B. Tech.

EIGHT SEMESTER EXAMINATION, 2006-07

DISTRIBUTED SYSTEM

Time : 3 Hours] [Total Marks : 100

Note : 1. Attempt all questions.

2. All questions carry **equal** marks.

3. Be precise in your answer.

Q. 1. Attempt any four parts. $5 \times 4 = 20$

Q. 1. (a) What are distributed systems? Name two advantages and two disadvantages of distributed system over centralized ones. Explain in your own words the concept of parallelism transparency.

Ans. Distributed system appears to its user as centralized operating system for a single machine, but it runs on multiple independent computers. An identical copy of the operating system may run at every computer. The key concept is transparency. A distributed operating system extends the concept of resource management and user friendly interface for shared memory computers consisting of several computer connected by a communication network. **Advantages of distributed system** are naming scalability process synchronization compatibility, resource management and **disadvantages of distributed system** are absence of global clock and absence of shared memory.

Q. 1. (b) Given five types of hardware resource and five types of data or software resource that can be shared. Give example of their sharing as it occurs in distribute systems.

Ans (i) Central processing unit **(CPU)** - Performs most of the calculations which enable a computer to function, sometimes referred to as the "brain" of the computer.

(ii) hard disk drive (HDD), commonly referred to as a **hard drive, hard disk** or **fixed disk drive,** is a non-volatile storage device which stores digitally encoded data on rapidly rotating platters with magnetic surfaces. Strictly speaking, "drive" refers to a device distinct from its medium, such as a tape drive and its tape, or a floppy disk drive and its floppy disk. Early HDDs had removable media; however, an HDD today is typically a sealed unit (except for a filtered vent hole to equalize air pressure) with fixed media.

A HDD is a rigid-disk drive, although it is probably never referred to as such. By way of comparison, a so-called "floppy" drive (more formally, a diskette drive) has a disc that is flexible. Originally, the term "hard" was temporary slang, substituting "hard" for "rigid", before these drives had an established and universally-agreed-upon name. Some time ago, IBM's internal company term for an HDD was "file".

(iii) CD-ROM (an abbreviation of "Compact Disc read-only memory") is a Compact Disc that contains data accessible by a computer. While the Compact Disc format was originally designed for music storage and playback, the format was later adapted to hold any form of binary data. CD-ROMs are popularly used to distribute computer software, including games and multimedia applications, though any data can be stored (up to the capacity limit of a disc). Some CDs hold both computer data

and audio with the latter capable of being played on a CD player, whilst data (such as software or digital video) is only usable on a computer (such as PC CD-ROMs). These are called Enhanced CDs.

(iv) **microphone**, sometimes referred to as a **mike** or **mic** (both pronounced धड्उंचध), is an acoustic-to-electric transducer or sensor that converts sound into an electrical signal.Microphones are used in many applications such as telephones, tape recorders, hearing aids, motion picture production, live and recorded audio engineering, in radio and television broadcasting and in computers for recording voice, VoIP, and for non-acoustic purposes such as ultrasonic checking.

(v) A **loudspeaker, speaker,** or **speaker system** is an electromechanical transducer that converts an electrical signal to sound. The term *loudspeaker* can refer to individual devices (otherwise known as *drivers*), or to complete systems consisting of an enclosure incorporating one or more drivers and additional electronic components. Loudspeakers, as with other electro-acoustic transducers, are the most variable elements in an audio system and are responsible for the greatest degree of audible differences between sound systems.

To reproduce a wide range of frequencies, most loudspeaker systems require more than one driver, particularly for high sound pressure level or high fidelity applications. Individual drivers are used to cover different frequency ranges. The drivers are named subwoofers, for very low frequencies; woofers, for low frequencies; mid-range speakers, for middle frequencies; tweeters, for high frequencies; and, also, the so-called *supertweeters*, which are basically tweeters optimized for higher frequencies than a normal tweeter.

(vi) **Webcams (web cameras)** are small cameras, (usually, though not always, video cameras) whose images can be accessed using the World Wide Web, instant messaging, or a PC video conferencing application. The term webcam is also used to describe the low-resolution digital video cameras designed for such purposes, but which can also be used to record in a non-real-time fashion.

Web-accessible cameras involve a digital camera which uploads images to a web server, either continuously or at regular intervals. This may be achieved by a camera attached to a PC, or by dedicated hardware. Videoconferencing cameras typically take the form of a small camera connected directly to a PC. Analog cameras are also sometimes used (often of the sort used for closed-circuit television), connected to a video capture card and then directly or indirectly to the internet..

Q. 1. (c) Why is computer clock synchronization necessary? Describe the Design requirement for a system to synchronize the clocks in a distributed system?

Ans.Clock synchronization is a problem from computer science and engineering which deals with the idea that internal clocks of several computers may differ. Even when initially set accurately, real clocks will differ after some amount of time due to clock drift, caused by clocks counting time at slightly different rates. There are several problems that occur as a repercussion of rate differences and several solutions, some being more appropriate than others in certain contexts

Besides the incorrectness of the time itself, there are problems associated with clock skew that take on more complexity in a distributed system in which several computers will need to realize the same global time. For instance, in Unix systems the make command is used to compile new or modified code without the need to recompile unchanged code. The *make* command uses the clock of the machine it runs on to determine which source files need to be recompiled. If the sources reside on a separate file server and the two machines have unsynchronized clocks, the *make* program might not produce the correct result

Part-ii In a centralized system the solution is trivial; the centralized server will dictate the system time. Cristian's algorithm and the Berkeley Algorithm are some solutions to the clock synchronization problem in a centralized server environment. In a distributed system the problem takes on more complexity because a global time is not easily known. The most used clock synchronization solution on the internet is the Internet Network Time Protocol (NTP) which is a layered client-server architecture based on UDP message passing. Lamport timestamps and Vector clocks are concepts of the logical clocks in distributed systems.

Q. 1. (d) Give an example execution of the ring-based algorithm to show that processes are not necessarily granted entry to critical section in happened before order.

Ans.A distributed computing system is a collection of autonomous computing sites that do not share a global or common memory and communicate solely by exchanging messages over a communication facility. In a distributed computing system any given site (also referred to as "node") has only a partial or incomplete view of the total system and a system-wide common clock does not exist. Processes must share common hardware or software resources, cooperating in such a way that they can work in parallel and independently of each other. The access to a shared resource must be synchronized to ensure that only one process is making use of the resource at a given time. The problem of coordinating the execution of critical sections by each process is solved by providing mutually exclusive access in time to the critical section (CS). Each process must request permission to enter its critical section and must release it after it has completed its execution. A mutual exclusion algorithm must satisfy the following requirements [1, 2]:

i. At most one process can execute its critical section at a given time.

ii. If no process is in its critical section, any process requesting to enter its critical section must be allowed to do so at finite time.

iii. When competing processes concurrently request to enter their respective critical sections, the selection cannot be postponed indefinitely.

iv. A requesting process cannot be prevented by another one to enter its critical section within a finite delay.

To simplify, an algorithm must provide mutually exclusive access to the source, ensure deadlock freedom, ensure starvation freedom, and must provide some fairness in the order that requests are granted. The algorithm presented in this paper is based on

the token ring approach and satisfies the mentioned requirements in a way that minimize

the communication overhead and ensure deadlock freedom, ensure starvation freedom.

The competing nodes generate a token for the permission to enter CS. The token traverses the logical ring structure. A node can enter CS if and only if it receives back its generated token. The performance of the algorithm presented here will be evaluated using the total number of messages required for a node to enter the critical section as a criterion. Message traffic should be minimized in order to decrease the overhead in the communications network.

Q. 1. (e) Explain the Bully algorithm.

Ans. Bully Algorithm : Background : Any process Pi sends a message to the current coordinator: if no response in T time units, Pi tires to elect itself as leader. Details follows :

Algorithm for reprocess Pi that detected the lack of coordinator.

(i) Process Pi sends and "Election" message to every process with higher priority.

(ii) If no other process responds process Pi starts the coordinator code running and sends a message to all processes with lower priorities saying "Elected Pi".

(iii) Else Pi waits for T time units to hear from the new coordinator and if there is no response a start from (1) again.

Algorithm for other processes (also called Pi)

If Pi is not the coordinator then Pi may receive either of these messages from Pj.

if Pi sends "Elected Pj" [this message is only received if i < j]

Pi updates its records to say that Pj is the coordinator.

Else if Pj sends "election" message (i > j)

Pi sends a response to Pj saying it is alive

Pi starts an election.

Q. 1. (f) What are vector clocks? What are the advantage of vector clock over Lamport clock?

Ans. The concept of vector clock was proposed for deeping track of transitive dependencies among processes for recovery purpose. It gives us the ability to decide whether two events are causally or not by simply looking at their time stamps. The implementation rules for vector clocks are as follows :

(i) IR1 Clock Ci is incremented between any tow successive events in process Pi.

$$Ci [I] = Ci [I] + d \qquad (d > 0)$$

(ii) IR 2 if event a is the sending of message m but process Pi then message m is assigned a vector timestamp tm = ci (a), on receiving the same message m by process pj, cj is updated as follows:

For all k, cj [k] = max (cj [k], tm [k])

So by this rule this overcomes the limitation of Lamport logical clock i.e., if a and b are events in different processes and c (a) < c(b), then a – b is not necessarily true.

Q. 2. Attempt any two parts : **10 × 2 = 20**

Q. 2. (a) What are phantom Deadlocks ? Explain the algorithm which could detect phantom dead locks ?

Ans. A deadlock is a situation where in two more competing actions are waiting for the other to finish and thus neither ever does. It is often seen in a paradox like the chicken or the egg.

In the computing world deadlock refers to a specific condition when two or more processes are each waiting for another to release a resource, or more than two processes are waiting for resources in circular chain. Deadlock is a common problem in multiprocessing where many processes share a specific type of mutually exclusive resource known as a software, or soft, lock Computers intended for the time sharing and/or real-time markets are often equipped with a hardware lock (or hard lock) which guarantees exclusive access to processes forcing serialization. Deadlocks are particularly troubling because there is no general solution avoid (soft) deadlocks.

This situation may be likened to two people who are drawing diagrams with only one pencil and one ruler between them. If one person takes the pencil and the other takes the ruler, a deadlock occurs when the person with the pencil needs the ruler and person with the ruler needs the pencil before he can give up the ruler. Both recrests cannot be satisfied, so a deadlock occurs.

The telecommunication's description of deadlock is a little stronger deadlock occurs when none of the processes meet the condition to move to another state is described in the processe's finite state

machine and all the communication channels are empty. The second condition is often left out on other system but is important in the telecommunication context.

Q. 2. (b) Construct a solution to reliable totally ordered multicast in a synchronous system using a reliable multicast and a solution to the consensus problem.

Ans. In many distributed application all the processes that cooperate to achieve a common goal have to share a common view of the state of the system. To build seen a common view processes have to execute an agreement protocol during which can process proposes its own partial view and gets a final value which must be the same for every one. Among also the agreement problems (atomic commit. atomic broadcast electron, membership etc.,) the consensus problem is the simplest paradigm. Unfortunately it has been shown that the consensus problem has not deterministic solution in a purely asynchronous distributed system. To design correct agreement protocol one needs either to weaken the problem or strengthen the underlying system by adding synchrony assumptions.

The consensus problem is central paradigm of fault-tolerant distributes computing informally stated as follows. Each process of a set of processes proposes a value and each non-faulty process has to decide a value (termination) in such a way that a decided value is a proposed value (validity) and the non-faulty processes decides the same value (agreement). Uniform consensus is a stronger problem in the sense that requires that no two processes decide distinct values (uncifor n agreement). So uniform consensus prevents a faulty process from deciding differently from the non-faulty processes. Consensus is important for several reasons. Consensus is an abstraction in the agreement part of several fault-tolerant distributed computing problems. It constitutes a basic building block on top of which solutions to those problems can be designed. Consensus is also important because of its relation to problem tractability. It has been shown that some distributed agreement problem can be solved in some system (with a given fault model) if and only if, consensus can be solved in the same system (e.g. atomic broadcast and consensus are equivalent problems in asynchronous distributed systems serone to process crashes). Yet another justification of the importance of consensus in the principles and mechanisms that underlie the design of most consensus lies principles and mechanisms that underlie the design of most consensus protocols. Understanding those basic principles and mechanisms can provide system designers two have to copeverifying signatures which involves the user public key. the output of the signature process is called the "digital signature".

Digital signature, like written signatures, are used to provide authentication of the associated input usually called a "message". Messages may be anything, from electronic mail to a contract or even a massage sent in a more complicated cryptographic protocol. Digital signatures are used to create public key infrastructure (PKI) schemes in which a user's public key (whether for public-key encryption, digital signatures, or any other purposes) is tied to a user by digital identity certificate issued by certificated authority. PKI schemes attempt to unbreakably bind user information (name, address, phone number, etc.) to a public key, so that public keys can be used as a form of identification.

Digital signatures are often used to implement electronic signatures, a broader term that refers to any electronic data carrels the intent of a signature [1] but not all electronic signatures use digital signatures. In some countries including the United States, and in the European Union electronic signatories have legal significance. However, laws concerning electronic signatures do not always

make clear their applicability towards cryptographic digital signatures, leaving their legal importance some what unspecified.

These are common reasons for applying a digital signature to communications : Although messages may often include information about the entity sending a message, that information may not be accurate.

Although messages may often include information about the entity sending a message, that information may not be accurate. Digital signatures can be used to authenticate the sources of message. When ownership of a digital signature secret key is bound to a specific user a valid signature shows that the message was sent by that user. The importance of high confidence in sender authenticate is especially obvious in a financial context. For example suppose a bank's branch office sens instructions to the central office requesting a change int he balance or an account. If the central office is not convinced that such a message is truly sent from an authorized source, acting on such a request could be a grave mistake.

In many scenarios, the sender a receiver of a message may have a need for confidence that the message may have a need confidence that the message has not been altered during transmission. Although encryption hides the contents of a message it may be possible to change an encrypted message without understanding it. (Some encryption algorithms, known as noncallable ones prevent this, but other do not). However, if a message is digitally signed any change in the message will invalidate the signature. Further more there is no efficient way to modify a message and its signature to produce a new message cryptographic hash functions with a valid signature.

Q. 2. (c) What are Agreement Protocols? What are Agreement and validity objectives of Byzantine Agreement Problems?

Ans. In distributed system where sites often compete as well as cooperate to achieve a common goal it is often required that sites reach mutual agreement. The process of reaching an agreement is called a agreement protocol. When a system is free from failures an agreement can easily be reached among the processors.

In Byzantine agreement problem an arbitrary chosen processor, called the source processor broadcasts it initial value to all other processors. A solution to the Byzantien agreement problem should meet the following two objectives.

Agreement : All non -faulty processors agree on the same value.

Validity : If the source processor is non faulty then the common agreed upon value by all non faulty processors should be the initial value of the sources. The point should be noted :

(i) If the source processor is faulty then all non faulty processors can agree on any common value.

(ii) It irrelevant what value faulty processors agree on or whether they agree on a value at all.

Q. 3. Attempt ant two parts : **$10 \times 2 = 20$**

Q. 3. (a) What do you mean by the Distributed object model? Write a short note on Remote method invocations.

Ans. According to this model, each process contain a collection of objects some of which can receive both local and remote invocations where as the other objects can receive only local invocations. Method invocation between objects in different processes. Whether in the same computer

or not are known as remote method invocations. Method invocations between objects in the same process are local method invocations.

The following two fundamental concepts are the hearty of distributed object model Remote object reference

Remote interface

Java RMI extends the java object model to provide the support for distributed objects in java language. In particular it allows objects to invoke methods on remote objects using the same syntax as for local invocations. In addition type checking applies equally to remote invocations as to local ones. However an object making a remove invocation is aware that its target is remote because it must Handel remote exceptions.

Remote interfaces are defined by extending an interface called remote provided int he java. rmi package. The method must throw remote exceptions, but application specific exception may also be thrown.

In java RMI the parameters of a method are assumed to be input parameters and the result of a method is a single output parameter.

Q. 3. (b) What do you mean by Digital Signature ?

Ans. Digital Signature: digital signature or **digital signature scheme** is a type of asymmetric cryptography used to simulate the security properties of a handwritten signature on paper. Digital signature schemes normally give two algorithms, one for signing which involves the user's secret or private key, and one for verifying signatures which involves the user's public key. The output of the signature process is called the "digital signature."

A signature provides authentication of a "message". Messages may be anything, from electronic mail to a contract, or even a message sent in a more complicated cryptographic protocol. Digital signatures are used to create public key infrastructure (PKI) schemes in which a user's public key (whether for public-key encryption, digital signatures, or any other purpose) is tied to a user by a digital identity certificate issued by a certificate authority. PKI schemes attempt to unbreakably bind user information (name, address, phone number, etc.) to a public key, so that public keys can be used as a form of identification.

Digital signatures are often used to implement electronic signatures, a broader term that refers to any electronic data that carries the intent of a signature[1], but not all electronic signatures use digital signatures.[2][3][4] In some countries, including the United States, and in the European Union, electronic signatures have legal significance. However, laws concerning electronic signatures do not always make clear their applicability towards cryptographic digital signatures, leaving their legal importance somewhat unspecified.

A digital signature scheme typically consists of three algorithms:

- A key generation algorithm G that randomly produces a "key pair" (*PK*, *SK*) for the signer. *PK* is the verifying key, which is to be public, and *SK* is the signing key, to be kept private.
- A *signing* algorithm *S*, that on input of a message *m* and a signing key *SK*, produces a signature *ó*.
- A *signature verifying* algorithm *V*, that on input of a message *m*, a verifying key *PK* and a signature *ó*, either accepts or rejects.

Two main properties are required. First, signatures computed honestly should always verify. That is, V should accept (m, PK, S (m, SK)) where SK is the secret key related to PK, for any message m. Secondly, it should be hard for any adversary, knowing only PK, to create valid signature(s)

Q. 3. (c) Explain the different methods for generating and verifying signatures.

Ans. The different FSs use a subset of "the optimal list of features a distributed file system should have some say. Others (mainly developers) think they do have good reasons to use one implementation above the other : the efficiency of the technology is dependent on the intention of the use of it, its purpose. Therefore these page first deals with the general concepts of a distributed file system, afterwards the most relevant definitions are explained.

The reason why these pages are focused on (attempts to) distributed file systems is because I think they have the future. With the development of high speed LANs, WANs and the Internet, a stand alone machine is kind of outdated.

An important development towards a distributed FS was/is the separation between the File Server and the File Service. In short, the file server is the process that runs on a certain machine and takes care of the implementation of the file service. The file service can be described as the specification what exactly the file server can offer the client machines.

File Server : The file (directory) server defines the alphabet and syntax for the naming scheme of files and directories. The structure is with either trees or "grafen (ok, translate later). In the situation of a sharing system, the FS will have the same visual representation of each process. An important factor is the name transparency and location independence. Naming itself is on the two levels with the symbolic and binary name. The files service provides the operations to read/write certain file attributes the capability is a specification of the types of access rights (compare the ACL).

There are two methods for accessing a file on a file server :

Upload/download model: the whole file requested is downloaded to the client machine. read/changed and then sent back to the server.

Remote access model: the requested file resides on the server during client operations on that file.

Here's a nice example that you can't say one is better over the other. The determining factor for using either of the technology is the type of network laid. it isn't very user friendly to download a whole file over a slow WAN connection, or say a huge logistics database over a fast LAN. The performance per action may be a fraction slower when accessing a file lots of time via the remote access model, but you can not expect from a user to up/download e.g., a 5 MB power point presentation from his laptop with the 28K modem. With a read-a head mechanism the performance can be improved some what. The upload/download model is a good option in a (fast) LAN environment, and when the file is cached locally, access speed is highly reduced.

Q. 3. (d) What is the difference between a life service using the upload/download model and one using the remote access model? What is the difference between tree structured directory system from a general graph structured system?

Ans. Remote Access refers to any technology that enables you to connect users in geographically dispersed locations. This access is typically over some kind of dial-up connection, although it can include WAN connections.

Remote access by staff and other non-NHS organisations is a method of accessing files and systems that is becoming more common in the NHS. Often, critical business processes such as PACS

(Picture Archiving and Communications Systems) rely on easy and reliable access to corporate information systems. In practice, the benefits of securing remote access are considerable – business can be conducted remotely with confidence and sensitive corporate information remains confidential. This document sets out the policy for remote access and includes a set of common controls, which can be applied to reduce the risks associated with a remote access service.

Willful or negligent disregard of this policy will be investigated and may be treated as a disciplinary offence.

This policy covers all types of remote access, whether fixed or 'roving' including:

1.1. Travelling users (e.g. Staff working across sites or are temporarily based at other locations)

1.2. Home workers (e.g. IT support, Corporate Managers, IT development staff, Clinicians)

1.3. Non NHS staff (e.g. Social Services, contractors and other 3rd party organisations)

In providing remote access to staff, the following high-level principles will be applied:

1.4. The IT manager **[or enter appropriate officer title]** will be appointed to have overall responsibility for each remote access connection to ensure that the Trust's policy and standards are applied.

1.5. A formal risk analysis process will be conducted for each application to which remote access is granted to assess risks and identify controls needed to reduce risks to an acceptable level.

1.6. Remote users will be restricted to the minimum services and functions necessary to carry out their role.

Part-ii Tree structured directory system vs General graph structure system:

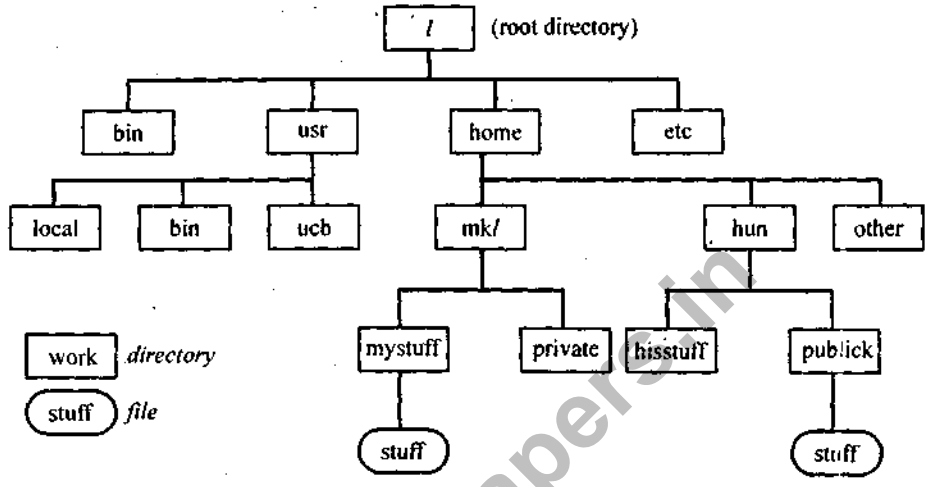
a **directory**, **catalog**, **folder** or **drawer** is an entity in a file system which contains a group of files and/or other directories. A typical file system may contain thousands (or even hundreds of thousands) of directories. Files are kept organized by storing related files in the same directory. A directory contained inside another directory is called a *subdirectory* of that directory. Together, the directories form a hierarchy, or tree structure.

A **hierarchy** is a system of ranking and organizing things or people, where each element of the system (except for the top element) is a subordinate to a single other element.

A hierarchy can link entities either directly or indirectly, and either vertically or horizontally. The only direct links in a hierarchy, insofar as they are hierarchical, are to one's immediate superior or to one of one's subordinates, although a system that is largely hierarchical can also incorporate other organizational patterns. Indirect hierarchical links can extend "vertically" upwards or downwards via multiple links in the same direction. All parts of the hierarchy which are not vertically linked to one another can nevertheless be "horizontally" linked by traveling up the hierarchy to find a common direct or indirect superior, and then down again. This is akin to two co-workers, neither of whom is the other's boss, but both of whose chains of command will eventually meet.

Tree-Structured Directories

- Efficient searching



- Grouping Capability
- Current directory (working directory)
- cd /spell/mail/prog
- **type** list
- **Absolute** or **relative** path name
- Creating a new file is done in current directory.
- Delete a file **rm** <file-name>
- Creating a new subdirectory is done in current directory.

  mkdir <dir-name>

  Example: if in current directory **/mail**

  mkdir count

Q. 4. Attempt any two parts : $10 \times 2 = 20$

Q. 4. (a) Describe how a non-recoverable situation could arise if write locks are released after the last operation of a transaction but before its commitment.

Ans. The DLM uses a generalized concept of resource, which is some entity to which shared access must be controlled. This may relate to file a record, or an area of shared memory but can be anything that application designer chooses. A hierarchy of resources may be defined so that number of levels of locking can be implemented. For instanced, a hypothetical database might define so that number of levels of locking can be implemented. For instance, a hypothetical database might define a resource hierarchy as follows :

- Database
- Table
- Record
- Field

A Process can then acquire locks on the database as a whole and then on particular parts of the database. A lock must be obtained on a parent resource before a subordinate resource can be looked.

Lock Modes : A process running within a VMS Cluster may obtain a lock on a resource. There are six lock modes that can be granted, and these determine the level of exclusivity of access to the resource. Once a lock has been granted it is possible to convert the lock to a higher or lower level o lock mode. when all processes have unlocked a resource, the system's information about the resource is destroyed.

Null lock (NL) : Indicates interest in the resource but does not prevent other processes from locking it. It has the advantage that the resource and its lock value block are preserved even when no processes are locking it.

Concurrent Read (CR) : Indicates a desire to read (but not update) the resource. It allows other processes to read or update the resource but prevents others from having exclusive access to it. this is usually employed on high-level resources in order that higher levels of lock can be obtained on subordinate resources.

Concurrent Write (CW) : Indicates a desire to read lock, which indicates a desire to read the resource. It also allows the processes to read or update the resource, but prevents other processes to read or update the resource, but prevents others from having exclusive access to it. This is aldos usually employed on high-level resources in order that higher levels of lock can be obtained on subordinate resources.

Protected Read (PR) : This is the traditional update lock, which indicates a desire to read the resource but prevents other from updating it. Others can however also read the resource

Protected Write (PW) : This is the traditional update lock which indicates a desire to read and update the resource and prevents others from updating it. Others with Concurrent tread access can however read the resource.

Exclusive (EX) : This is the traditional exclusive lock which allows read and update access to three resource, and prevents others from having any access to it.

Q. 4. (b) What are the advantages and drawbacks of multi version timestamp ordering in comparison with the ordering timestamp ordering.

Ans. Multi-version Timestamp Ordering is a transaction management protocol that relies on having multiple versions of an object to increase throughout. As in Time Warp there are two basic object types : transaction objects and data objects. The concept of virtual time is not used with MVTO though the concept of virtual time and an MVTO timestamp can be unified. MVTO has many advantages to the Time Warp protocol. Since there is no locking MVTO will be deadlock free. Higher concurrence levels are also possible. Where Time Warp relies on the rollback mechanism for database synchronization, MVTO uses abort-tetry to synchronize database access. Processing is done until an error is found at which time the offending transaction is required to abort and retry with a later timestamp. Communication between objects is done via message passing. Message in MVTO can be constructed like those of the Time Warp protocol. the idea behind MVTO is to make all operations occur in apparent timestamp prder even through messages may arrive out of timestamp order. Aborts occur because massage arrive out of timestamp order and the sequence they are in well cause a conflict.

The read/write ratio is defined to be the number of reads versus the number of writes in a transaction. This ratio was varied to provide three scenarios : a high read, low write scenario,; a balanced scenario where reads and writes were equal; and a low read high write scenario. In actuality read heavy transactions tend to be the most common kind of transaction [8]. Transaction size is also varied to study the effect of transaction length on concurrence and throughput in an open

model. transaction size is alternatively 12, 16 and 20 total read/writes. Al transactions in this model were two stage transactions to prevent possible deadlocks due to circular dependencies during MVTO commit which used the optimistic recovery protocol. Two stage transaction are characterized by placing all reads before write. While this may be an undue restriction to place on a programmer writing a transaction. in actuality the programmer is free to write the transaction interleaving the read and writes at will. The translation to a two phase transaction can occur at compile time.

Q. 4. (c) Explain the difference between lineraizability and sequential consistency and why the latter is more practical to implement in general?

Ans. In concurrent programming an operation is atomic or linearizable if it appears to take effect instantaneously. An atomic object can be understood immediately and completely from its sequential definition as a set of operations run in parallel will always appear to occur one after the other no inconsistence is may emerge.

Linearizability was first introduced as a consistency model by Herlihy and Wing in 1887. It encompasses more restrictive definitions of atomic such as "an atomic operation is one which cannot be (or is not) interrupted by concurrent operations", which are usually vague about when an operation is considered to begin and end.

Linearizability guarantees that the in variants of a system are observed and preserved by all operations : it all operations individually preserve an invariant the system as a whole will.

Those familiar with the ACID properties of databases should note that, in concurrent programming atomicity is an isolation level strictly stronger than serializable. Databases have a different definition of the term atomicity.

Sequential consistency is of the consistency models used in the domain of the concurrent programming (e.g. in distributed shared memory distributed transactions etc). It was first defend as the property that requires that ".... the results of any the processors were executed in some sequential order, and the operations of each individual processor appear in this sequence in the order specified by its program. The system provides sequential consistency if every node of the system sees the (write) operation son the same memory part (page virtual cell etc) in the same order, although the order may be different form the order as defined by real time (as observed by hypothetical external observation or global clock) of issuing the operations."

The sequential consistency is weaker than strict consistency (which would demand that operations are seen in order in which they were actually issued-which is essentially impossible to secure in distributed system where deciding global time in virtually impossible) and is the easiest consistency model to understand [citation needed] since a system preserving that model is behaving in a way expected by an average programmer.

The simplicity is achieved at cost of efficiency: distributed systems with sequential consistency model are without further optimization such as speculation, one magnitude slower than those providing weaker model.

Q. 5. Attempt any two parts : **10 × 2 = 20**

Q. 5. (a) What is Routing ? What is Destination based routing?

Ans. The network is the following illustration is more complex. It consists of a mesh of interconnected networks that provide multiple routing paths. While each router may have its own attached networks and only two hosts are shown for simplicity. Host A wants to send a packet to host B. Multiple paths exist. Host A sends the packet to its locally attached router and lets the router handle the forwarding.

Routers are responsible for determining the next hop that will get a packet to its destination not the complete path to the destination. Basic IP routing is called hop-by-hop or destination-based routing. The technique is like getting directions-a person may point you in the right direction at an intersection. At the next intersection another person points you in the right direction. Eventually, you get to where you wants to go not by knowing the exact path from the start, but b y being pointed a long the way as you go. Someone might point you in a direction that avoids construction or dead ends. On a large meshed network with many possible paths routers may choose paths that avoid congested or temporarily disabled links.

The numeric IP addresses of networks hosts and routers are replaced with abbreviations.

(i) At the source (A1), a datagram is created with the IP address of the destination host (C1).

Since the distinction network address is not the same as the current network address, host A1 forwards the datagram directly to the default gateway router A/B. Note that the datagram is put into one or more frames that have the MAC address of router A/b.

The frame arrives at router A/b on port A. The datagram is extracted and the IP address is inspected. The router determines that the destination can be reached through router B/C so it puts the datagram in a frame type to match network B and attaches the MAC address of router B/C.

At router B/C the frame arrives on port B. The datagram is extracted and the IP address is inspected. The router determines that the host is attached to subnet C, so i does a table lookup or uses ARP to resolve the IP address into a MAC address. The router then puts the datagram in a frame attaches the MAC address of destination CI, and transmits the frame on the network.

Host CI sees the frame on the network as being addressed to it and receives the frame.

To transmit large files the file is broken into pieces that are put inside numerous packets. Then in one of the packets is lost only that packet must be re-transmitted and out the entire file.This reliability feature is handled by TCP in the background.

Q. 5. (b) What is Election ? Suppose that two processes detect the demise of the coordinator simultaneously and both decide to hold an election using the bully algorithm. What happens?

Ans. The coordinator election problem is to choose a process from among a group of process on different processors in a distributed system to act as the central coordinator.

An election algorithm is an algorithm for solving the coordinator election problem. By the nature of the coordinator election problem, any election algorithm must be a distributed algorithm.

- a group of processes on different machines need to choose a coordinator.
- peer to peer communication : every process can send messages to every other process.

Assume that processes have unique IDS, such that one is highest.

(a) Bully Algorithm : Background : Any process Pi sends a message to the coordinator : if no response in T time unites Pi tries to elect itself as leader. Details follows :

(i) Process Pi sends an "Election" message to every process with higher priority.

(ii) If no other process responds, process Pi starts the coordinator code running and sends a message to all processes with lower priorities saying "Elected Pi".

(iii) Else, Pi waits for T time units to hear from the new coordinator and if there is no response a start from step (1) again.

Algorithm for other processes (also called Pi).

If Pi is not the coordinator then Pi may receive either of these messages from Pj if Pi sends "Elected Pj" : {this message is only received if i < j]

Pi updates its records to say that Pj is the coordinator.

Else if Pj sends "election" message (i > j) Pi sens a response to Pj saying it is alive Pi starts an election.

(b) Election In A Ring = Ring Algorithm

Assume that processes from a ring each process only sends messages to next process in the ring.

Active list : its info on all other active processes.

Assumption : message continues around the ring even if a process along the way has crashed.

Background : any process Pi sens a massage to the current coordinator ; if no response in T tiem units, Pi initiates an election.

(i) Initialize active list to empty.

(ii) Send an "Elect (i)" message to the right. + add i to active list.

If a process receives an "Elect (j)" message

(a) This is the first message sent or seen Initialize its active to [ij]; send "elect (i)" + send "Elect (i)"

(b) if i! = j add i to active list + forward "Elect (j)" message to active list

(c) otherwise (i = j), so process i has complete set of active processes in its active list.

Choose highest process ID + send "Elected (x)" message to neighbor. If a process receives "Elected (x)" message.

Set coordinator to x

Example:

Suppose that we have four processes arranged in

arraign : P1 a P2 a P3 a P4 a P1 ...

P4 is coordinator

Suppose P1 + P4 crash

Suppose P2 detects that coordinator P4 is not responding

P2 sens active list to []

P2 sends "Elect (2)" message to P3; P2 sets active lis [2]

P3 receives "Elect (2)"

This message is the first message seen so P3 sets its active list to [2, 3]

P3 sends "Elect (3)" towards P4 and then sends

"Elect(2)" towards P4

The message pass P4 + P1 and then reach P2

P2 adds 3 to active list [2, 3]

P2 forwards "Elect (3)" to P3

P2 receives the "Elect(2) message

P2 chooses P3 as the highest process in its list [2, 3]

and sends an "Elected (P3)" message

P3 receives the "Elected (3)" message

P3 receives the "Elect (3)" message

P3 chooses P3 as the highest process in its list [2, 3] + "Elected (P3)" message

Q. 5. (c) Write shorts notes on :

(i) ARP problem (ii) Deadlock free Packet Switching

Ans. (i) If you follow the instructions and setup the examples in the LVS-mini-HOWTO, then you don't need to know about the arp problem. If you're going to setup grander LVS's, then you'll need to understand the arp problem.

Although this section comes early in the HOWTO, it has lots of pitfalls. You shouldn't be reading this unless you've at least setup a working LVS-NAT and LVS-DR LVS using the canned instructions in the LVS-mini-HOWTO.

The LVS allows several machines to function as one machine. For LVS-DR and LVS-Tun, some trickery was needed to split the various handshakes involved in establishing and maintaining a tcpip connection, so that some parts of the handshake come from one machine and other parts from another machine. The worst problem, which ironically only happens with realservers running Linux (2.2 and later kernels), is the "arp problem". It's just as well we have the source code :-(.

With LVS-DR and LVS-Tun, all the machines (director, realservers) in the LVS have an extra IP, the VIP. Here's a LVS-DR in a test setup where all machines and IPs are on the same physical network (*i.e.* are using the same link layer and can hear each other's broadcasts).

When the client requests a connection to the VIP, it must connect to the VIP on the director and not to the VIP on the realservers.

The director acts as a layer-4 IP router, accepting packets destined for the VIP and then sending them on to a realserver (where the real work is done and a reply is generated). For the LVS to function, when the client (or if present, the router) puts out the arp request "who has VIP, tell client", the client/router must receive the MAC address of the director (and not the MAC address of one of the realservers). After receiving the arp reply, the client will send the connect request to the director. (The director will update its **ipvsadm** tables to keep track of the connections that it's in charge of and then forward the connect request packet to the chosen realserver).

If instead, the client gets the MAC address of one of the realservers, then the packets will be sent directly to that realserver, bypassing the LVS action of the director. If nothing is done to direct arp requests, by the router for the VIP, to the director (arp bouncing), then in some setups, one particular realserver's MAC address will be in the client/router's arp table for the VIP and the client will only see one realserver. If the client's packets are consistently sent to the same realserver, then the client will have a normal session connected to that realserver. You can't count on this happening: in the middle of a tcpip sesssion, the client/router might get the MAC address of another realserver as a result of an arp request, and the client will start getting packets for connections it knows nothing about (and the realserver will send tcp resets). (In my setup, the machine with the fastest CPU is in the client's arp table, suggesting that it's the first machine to reply that gets in. Horms and Steven WIlliams have written that they think it's the last machine to reply whose entry in in the client's arp table.) In other setups where the realservers are identical, the client will connect to different realservers each time the arp cache times out (see comment by Steven WIlliams elsewhere). If the director always gets its MAC address in the router arp table, then the LVS will work without any changes to the realservers (as happened in my case with a director with the fastest CPU in the LVS), although this is not a reliable solution for production.

Getting the MAC address of the VIP on the director (instead of the MAC address of the VIP on the realservers) to the client when the client/router does an arp request is the key to solving the "arp problem"

The traditional ways of handling the arp problem (as explained here) all require fiddling with the settings of the VIP on the realservers. The assumption in the early days of LVS was that you wouldn't have access to the router (this being under the control of the IT department or your ISP and you would have to go through a lot of bureaucracy to changed the settings on the router). However if you're paying good money to an ISP to house your LVS, or your in house LVS is doing something useful for your establishment, then you should have no trouble in having the router setup the way you want.

If you have access to the router (or can put one in front of your LVS - a low power linux box is just fine) and you can set it to route packets for the VIP only to the director(s) and not to the realservers, or you can use the arp filtering tools of `iptables`, and you understand what's been said above, then you've handled the arp problem and need read no futher.

For those who don't have access to the router, or who want to setup an LVS on one network, then read on...

The arp problem is handled in Linux 2.0.x kernels, as dummy0, tunl0, lo:0, are available on the realserver which don't reply to arp requests. For other OS's, the NOARP flag for ifconfig stops the VIP on the realservers from replying to arp requests.

(ii) Deadlock is one of the most serous system failure that can occur in a computer system or a network. Deadlock states have been observed in existing computer networks emphasizing the need of carefully designed flow control procedures (controllers)Sch a deadlock free controller is readily found if we allow it global information about the overall network state. Generally this assumption is not realistic and we must resort to deadlock free local controllers using only packet and node information. We present her several types of such controllers; we study their relationship and give a proof of their optimality with respect to deadlock re controllers using the same set of local parameters.

A method an apparatus for establishing deadlock free routing in a bi-directional multistage, interconnected, cross-point based packet switch particularly, though not exclusively employed within a high speed packet network of a massively parallel processing system. Specifically a group of sets of restricted routes traversing a source, intermediated and destination switch chip are determined by establishing a number of route restrictions from each source switch in the network and determining a number of routes restricted between each source destination pair of switch chips there in such that the standard deviation for the number of routes left unrestricted between all source destination pairs of switch chips for the packet network is minimized. The group of sets for restrictions is created b y analyzing a first portion of the network to determine deadlock free route restrictions that comply with the established per with restrictions and the determined source-destination pari restrictions therefore and then incrementally adding each remaining switch chip for the network and repeating the analysis. Any number of sets from the resultant group of sets of route restrictions may be implemented within the network in accordance with determined link usage and intermediate switch chip usage balancing techniques.