selected to include the area, which has undergone code changes many a times.
- Tests are selected based on the criticality of the features.

Regression Testing Steps:
Regression tests are the ideal cases of automation which results in betterReturn On Investment (ROI).

- Select the Tests for Regression.
- Choose the apt tool and automate the Regression Tests
- Verify applications with Checkpoints
- Manage Regression Tests/update when required

- Manage Regression Tests/update when required
- Schedule the tests
- Integrate with the builds
- Analyze the results

Types of Regression testing techniques:
We have four types of regression testing techniques. They are as follows:

1) Corrective Regression Testing: Corrective regression testing can be used when there is no change in the specifications and test cases can be reused.

2) Progressive Regression Testing: Progressive regression testing is used when the modifications are done in the specifications and new test cases are designed.

3) Retest-All Strategy: The retest-all strategy is very tedious and time consuming because here we reuse all test which results in the execution of unnecessary test cases. When any small modification or change is done to the application then this strategy is not useful.

4) Selective Strategy: In selective strategy we use a subset of the existing test cases to cut down the retesting effort and cost. If any changes are done to the program entities, e.g. functions, variables etc., then a test unit must be rerun. Here the difficult part is to find out the dependencies between a test case and the program entities it covers.

When to use it:

Regression testing is used when:

- Any new feature is added
- Any enhancement is done
- Any bug is fixed
- Any performance related issue is fixed

Advantages of Regression testing:

- It helps us to make sure that any changes like bug fixes or any enhancements to the module or application have not impacted the existing tested code.
- It ensures that the bugs found earlier are NOT creatable.
- Regression testing can be done by using the automation tools
- It helps in improving the quality of the product.

Disadvantages of Regression testing:

- If regression testing is done without using automated tools then it can be very tedious and time consuming because here we execute the same set of test cases again and again.
- Regression test is required even when a very small change is done in the code because this small modification can bring unexpected issues in the existing functionality.

When a software testing performed without proper planning and documentation, it is said to be Adhoc Testing. Such kind of tests are executed only once unless we uncover the defects.

Adhoc Tests are done after formal testing is performed on the application. Adhoc methods are the least formal type of testing as it is NOT a structured approach. Hence, defects found using this method are hard to replicate as there are no test cases aligned for those scenarios.

Testing is carried out with the knowledge of the tester about the application and the tester tests randomly without following the specifications/requirements. Hence the success of Adhoc testing depends upon the capability of the tester, who carries out the test. The tester has to find defects without any proper planning and documentation, solely based on tester's intuition.

When to Execute Adhoc Testing?

Adhoc testing can be performed when there is limited time to do exhaustive testing and usually performed after the formal test execution. Adhoc testing will be effective only if the tester has in-depth understanding about the System Under Test.

Forms of Adhoc Testing :

1. **Buddy Testing:** Two buddies, one from development team and one from test team mutually work on identifying defects in the same module. Buddy testing helps the testers develop better test cases while development team can also make design changes early. This kind of testing happens usually after completing the unit testing.
2. **Pair Testing:** Two testers are assigned the same modules and they share ideas and work on the same systems to find defects. One tester executes the tests while another tester records the notes on their findings.
3. **Monkey Testing:** Testing is performed randomly without any test cases in order to break the system.

Various ways to make Adhoc Testing More Effective

1. **Preparation:** By getting the defect details of a similar application, the probability of finding defects in the application is more.
2. **Creating a Rough Idea:** By creating a rough idea in place the tester will have a focussed approach. It is NOT required to document a detailed plan as what to test and how to test.
3. **Divide and Rule:** By testing the application part by part, we will have a better focus and better understanding of the problems if any.
4. **Targeting Critical Functionalities:** A tester should target those areas that are NOT covered while designing test cases.
5. **Using Tools:** Defects can also be brought to the lime light by using profilers, debuggers and even task can also be tested using automated tools.

monitors. Hence being proficient in using these tools one can uncover several defects.

6. **Documenting the findings:** Though testing is performed randomly, it is better to document the tests if time permits and note down the deviations if any. If defects are found, corresponding test cases are created so that it helps the testers to retest the scenario.

Swami Keshvanand Institute of Technology, Management &Gramothan,
Ramnagaria, Jagatpura, Jaipur-302017

Question Paper Solution

Branch: IT **Semester: VIII** **Subject: DCT** **Mid Term: I**

Submitted By: Pankaj Jadwal

Q 1. What do you mean by compression? Explain different type of compression techniques along with measure of performance?

In signal processing, data compression, source coding, or bit-rate reduction involves encoding information using fewer bits than the original representation. Compression can be either lossy or lossless. Lossless compression reduces bits by identifying and eliminating statistical redundancy. No information is lost in lossless compression. Lossy compression reduces bits by removing unnecessary or less important information. The process of reducing the size of a data file is often referred to as data compression. In the context of data transmission, it is called source coding; encoding done at the source of the data before it is stored or transmitted. Source coding should not be confused with channel coding, for error detection and correction or line coding, the means for mapping data onto a signal. Compression is useful because it reduces resources required to store and transmit data. Computational resources are consumed in the compression process and, usually, in the reversal of the process (decompression). Data compression is subject to a space–time complexity trade-off. For instance, a compression scheme for video may require expensive hardware for the video to be decompressed fast enough to be viewed as it is being decompressed, and the option to decompress the video in full before watching it may be inconvenient or require additional storage. The design of data compression schemes involves trade-offs among various factors, including the degree of compression, the amount of distortion introduced (when using lossy data compression), and the computational resources required to compress and decompress the data.

Lossless

Lossless data compression algorithms usually exploit statistical redundancy to represent data without losing any information, so that the process is reversible. Lossless compression is possible because most real-world data exhibits statistical redundancy. For example, an image may have areas of color that do not change over several pixels; instead of coding "red pixel, red pixel,." the data may be encoded as "279 red pixels". This is a basic example of run-length encoding; there are many schemes to reduce file size by eliminating redundancy.

The Lempel–Ziv (LZ) compression methods are among the most popular algorithms for lossless storage.[7] DEFLATE is a variation on LZ optimized for decompression speed and compression ratio, but compression can be slow. DEFLATE is used in PKZIP, Gzip, and PNG. LZW (Lempel–Ziv–Welch) is used in GIF images. LZ methods use a table-based compression model where table entries are substituted for repeated strings of data. For most LZ methods, this table is generated dynamically from earlier data in the input. The table itself is often Huffman encoded (e.g. SHRI, LZX). Current LZ-based coding schemes that perform well are Brotli and LZX. LZX is used in Microsoft's CAB format

The best modern lossless compressors use probabilistic models, such as prediction by partial matching.

The Burrows–Wheeler transform can also be viewed as an indirect form of statistical modeling.

The class of grammar-based codes are gaining popularity because they can compress highly repetitive input extremely effectively, for instance, a biological data collection of the same or closely related species, a huge versioned document collection, internet archival, etc. The basic task of grammar-based codes is constructing a context-free grammar deriving a single string. Sequitur and Re-Pair are practical grammar compression algorithms for which software is publicly available.

In a further refinement of the direct use of probabilistic modeling, statistical estimates can be coupled to an algorithm called arithmetic coding. Arithmetic coding is a more modern coding technique that uses the mathematical calculations of a finite-state machine to produce a string of encoded bits from a series of input data symbols. It can achieve superior compression to other techniques such as the better-known Huffman algorithm. It uses an internal memory state to avoid the need to perform a one-to-one mapping of individual input symbols to distinct representations that use an integer number of bits, and it clears out the internal memory only after encoding the entire string of data symbols. Arithmetic coding applies especially well to adaptive data compression tasks where the statistics vary and are context-dependent, as it can be easily coupled with an adaptive model of the probability distribution of the input data. An early example of the use of arithmetic coding was its use as an optional (but not widely used) feature of the JPEG image coding standard. It has since been applied in various other designs including H.263, H.264/MPEG-4 AVC and HEVC for video coding.

Lossy

Lossy data compression is the converse of lossless data compression. In these schemes, some loss of information is acceptable. Dropping nonessential detail from the data source can save storage space. Lossy data compression schemes are designed by research on how people perceive the data in question. For example, the human eye is more sensitive to subtle variations in luminance than it is to the variations in color. JPEG image compression works in part by rounding off nonessential bits of information. There is a corresponding trade-off between preserving information and reducing size. A number of popular compression formats exploit these perceptual differences, including those used in music files, images, and video.

Lossy image compression can be used in digital cameras, to increase storage capacities with minimal degradation of picture quality. Similarly, DVDs use the lossy MPEG-2 video coding format for video compression.

In lossy audio compression, methods of psychoacoustics are used to remove non-audible (or less audible) components of the audio signal. Compression of human speech is often performed with even more specialized techniques; speech coding, or voice coding, is sometimes distinguished as a separate discipline from audio compression. Different audio and speech compression standards are listed under audio coding formats. Voice compression is used in internet telephony, for example, audio compression is used for CD ripping and is decoded by the audio players.

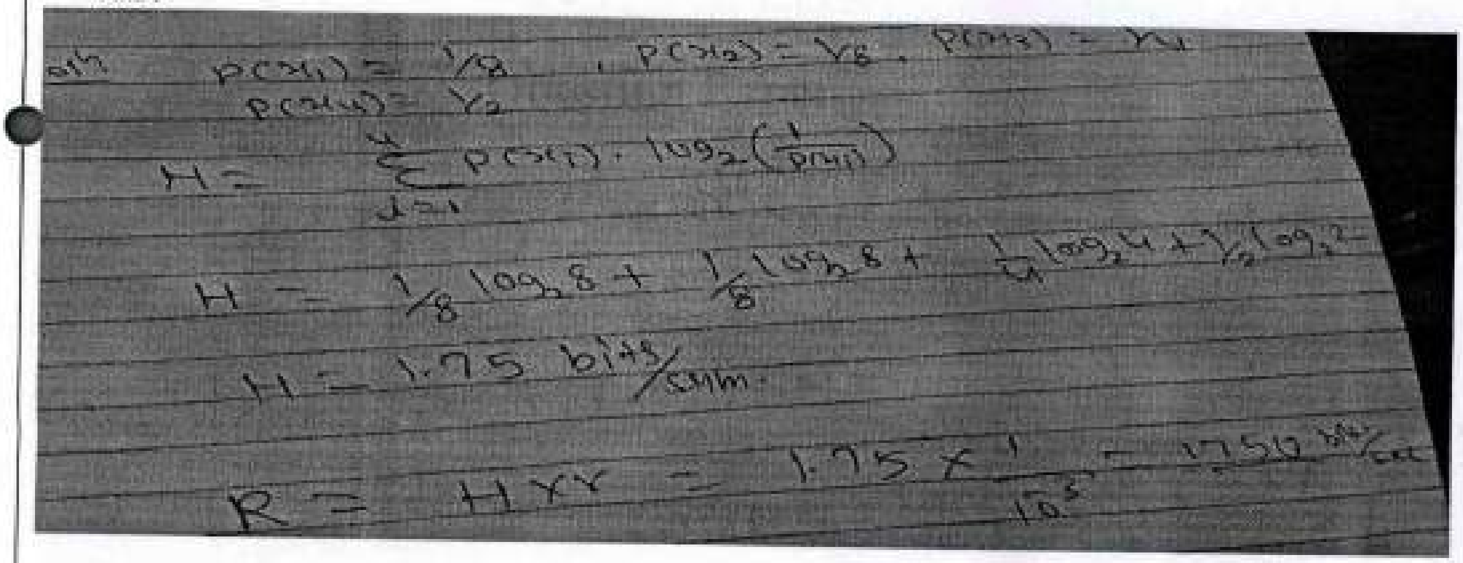
OR

Q. 1 .A source is transmitting four possible symbol, with the profanities of 1/8, 1/4, 1/2 respectively.

i. Find entropy

ii Find information rate if source transmit one symbol /msec.

Ans :-



ans $P(x_1) = 1/8$, $P(x_2) = 1/8$, $P(x_3) = 1/4$

$P(x_4) = 1/2$

$$H = \sum_{i=1}^{4} P(x_i) \cdot \log_2\left(\frac{1}{P(x_i)}\right)$$

$$H = \frac{1}{8}\log_2 8 + \frac{1}{8}\log_2 8 + \frac{1}{4}\log_2 4 + \frac{1}{2}\log_2 2$$

$H = 1.75$ bits/sym.

$R = H \times r = 1.75 \times \frac{1}{10^{3}} = 1750$ bits/sec

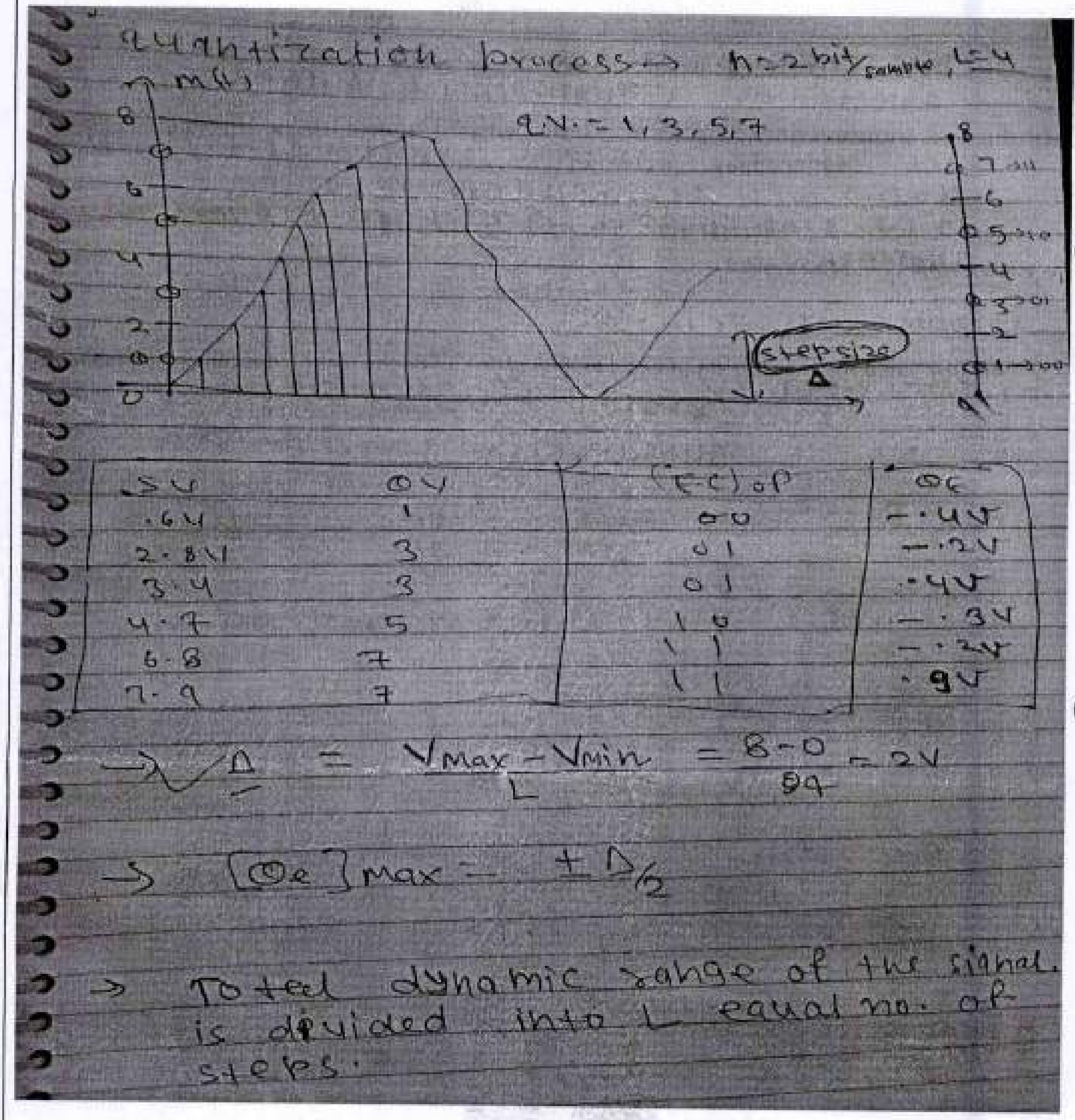
Question Paper Solution

Branch: IT **Semester:** VIII **Subject:** DCT **Mid Term:** I

Submitted By: Pankaj Jadwal

Q2. What do you mean by quantization? Explain with the help of proper example along with all formulas concern with quantization.

Ans :-



quantization process → $n = 2$ bit/sample, $L = 4$

m(t)
8
6
4
2
0
Q.V. = 1, 3, 5, 7
8
7 – 11
6
5 – 10
4
3 – 01
2
1 – 00
step size
Δ

SV	QV	(E-C) o/p	Qe
.6V	1	0 0	–.4V
2.8V	3	0 1	–.2V
3.4	3	0 1	.4V
4.7	5	1 0	–.3V
6.8	7	1 1	–.2V
7.9	7	1 1	.9V

→ $\Delta = \frac{V_{max} - V_{min}}{L} = \frac{8-0}{4} = 2V$

→ $[Q_e]_{max} = \pm \Delta/2$

→ Total dynamic range of the signal is divided into L equal no. of steps.

→ The middle of each step will be selected as a quantization voltage.

→ each of the sample corresponds to a step will be roundoff to, middle of the step or to the nearest quantization Voltage

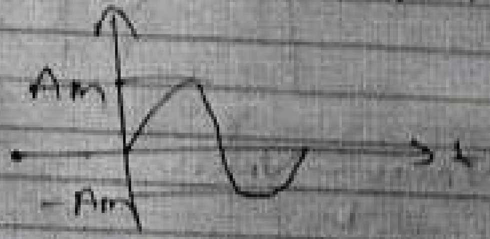
Formule in PCM →

→ No. of bits/sample = n

→ No. of quantization levels L = 2^n

→ $\Delta = \frac{V_{max} - V_{min}}{L} = \frac{V_{pp}}{2^n}$

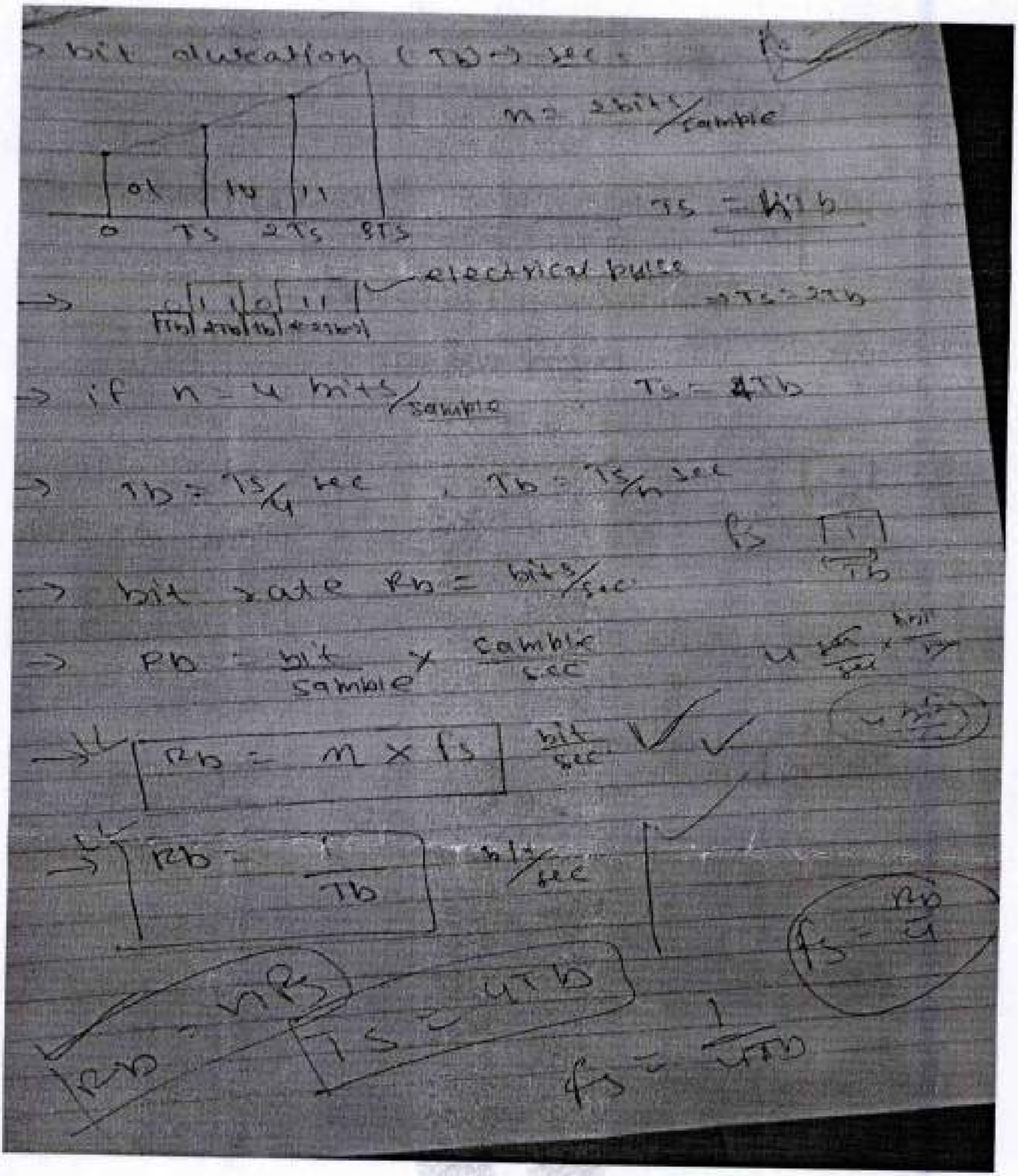
For $m(t) = A_m \cos 2\pi f_m t$ or $A_m \sin 2\pi f_m t$



$\Delta = \frac{2A_m}{2^n}$

$Q_e = SV - QV$

(Q_e) Max $= \pm \Delta/2$



→ bit duration (T_b) → sec

$n = 2$ bits/sample

01 10 11

0 T_s $2T_s$ $3T_s$

$T_s = 2T_b$

→ 0 1 1 0 1 1 ~ electrical pulse

T_b 2T_b T_b 2T_b

→ $T_s = 2T_b$

→ if $n = 4$ bits/sample $T_s = 4T_b$

→ $T_b = T_s/4$ sec , $T_b = T_s/n$ sec

T_b 1 1

→ bit rate R_b = bits/sec

→ $R_b = \frac{\text{bit}}{\text{sample}} \times \frac{\text{sample}}{\text{sec}}$ $4 \frac{\text{bit}}{\text{sec}} \times \frac{\text{bit}}{T_b}$

$4 \frac{\text{bit}}{T_b}$

→ $\boxed{R_b = n \times f_s}$ $\frac{\text{bit}}{\text{sec}}$ ✓ ✓

→ $\boxed{R_b = \frac{1}{T_b}}$ bits/sec

$f_s = \frac{R_b}{4}$

$R_b = nB$

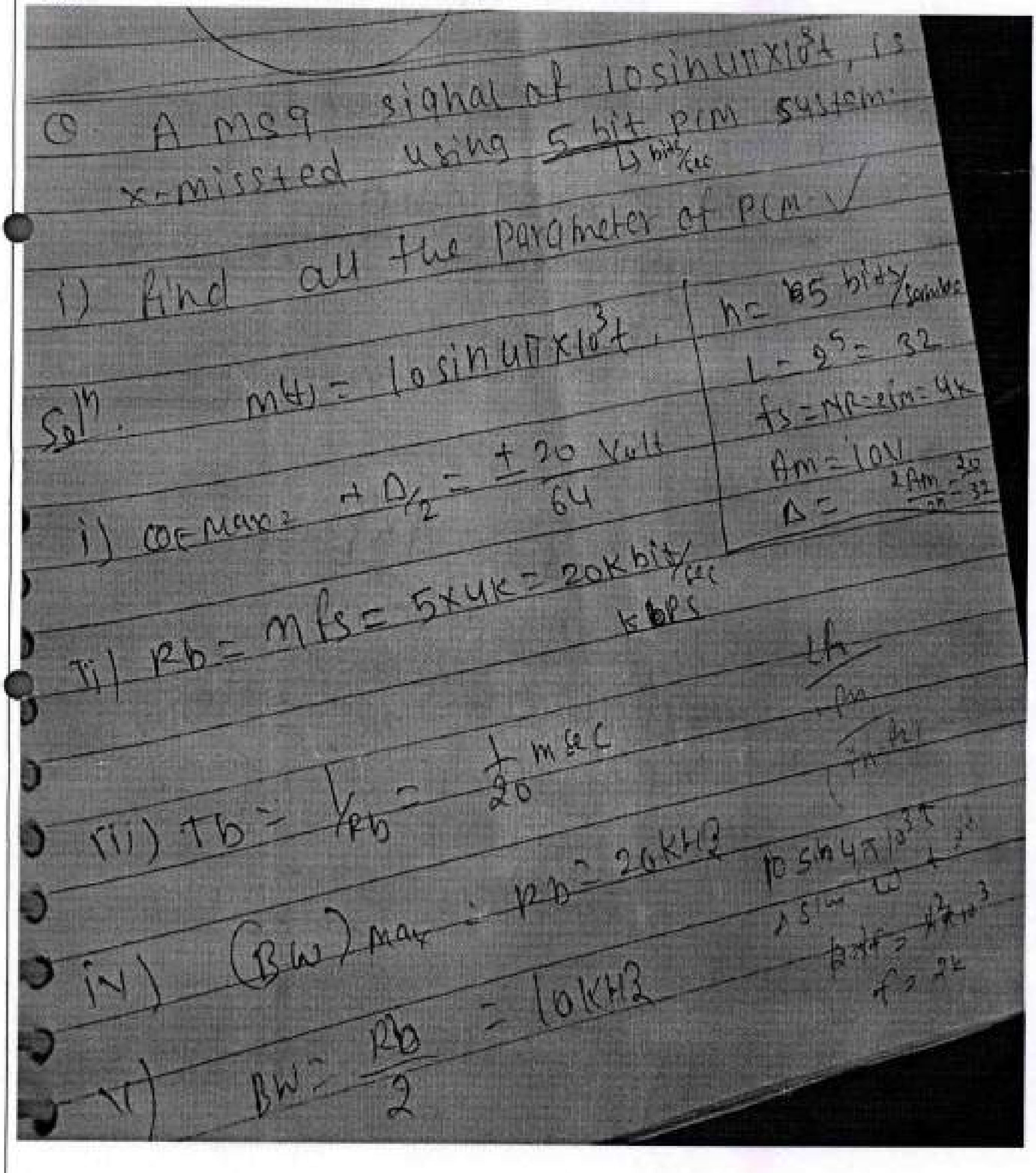
$T_s = 4T_b$

$f_s = \frac{1}{4T_b}$

OR

Q2. A message signal of 10sin(4π10^{3}t) is transmitted using 5 bit PCM (quantisation) system. Find all parameter of the system.

Ans:-



Q A msg signal of $10\sin 4\pi \times 10^3 t$, is x-missted using 5 bit PCM system. 5 bits/sec

i) find all the parameter of PCM ✓

Sol^n. $m(t) = 10\sin 4\pi \times 10^3 t$.

$n = 5$ bits/sample
$L = 2^5 = 32$
$f_s = NR = 2f_m = 4k$
$A_m = 10V$
$\Delta = \frac{2A_m}{2^n} = \frac{20}{32}$

i) $Q_{E\,max} = \pm \Delta/2 = \frac{\pm 20}{64}$ Volt

ii) $R_b = n f_s = 5 \times 4k = 20k$ bits/sec
kbps

iii) $T_b = 1/R_b = \frac{1}{20}$ msec

iv) $(BW)_{max} = R_b = 20kHz$

v) $BW = \frac{R_b}{2} = 10kHz$

$10\sin 4\pi 10^3 t$
$A\sin \omega t$
$2\pi f t = 4\pi \times 10^3$
$f = 2k$

Q3 **Write short note on.**

a. Prefix Code

Ans :- A prefix code is a type of code system (typically a variable-length code) distinguished by its possession of the "prefix property", which requires that there is no whole code word in the system that is a prefix (initial segment) of any other code word in the system. For example, a code with code words {9, 55} has the prefix property; a code consisting of {9, 5, 59, 55} does not, because "5" is a prefix of "59" and also of "55". A prefix code is a uniquely decodable code: given a complete and accurate sequence, a receiver can identify each word without requiring a special marker between words. However, there are uniquely decodable codes that are not prefix codes; for instance, the reverse of a prefix code is still uniquely decodable (it is a suffix code), but it is not necessarily a prefix code.

Prefix codes are also known as prefix-free codes, prefix condition codes and instantaneous codes. Although Huffman coding is just one of many algorithms for deriving prefix codes, prefix codes are also widely referred to as "Huffman codes", even when the code was not produced by a Huffman algorithm. The term comma-free code is sometimes also applied as a synonym for prefix-free codes but in most mathematical books and articles a comma-free code is used to mean a self-synchronizing code, a subclass of prefix codes.

Using prefix codes, a message can be transmitted as a sequence of concatenated code words, without any out-of-band markers or (alternatively) special markers between words to frame the words in the message. The recipient can decode the message unambiguously, by repeatedly finding and removing sequences that form valid code words. This is not generally possible with codes that lack the prefix property, for example {0, 1, 10, 11}: a receiver reading a "1" at the start of a code word would not know whether that was the complete code word "1", or merely the prefix of the code word "10" or "11"; so the string "10" could be interpreted either as a single codeword or as the concatenation of the words "1" then "0".

The variable-length Huffman codes, country calling codes, the country and publisher parts of ISBNs, the Secondary Synchronization Codes used in the UMTS W-CDMA 3G Wireless Standard, and the instruction sets (machine language) of most computer micro architectures are prefix codes.

Prefix codes are not error-correcting codes. In practice, a message might first be compressed with a prefix code, and then encoded again with channel coding (including error correction) before transmission.

b. Minimum Varience Huffman Code vs Huffnan codes

Ans:- Huffman coding is based on the frequency of occurance of a data item (pixel in images). The principle is to use a lower number of bits to encode the data that occurs more frequently. Codes are stored in a ***Code Book*** which may be constructed for each image or a set of images. In all cases the code book plus encoded data must be transmitted to enable decoding.

The Huffman algorithm is now briefly summarised:

- A bottom-up approach

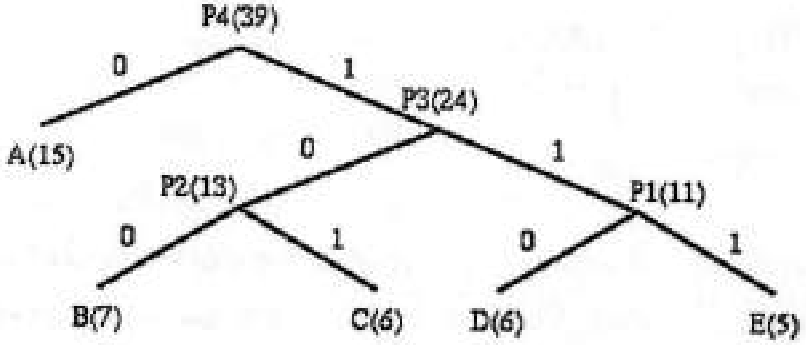
1. Initialization: Put all nodes in an OPEN list, keep it sorted at all times (e.g., ABCDE).

2. Repeat until the OPEN list has only one node left:

(a) From OPEN pick two nodes having the lowest frequencies/probabilities, create a parent node of them.

(b) Assign the sum of the children's frequencies/probabilities to the parent node and insert it into OPEN.

(c) Assign code 0, 1 to the two branches of the tree, and delete the children from OPEN.



Symbol	Count	log(1/p)	Code	Subtotal (# of bits)
A	15	1.38	0	15
B	7	2.48	100	21
C	6	2.70	101	18
D	6	2.70	110	18
E	5	2.96	111	15

TOTAL (# of bits): 87

The following points are worth noting about the above algorithm:

- Decoding for the above two algorithms is trivial as long as the coding table (the statistics) is sent before the data. (There is a bit overhead for sending this, negligible if the data file is big.)
- **Unique Prefix Property**: no code is a prefix to any other code (all symbols are at the leaf nodes) ->

great for decoder, unambiguous.

- If prior statistics are available and accurate, then Huffman coding is very good.

In the above example:

Number of bits needed for Huffman Coding is: 87 / 39 = 2.23