UNIT - 1

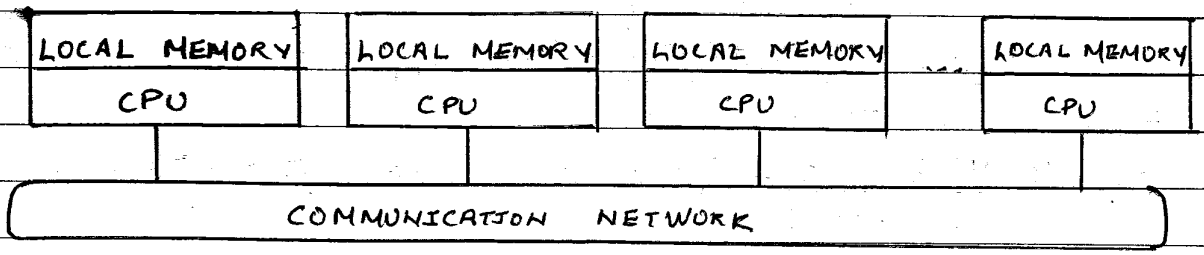
Date ____/____/____
Page _____

INTRODUCTION TO DISTRIBUTED SYSTEMS

(1) Distributed Systems -

It is a collection of independent computers that appear to the users of the system as a single computer.

(2) Architecture for Distributed System -



Distributed computing system (loosely coupled systems) is a collection of processors (nodes) interconnected by a communication network in which each processor has its own local memory and other peripherals, and the communication between any two processors of the system takes place by message passing over the communication network.

(3) Goals of Distributed System -

(1) Making Resources Accessible for the users (and applications) to access remote resources, and to share them in a controlled ~~way~~ and efficient way.

(2) Distribution Transparency that is to hide the fact that its processes and resources are physically distributed across multiple computers.

(3) Openness or open distributed system is a system that offers services according to standard rules that describe the syntax and semantics of those services.

(4) Scalability with respect to its size (more users & resources), geographically and administratively.

Date ___/___/___
Page _____

③ Advantages of Distributed Systems -

over centralized systems

(1) Economics - Microprocessors offer a better price/performance than mainframes

(2) Speed - A distributed system may have more total computing power than a mainframe

(3) Inherent Distribution - Some applications involve spatially separated machines

(4) Reliability - If one machine crashes, the system as a whole can still survive.

(5) Incremental Growth - Computing power can be added in small increments

over isolated (personal) computers

(6) Data Sharing - Allow many users access to a common data base

(7) Device Sharing - Allow many users to share expensive peripherals

(8) Communication - Make human-to-human communication easier

(9) Flexibility - Spread the work load over the available machines in the most cost effective way.

④ Disadvantages of Distributed Systems -

(1) Software - little software exists at present for distributed systems.

(2) Networking - The network can saturate or cause other problems

(3) Security - Easy access also applies to secret data.

⑤ Hardware Concepts -

All distributed systems are MIMD (multiple instruction streams and multiple data streams) but not all MIMD are distributed systems.

MIMD computers are divided into two groups -

(1) Multiprocessors (have shared memory)

(2) Multicomputers (have private memory)

Date ___/___/___
Page _____

MIMD

PARALLEL AND DISTRIBUTED COMPUTERS

Tightly Coupled — Loosely Coupled

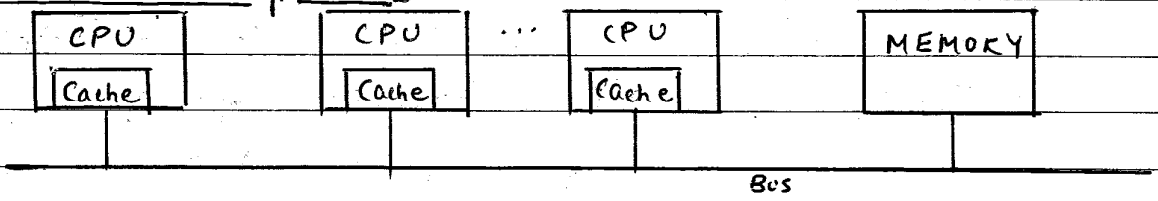
MULTIPROCESSOR (shared memory) — MULTICOMPUTERS (private memory)

Divided based on interconnection network

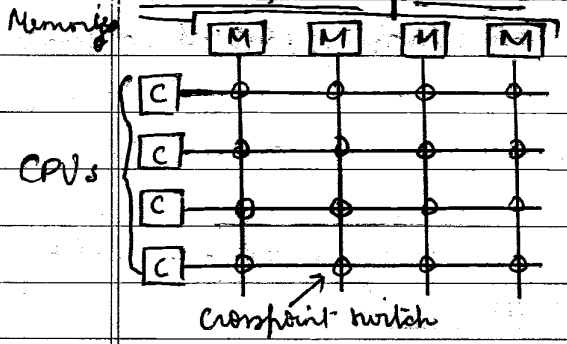
BUS — SWITCH — BUS — SWITCH

Sequent, Encore — Ultracomputer, RP3 — Workstations on a LAN — Hypercube, Transputer

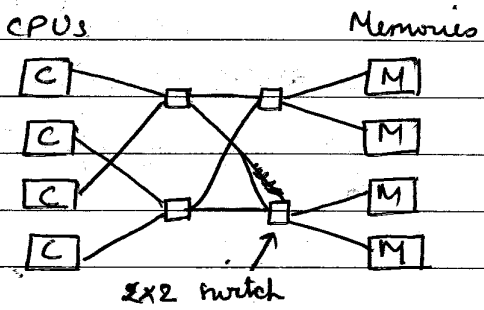
Bus-based multiprocessors -



Switched Multiprocessors -

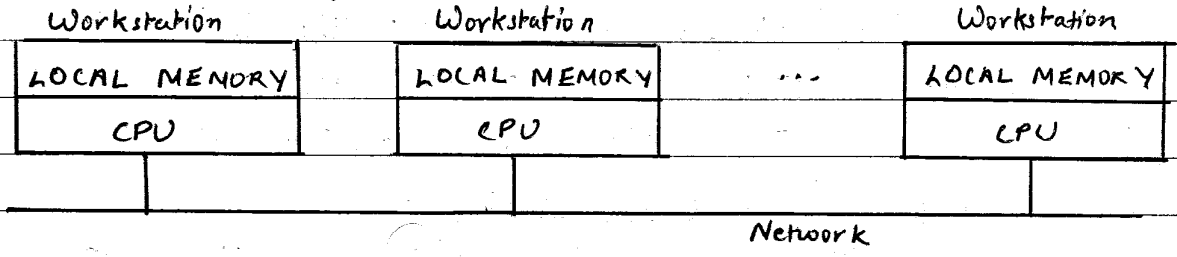


CROSSBAR SWITCH



OMEGA SWITCHING NETWORK

Bus-based microcomputers -



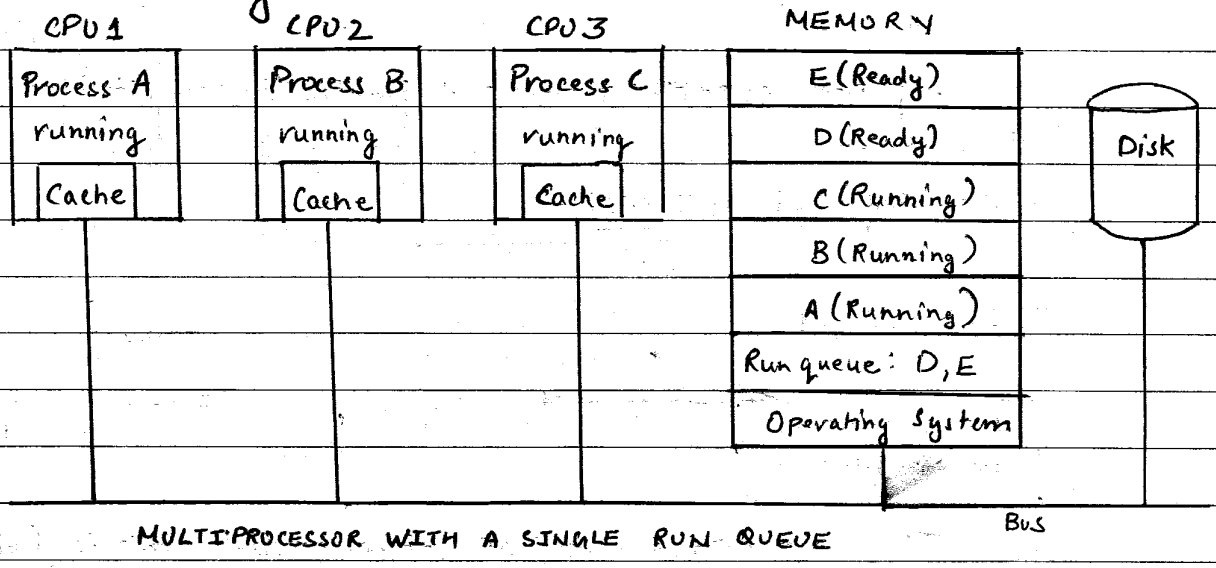
A multicomputer consisting of workstation on a LAN

Date ___/___/___
Page _____

Multiprocessor Timesharing Systems –

Tightly coupled software on tightly coupled hardware. Key characteristic is the existence of a single run queue that is a list of all the processes in the system that are logically unblocked and ready to run. The run queue is a data structure kept in the shared memory.

Eg –

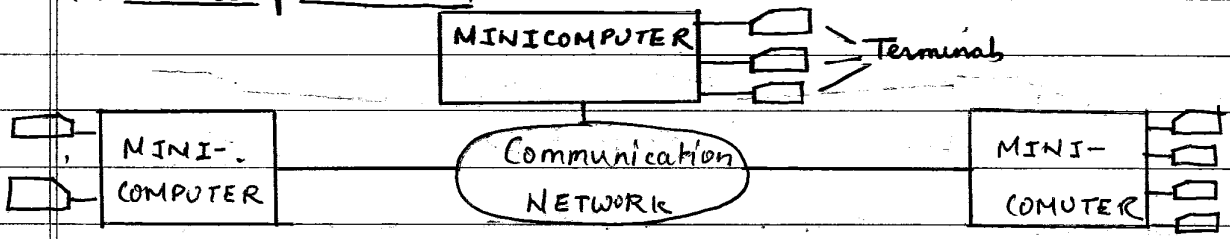


MULTIPROCESSOR WITH A SINGLE RUN QUEUE

(7) Distributed Computing Model –

It is classified into five categories –

(1) Minicomputer Model –



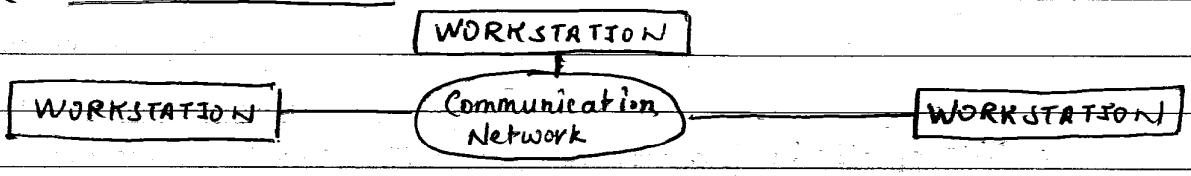
It is a simple extension of the centralized timesharing system. Each user can remotely access to other minicomputers.

It is basically used for resource sharing that are available on some machine other than the one on to which the user is currently logged.

Several terminals are used for multiple user simultaneously logged on to a single minicomputer.

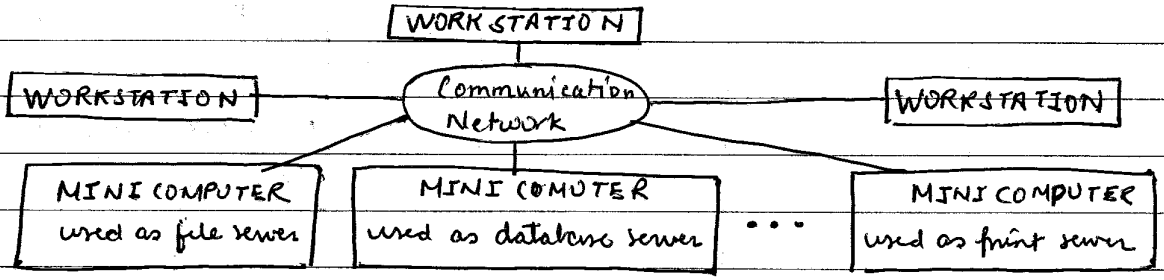
Eg:- Early ARPAnet.

(2) Workstation model -



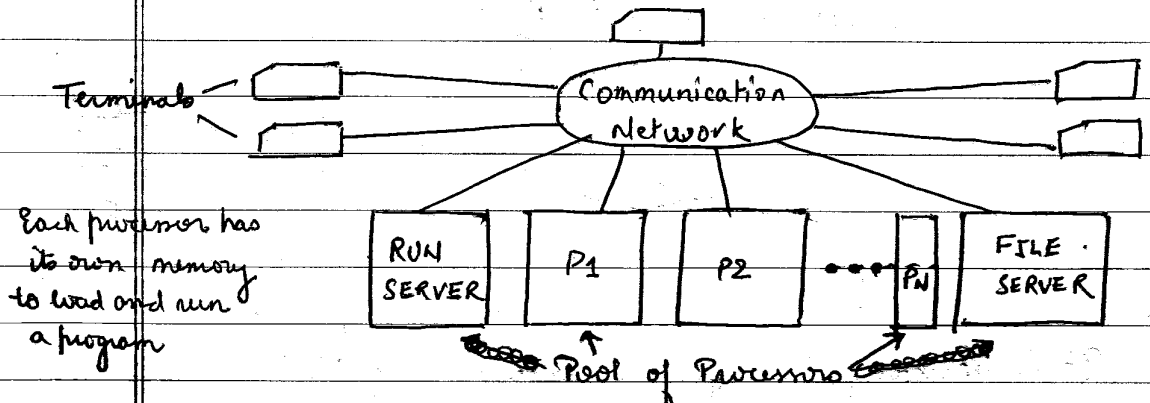
When the user's workstation doesn't have sufficient processing power, it transfers one or more of the processes from the user's workstation to some other workstation that is currently idle and gets the process executed there, and finally the result of execution is returned to the user's workstation. eg - Sprite system.

(3) Workstation-Server Model -



Those workstations that do not have disk that is diskless workstations uses minicomputers for different types of services
eg - V-system.

(4) Processor-Pool Model -



When a very large amount of computing power is needed in short time then in this model, the processors are pooled together to be shared by the users as needed.

Run Server - allocates & manages the processors in the pool to different users on a demand basis.

eg - Amoeba, Plan 9

UNIT- 2

Date ___/___/___
Page _____

DISTRIBUTED SHARE MEMORY & DISTRIBUTED FILE SYSTEM

① Basic concept of Distributed Share Memory (DSM) -

DSM system is a resource management component of a distributed operating system that implements the shared memory model in distributed system, which have no physical shared memory. The shared memory model provides a virtual address space that is shared among all nodes in a distributed system.

§ DSM is also refered as Distributed Shared Virtual Memory (DSVM).

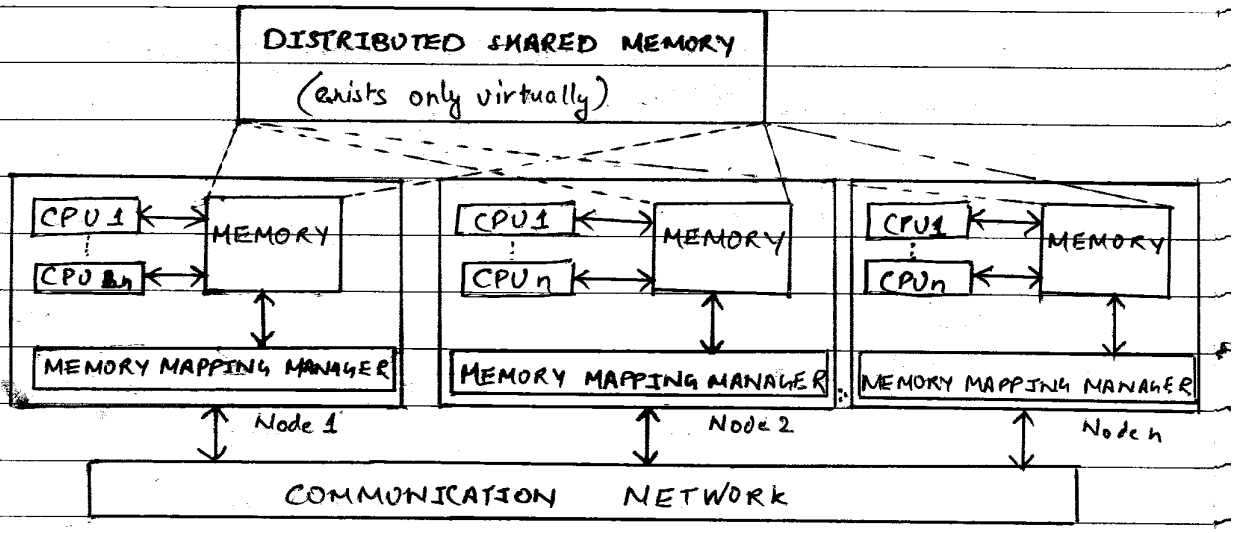
Advantages -

(1) Shields programmer from send/receive primitives
(2) Large virtual memory space.
(3) Single address space
(4) Simple software interfaces.
(5) Programs portable

Disadvantages -

(1) May incur a performance penalty
(2) No protection against shared data.

② DSM Architecture and its types -



DSM ARCHITECTURE

Date ___/___/___
Page _____

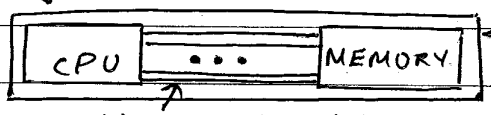
~~Each node~~ A simple message-passing system allows processes on different nodes to exchange messages with each other.

A software memory-mapping manager routine in each node maps the local memory onto the shared virtual memory. To facilitate the mapping operation, the shared-memory space is partitioned into blocks.

Data caching is used in DSM systems to reduce network latency.

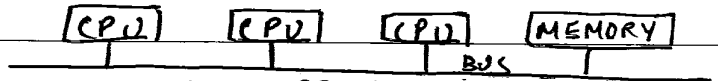
→ Types of DSM -

(1) On-chip memory - Several processors & a shared memory are on the same chip.



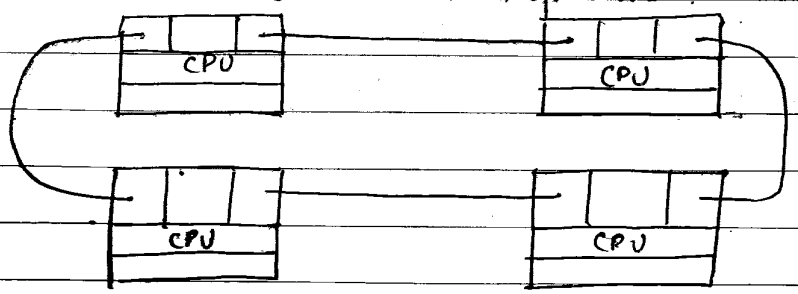
SINGLE CHIP COMPUTER

(2) Bus-based systems - CPUs and memories are connected to a BUS. CPUs either have or do not have local memories.



MULTI PROCESSOR

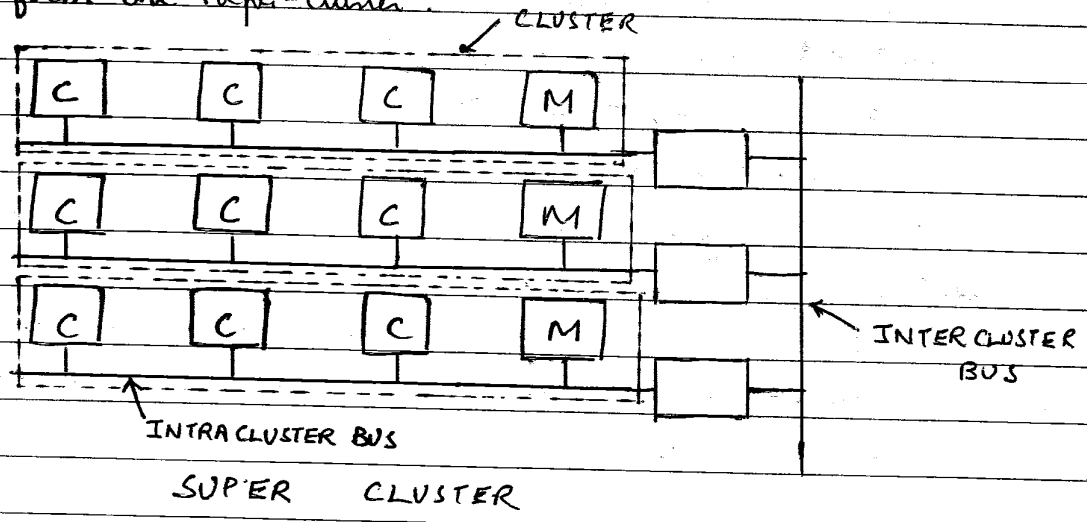
(3) Ring-based multiprocessors - Single address space is divided into a private part and a ~~public~~ shared part. The private part is divided up into regions so that each machine has a piece for its stacks and other unshared data and code. The shared part is common to all machines and is kept consistent by a hardware protocol roughly similar to those used on bus-based multiprocessors.



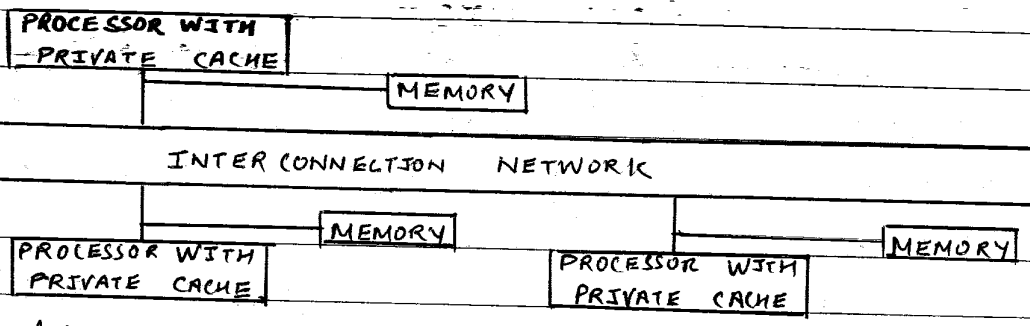
Ring based multiprocessor (Memnet Ring)

Memnet ring is a modified token-passing ring which has 20 parallel wires (16 data bits + 4 control bits) to sent bits at every 100 nsec for a data rate of 160 Mbps. Shared memory is divided into 32-byte blo

(4) Switched Multiprocessor - Three clusters connected by an inter-cluster bus to form one super-cluster.



(5) NUMA (Non-Uniform Memory Access) multiprocessors -



A processor's own internal computations can be done in its own local memory leading to reduced memory contention -

Accessing remote memory is possible but slower than accessing local memory

Remote access times are not hidden by caching.

(3) Design and Implementation Issues in DSM System -

(1) Granularity - When a nonlocal memory word is referenced, a chunk of memory containing the word is fetched from its current location and put on the machine making the reference. An important design issue is how big should the chunk would be?

A word, block, page, or segment (multiple pages).

(2) Structured of shared memory space - depends on the type of application that the DSM system intended to support.
(3) Data location and access - to share data in a DSM system
(4) Replacement strategy - Data block of local memory must be replaced by a new data block.
(5) Thrashing - Data blocks migrate between nodes on demand.

→ Challenges in DSM -
(1) How to keep track of the location of remote data?
(2) How to overcome the communication delays & high overhead associated with the references to remote data?
(3) How to allow "controlled" concurrent access to shared data?

④ Structure of shared memory space -
Three commonly used approaches for structuring -
(1) No structuring - linear array of words, simple & easy to design. Eg - IVY.
(2) Structuring by data type - Collection of objects or variables in a source language.
(3) Structuring as a database
→ Shared memory space is ordered as an associative memory, called a tuples space, which is a collection of tuples with data items in their fields.

⑤ Consistency Model -
It determines when the data updates are propagated and what level of inconsistency is acceptable. Models are as follows -
(1) Strict Consistency -
A shared memory system is said to support the strict consistency model if the value returned by the read operation on a memory address is always the same as the value written by the most recent write operation to that address.
Not suitable in distributed systems. Difficult to achieve in real systems as network delays can be variable.

(2) Sequential Consistency - (Strongest memory model for DSM)

A DSM system is said to be sequentially consistent if for any execution there is some interleaving of the series of operations issued by all the processes that satisfies the following two criteria -

(i) SC1 - The interleaved sequence of operations is such that if occurs in the sequence, then either the last write operation that occurs before it in the interleaved sequence is, or no write operation occurs before it.

(ii) SC2 - The order of operations in the interleaving is consistent with the program order in which each individual client executed them.

It provides single-copy semantics because all processes sharing a memory location always see exactly the same contents stored in it.

(3) Linearizability -

The result of any execution is the same as if the operations by all processes on the data were executed in some total order.

→ Bring in servers view to define the ordering of concurrent events

→ Real time actions performed on the servers.

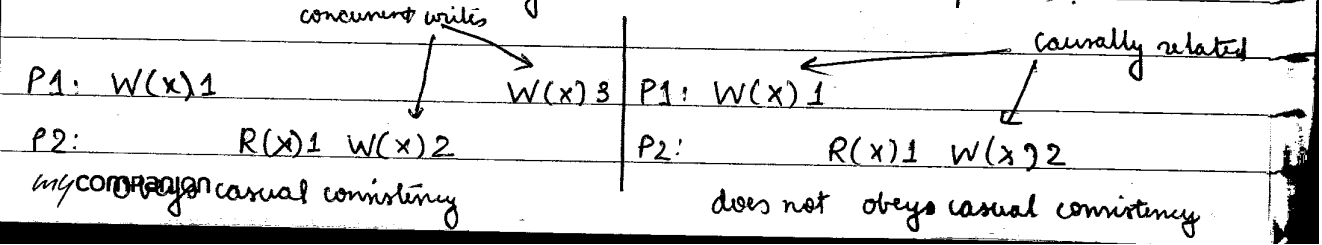
→ Non-overlapping request.

→ Overlapping request - Enqueuing times of the requests are in different orders on different servers can have arbitary order, but sequentially consistent.

(4) Causal consistency -

Operations that are causally related must be seen by all processes in the same corresponding order. Concurrent writes from different processes do not have any causal relationship and can be seen in different order by different processes. There is no need to write exclusively, cheaper write operation.

Need to keep track of dependency relations in order to determine whether two events are causally related. This model is expensive.



any comparison casual consistency

does not obeys casual consistency

(5) Pipelined RAM Consistency –

It can be implemented by simply sequencing the write operations performed at each node ~~to~~ independently of the write operations performed on other nodes.

It is simple and easy to implement and also has good performance.

(6) Weak Consistency –

It has three properties –

(1) Accesses to synchronization variable are sequentially consistent.

(2) No access to a synchronization variable is allowed to be performed until all previous writes have completed everywhere.

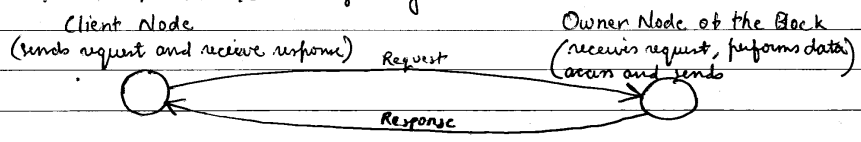
(3) No data access (write or read) is allowed to be performed until all previous accesses to synchronization variables have been performed.

Weak consistency requires the programmer to use locks to ensure reads and writes are done in the proper order for data that needs it.

→ Implementing sequential consistency model –

There are different types of replication & migration techniques –

(1) Non Replicated, Non Migrating Blocks (NRNMBs)



Characteristics –

(1) Single copy of each block in the entire system.

(2) location of a block never changes.

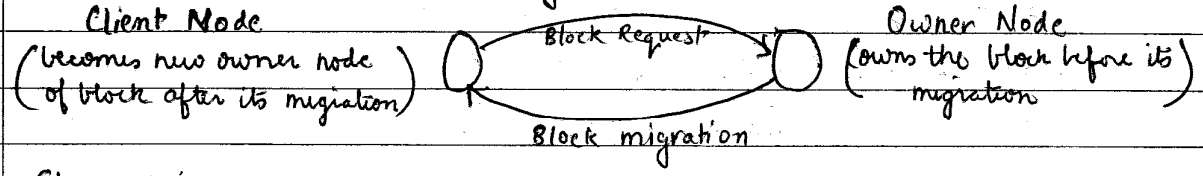
Drawbacks –

(1) Serializing data access creates a bottleneck.

(2) Parallelism is not possible.

This method is simple and easy to implement.

(2) Non Replicated Migrating Blocks (NRMBs) -



Characteristics -

(i) No communication cost incurred when a process access data currently held locally.

(ii) Allows applications to take advantage of data access locally.

Drawbacks -

(i) Prone to thrashing problem (poor performance)

(ii) Parallelism is not possible.

(3) Replicated Migrating Blocks (RMBs) -

Characteristics -

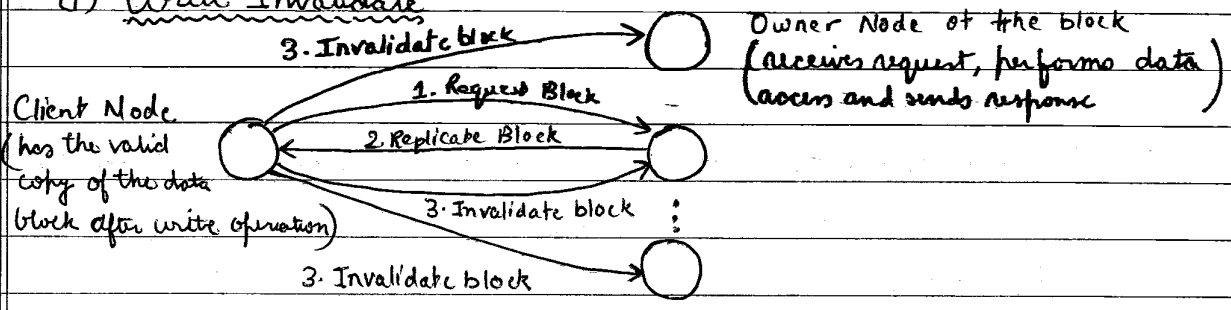
(i) Parallelism is possible due to application of blocks.

Drawbacks -

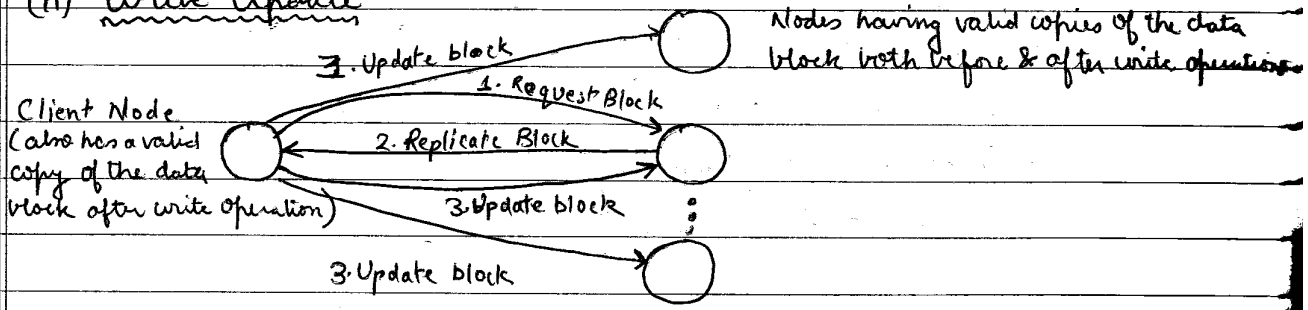
(i) Increases the cost of write operation to be perform in all replicas.

→ Two protocols is used for ensuring sequential consistency -

(i) Write Invalidate -



(ii) Write Update -



(4) Replicated Non-migrating Blocks (RNMBs) -
→ Replication of Blocks.
→ location of Replica is fixed.
→ Write-update protocol is used.
→ Sequential consistency is ensured by using a Global-sequencer.

(6) Thrashing -

It occurs when network resources are exhausted, and more time is spend invalidating data and sending updates than is used doing actual work.

Two or more process try to write the same shared block.

The larger the a block, the more chances of false sharing (i.e. it occurs when two different process access two unrelated variables that reside in the same data block) that causes thrashing.

Solutions -
(1) Allow a process to prevent a block from accessed from the others, using a lock.
(2) Allow a process to hold a block for a certain amount of time.
(3) Apply a different coherence algorithm to each block.

DISTRIBUTED FILE SYSTEM -

(1) Distributed file system is a resource management component of a distributed operating system. It provides a user with a unified view of the files on the network. A machine that holds the shared files is called a server. A machine that accesses the files is called a client.

Goals of distributed file systems are -
(1) Network transparency
(2) High availability.

Date ___/___/___
Page ___

(2) Desirable features of good distributed file system -

(1) Transparency -

(i) Structure Transparency - Client should not know the number or locations of the file server and storage devices.

(ii) Access Transparency - Both local and remote file should be accessible in the same way.

(iii) Naming Transparency - Name of the file should give no hint as to where the file is located.

(iv) Replication Transparency - Clients do not need to know the existence or locations of multiple file copies.

(2) User mobility -

User should not force to work on a specific node but should have the flexibility to work on different nodes at different times.

(3) Performance -

Average amount of time needed to satisfy client request

(4) Scalability

(5) High availability

(6) High reliability

(7) Security.

(3) File models -

File model are based on following criteria -

(1) Unstructured and structured files -

Simple file model - File is an unstructured sequences of data. There is no substructure known to the file server. Eg - UNIX, MSDOS

Structured file model - A file appears to the file server as an ordered sequence of records. They are two types -

Indexed records - Records have one or more key fields and can be addressed by specifying the values of the key fields. Eg - B-tree, RSS, Oracle.

Non-Indexed records - A file record is accessed by specifying its position within the file. Eg - IBM mainframe.

Date __/__/__
Page _____

*****

Most of the modern operating system uses the unstructured file model. This is mainly because sharing of a file by different applications is easier with the unstructured file model as compared to the structured file model.

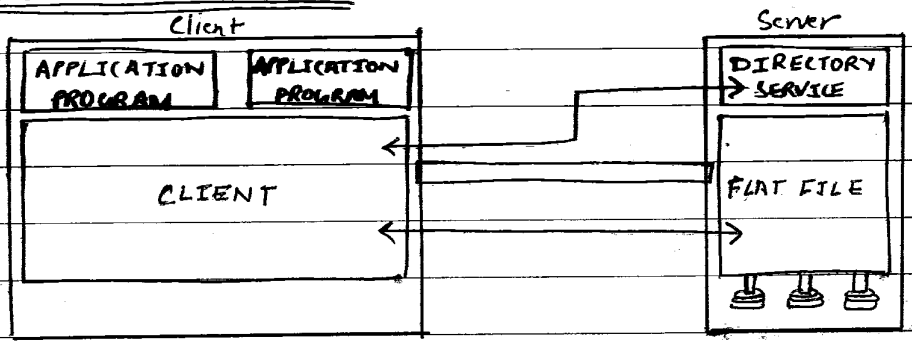
(2) Mutable and Immutable files -

Mutable files - An update is performed on a file overwrites on its old contents to produce the new contents. File is represented as a single stored sequence that is altered by each update operation.

Immutable files - A file cannot be modified once it has been created except to be deleted. File versioning approach is used. Eg - Cedar file system.

Problems - Increased use of disk space and disk allocation activity.

(4) File Service Architecture -



There are three components -

(1) Flat file Service -

Concerned with the implementation of operations on the contents of file. Unique File Identifiers (UFIDs) are used to refer to files in all requests for flat file service operations.

(2) Directory Service -

It provides mapping between text names for the files and their UFIDS. Client may obtain the UFID of a file by quoting its text name to directory service.

(3) Client module -

It provides integrated service (flat file & directory) as a single API to application programs. It holds information about the network

location of flat-file and directory server processes; and achieves better performance through implementation of a cache of recently used file blocks at the client.

→ Flat file service operations -

(1) Read (FileId, i, n) → Data - throws BadPosition -

If $1 \leq i \leq$ Length(File): Reads a sequence of up to n items from a file starting at item i and returns it in Data.

(2) Write (FileId, i, Data) - throws BadPosition -

If $1 \leq i \leq$ Length(File)+1: Write a sequence of Data to a file, starting at item i, extending the file is necessary.

(3) Create () → FileId -

Creates a new file of length 0 and delivers a UFID for it.

(4) Delete (FileId) - Removes the file from the file store

(5) GetAttributes (FileId) → Attr - Returns the file attributes for the file

(6) SetAttributes (FileId, Attr) - Set the file attributes

→ Directory service operations -

(1) Lookup (Dir, Name) → FileId - throws NotFound -

locates the text name in the directory and returns the relevant UFID.

(2) AddName (Dir, Name, FileId) - throws NameDuplicate -

If Name is not in the directory, adds (Name, File) to the directory and updates the file's attribute record.

(3) UnName (Dir, Name) - throws NotFound -

If Name is in the directory, removes the entry containing Name from directory

(4) GetNames (Dir, Pattern) → NameSeq -

Returns all the text names in the directory that match the regular expression ~~pattern~~ Pattern.

→ UNIX file system uses access control, hierarchy file system and file group (it is a collection of files that can be located on any server or moved between servers while maintaining the same names).

Date __/__/__
Page _____

⑤ File Accessing Models –

It depends on two factors –

(1) The method used for accessing remote files (2) The Unit of data access.

→ Accessing Remote files – Two models –

(1) Remote Service model –

Clients request for the file access is delivered to the server, the server machine performs the access request, and finally the result is forwarded back to the client. Request are transferred as messages.

Merit – A simple implementation Demerit – Communication Overhead

(2) Data Caching Model –

If the data needed to satisfy the client's access request is not present locally, it is copied from the server's node to the client node and is cached there.

Merit – Reducing network traffic Demerit – Cache consistency problem.

→ Unit of Data Transfer – Four models –

(1) File level transfer model – Complete file is moved. Eg – AFS, Amoeba

Merits – Simple, less communication overhead and immune to server

Demerits – A client required to have large storage space. Eg –

(2) Block level transfer model – Units of file blocks are moved Eg – NFS

Merits – A client not required to have large storage space.

Demerits – More network traffic / overhead

(3) Byte level transfer model – Units of bytes are moved. Eg – Cambridge file server

Merits – Flexibility maximized

Demerits – Difficult cache management to handle the variable-length data

(4) Record level transfer model – Units of records are moved.

Merits – Handling structured and indexed files

Demerits – More network traffic and more overhead to re-construct of a file

⑥ File Sharing Semantics -

Define when modifications of the file data made by a user are observable by other users.

(1) UNIX Semantics -

A file is associated with a single physical image that is accessed as an exclusive resource. Contention for this single image causes delays in user process. It is used in centralized and single processor systems.

Unix file system implements -

(1) Write to an open file visible immediately to other users of the same open file.
(2) Sharing file pointer to allow multiple users to read and write concurrently.

(2) Session Semantics -

No changes are visible to other processes until the file is closed. A file can be associated with multiple views. Almost no constraints are imposed on scheduling accesses. No user is delayed in reading or writing their personal copy of the file. Eg - Andrew File System (AFS).

(3) Immutable Shared file semantics -

No updates are possible; simplifies sharing and replication.

(4) Transactions like semantics -

All changes occur atomically. Begin transactions, perform operations and end transaction. The final file content is the same as if all transactions were run in some sequential order.

⑦ File Caching Schemes -

It is used to improve the I/O performance by reducing disk transfers. It also address the following key decisions -

(i) Cache location
(ii) Modification propagation
(iii) Cache validation.

~~Cost of cache miss = cost of access disk + network transfer~~

~~Cost of cache hit = Network transfer~~

Date ___/___/___

Page _____

→ Cache location -

Cache behaves just like "networked virtual memory."

(1) Server's main memory -

Cost of cache miss = cost of access disk + network transfer

Cost of cache hit = Network transfer

Merits - One-time disk access, Easy implementation, Unix like file sharing semantics.

Demerits - Busy Network traffic.

(2) Client's disk -

Cost of cache hit = Time to access from local disk.

Merits - One-time Network Access, No size restriction, Suitable for supporting disconnected operation.

Demerits - Cache consistency problem, file access semantics, Frequent disk access, No diskless workstation.

(3) Client's Main Memory -

Cost of cache hit = maximum (high performance)

Merits - Maximum Performance, Diskless workstation, Scalability.

Demerits - Size restriction, Cache consistency problem, file access semantics

→ Modification Propagation -

The aim is keeping file data cached at multiple client node consistent. It has a critical effect on the system's performance and reliability. The file semantics supported depends greatly on the modification propagation scheme used.

Cache update policy -

(1) Write through - When a cache entry is modified, the new value is immediately sent to the server for updating the original copy of the file.

Pros - Unix like semantics and high reliability

Cons - Poor write performance.

(2) Delayed Write - The aim is to reduce network traffic for writes. When a cache entry is modified, the new value is written only to the cache.

and client just makes a note.

Pros - Write accesses complete quickly, some writes may be omitted by the following writes, Gathering all writes mitigates network overhead.

Cons - Delaying of write propagation results in fuzzier file-sharing semantics

→ Cache Validation -

Cache consistency is another important issue that determines how caches are maintained when multiple clients may be accessing the same file

Client-initiated approach -

Client is responsible for checking with the server to verify that each file in its cache is consistent.

A single corrupt or malicious client could disrupt the complete system.

Server-initiated approach -

Server acts as a central authority over which clients have up to date or invalid caches.

Server is able to detect when reading and writing clients might conflict with each other, and will send messages to client to force them to invalidate their cache entries and request them again.

Concurrent write sharing approach -

A file is open at multiple clients and at least one client has its open for writing. File server keeps track of the clients sharing a file.

(8) File Replication -

A replicated file is a file that has multiple copies, with each file on a separate file server.

Advantages of replication -

(1) Increased Availability
(2) Increased Reliability
(3) Improved Response time
(4) Reduced Network traffic
(5) Improved system throughput
(6) Better scalability

Date ___/___/___
Page _____

Differente between Replication and Caching -

(1) A replica of a file is associated with server, whereas a cached copy is normally associated with a client

(2) The existence of a cached copy is primarily dependent on the locality in file access patterns, whereas the existence of a replica normally depends on availability and performance requirement

(3) As compared to cached copy, a replica is more persistent, widely known, secure, available, complete and accurate.

(4) A cached copy is contingent upon a replica. Only by periodic revalidation with respect to a replica can a cached copy be useful.

Replication Transparency -

Replication of files should be transparent to the users so that multiple copies of a replicated file appear as a single logical file to its users. This calls for the assignment of a single identifier/name to all replicas of a file.

In addition, replication control should be transparent i.e. the number and locations of replicas of a replicated file should be hidden from the user. Thus replication control must be handled automatically in a user-transparent manner.

Multicopy Update Problem -

Major design issue of a distributed file system that supports file replication. TO ~~remove~~ avoid this problem, we can use -

(1) Read only replication protocol.

(2) Read-any-write-all protocol.

(3) Available-copies protocol - Same as (2) protocol but writes to all available copies of the file

(4) Primary-Copy protocol - Read operations can be performed using any copy, primary or secondary but write operations are performed only on the primary copy. Each server having a secondary copy updates its copy.

Date ___/___/___
Page _____

⑨ Fault Tolerance -

The approach of fault-tolerance expect faults to be present during system operation, but employs design techniques which insure the continued correct execution of the computing process.

The primary file properties that directly influence the ability of a distributed file system to tolerate faults are as follows:

(1) Availability (2) Robustness (power to survive crashes) (3) Recoverability

Stable Storage -

Information never lost. Not actually possible, so approximated via replication or RAID to devices with independent failure modes.

Two basic operation -

(1) A read operation first attempts to read from disk1. If it fails, the read is done from disk 2.

(2) A write operation writes to both disks, but the write to disk 2 does not start until that for disk1 has been successfully completed.

It is suitable for applications that require high degree of fault tolerence. Eg - Atomic transactions.

Effects of Service Paradigm on fault tolerance -

The file servers that implement a distributed file service can be stateless or stateful.

(1) Stateful file server -

Server maintains information about a file opened by a client that means it store session state. File operations supported by this server are -

(1) Open (filename, mode)
(2) Read (fid, n, buffer)
(3) Write (fid, n, buffer)
(4) Seek (fid, position)
(5) Close (fid) → causes the server to delete from its file table the file state information of the file identified by fid.

→ Advantages of Stateless servers -

(1) Fault Tolerance.
(2) No OPEN/CLOSE calls needed
(3) No server space wasted on tables
(4) No limits on number of open files
(5) No problem if a client crashes

Advantages of Stateful servers -

(1) Shorter request messages
(2) Better performance
(3) Read ahead possible
(4) Idempotency easier
(5) File locking possible

→ Disadvantages of stateful servers -

(1) Problem of orphan detection of elimination
(2) It loses all its volatile state in a crash

Disadvantages of Stateless servers -

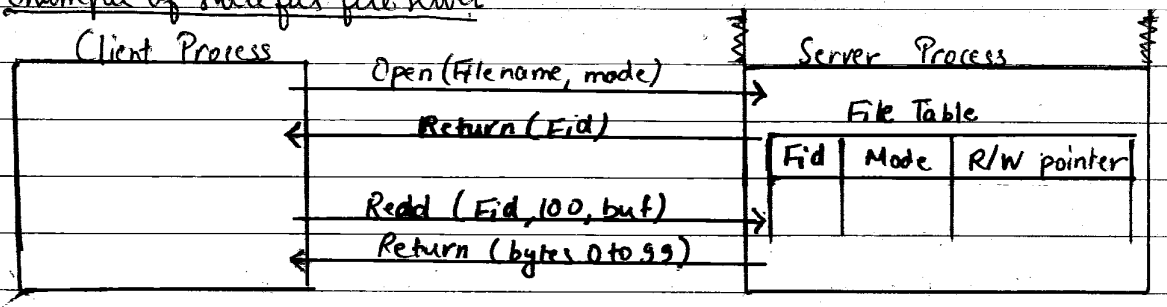
(1) less performance
(2) There is no longer request messages and slower processing of request

(2) Stateless file server -

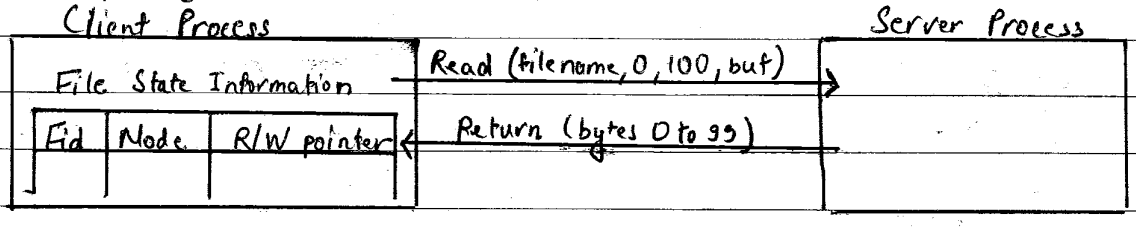
Server maintains no information about client access to files that means it do not store any session state. File operations are given as -

(1) Read (filename, position, n, buffer)
(2) Write (filename, position, n, buffer)

→ Example of stateful file server -



→ Example of stateless file server -



Date ___/___/___
Page _____

Naming –

① The naming facility (assign character-string names to objects) and locating facility (maps an object's name to the object's location) jointly form a naming system that provides the users with an abstraction of an object that hides the details of how and where an object is actually located in the network.

② Desirable features of a good naming system –

(1) location Transparency
(2) location Independency
(3) Scalability
(4) Uniform Naming Convention
(5) Multiple user defined names for the same object
(6) Group Naming
(7) Meaningful Names
(8) Performance
(9) Fault Tolerance
(10) Replication Transparency

③ System Oriented Names – (low level names)

They are of fixed size bit pattern that can easily manipulate and stored by machines. It basically meant for use by the system but may also be used by the user.

Characteristics –

(1) They are larger integers or bit strings.
(2) Also referred as unique identifier
(3) length is variable
(4) Automatically generated
(5) They are hard to guess and provide good security
(6) They are suitable for efficient handling by machines.

→ Centralized Approach for generating system-oriented names –

It generates structured and unstructured names. A standard & uniform global identifier name is generated for each object in the system by a centralized global unique identifier generator.

UNSTRUCTURED NAMES

A single field of large integers or bit strings

STRUCTURED NAMES

Node Identifier	Local Unique Identifier

Date __/__/__
Page _____

Advantages -
(1) Simple and easy to implement
(2) Only method used for generating unstructured global unique identifier.
Disadvantages -
(1) Poor efficiency and poor reliability
(2) Single global unique identifier generator may become a bottleneck for large name space.

→ Distributed approach for generating system-oriented names -
Hierarchical concatenation method is used to create global unique identifiers by concatenating the unique identifier of a domain.
Advantages -
(1) Better efficiency and reliability than centralized approach.
Disadvantages -
(1) Node boundaries or servers are explicitly visible
(2) The form and length of identifier may be different for different computer resulting in non-uniform global unique identifier.

(4) Object Locating Mechanisms -
It is the process of mapping an object's system-oriented unique identifier to the replica locations of the object. Various Types are -
(1) Broadcasting - Eg - Amoeba
A request is broadcast which is processed by all nodes and then the nodes currently having the object reply back to the client node
Advantages - simple, high reliability.
Disadvantages - Poor efficiency, poor scalability, number of node should be small, needed high communication speed.
(2) Expanding Ring Broadcast -
Modified form of broadcasting method. It consists of LAN connected by gateways. The distance metric used is a hop. A hop corresponds to a gateway between processors.

A name space is a collection of names which may or may not share an identical resolution mechanism.

Date ___/___/___
Page _____

It supplies nearest replica location but not necessarily all replica locations.

(3) Encoding location of object within its UID -

It uses structured object identifiers. It is straightforward and efficient scheme.

limitations -

(1) It is not clear how to support multiple replicas of an object
(2) An object is not permitted to move once it is assigned to a node.
(3) An object is fixed to one node throughout its lifetime.

**** One solution is to use forward location pointers but it increases object-locating cost, additional system overhead and difficult to locate if an intermediate pointer has been lost.

(5) Human-Oriented Names - (High level Names)

It is generally a character string and it is meaningful and identified by its user. Not unique for an object and are normally variable in length.

They cannot be easily manipulated, stored and used by the machines for identification purpose.

Characteristics -

(1) They are defined and used by the user
(2) Different users can define and use their own suitable names for a shared object
(3) Due to the facility of aliasing, the same name may be used by two different users at the same time to refer to two different objects

UNIT-3

Date ___/___/___
Page _____

INTER PROCESS COMMUNICATION AND SYNCRONIZATION

INTER PROCESS COMMUNICATION -

① API for Internet Protocol - (API → Application program Interface)

→ Characteristics of interprocess communication - (IPC)

(1) Synchronous and asynchronous communication -

In synchronous form of communication, the sending and receiving process synchronize at every message. In this case, both send and receive are blocking operations. Whenever a send is issued the sending process (or thread) is blocked until the corresponding receive is issued. Whenever a receive is issued the process (or thread) blocks until a message arrives.

In asynchronous form of communication, the use of the send operation is non-blocking in that the sending process is allowed to proceed as soon as the message has been copied to a local buffer, and the transmission of the message proceeds in parallel with the sending process. The receive operation can have blocking and non-blocking variants.

Non-blocking communication appears to be more efficient

(2) Message destination -

IPC can set messages to group of destinations (either ports or process). ~~A to~~ Messages are sent to (Internet address, local port) pairs. A local port is a message destination within a computer, specified as an integer.

Ports have advantage over processes bcz they provide several alternative points of entry to a receiving process

(3) Reliability -

Reliable communication should have validity and integrity property.

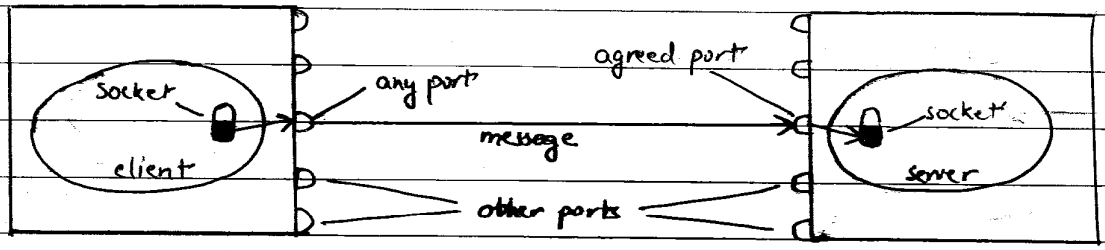
(4) Ordering -

Some application require that messages be delivered in sender order, that is, the order in which they were transmitted by the sender.

→ Sockets -

IPC consists of transmitting a message between a socket in one process and a socket in another process

Each socket is associated with a particular protocol - either UDP or TCP



Each computer has a larger number (2^{16}) of possible port numbers for use by local processes for receiving messages.

→ JAVA API for Internet Addresses -

As the IP packets underlying UDP and TCP are sent to internet addresses, Java provides a class, InetAddress, that represents Internet addresses. Users of this class refer to computers by DNS hostnames.

```
InetAddress aComputer = InetAddress.getByName("www.google.com");
```

→ UDP datagram communication -

A datagram sent by UDP is transmitted from a sending process to a receiving process without acknowledgment or retries. If a failure occurs, the message may not arrive.

→ Issues relating to datagram communication -

(1) Message size - The receiving process needs to specify an array of bytes of a particular size in which to receive a message. The underlying IP protocol allows packet lengths of up to 2^{16} bytes, which includes headers + message.

(2) Blocking - Sockets normally provide non-blocking sends and blocking receives for datagram communication.

(3) Timeouts - It is not appropriate that a process that has used a receive operation should wait indefinitely in situations where the potential sending process has crashed or the expected message has been lost. That's why timeouts can be set on sockets.

(4) Receive from any - The receive method does not specify an origin for messages. Only internet address and local port of the sender is known.

Date ___/___/___
Page _____

→ Failure Model for UDP datagrams - Two failures -

(1) Omission failures - Messages may be chopped occasionally, either because of a checksum error or because no buffer space is available at the source or destination.

(2) Ordering - Messages can sometimes be delivered out of sender order.

→ Use of UDP -

(1) Domain Naming service is implemented over UDP.

(2) Voice Over IP (VoIP).

(3) Do not suffer from overheads (need to store information, extra messages).

→ Java API for UDP datagrams -

Java API provides datagram communication by means of two classes DatagramPacket and DatagramSocket.

DatagramPacket -

array of bytes containing message	length of message	Internet Address	Port No.

getData method gives the message, getPort and getAddress access the port and Internet ~~Access~~ Address.

DatagramSocket -

send and receive methods are for transmitting datagrams between a pair of sockets. These methods throws IOExceptions.

setSoTimeout method allows the timeout to be set and ~~throws InterruptedIOExcpt~~ receive method will block for the time specified & then throw an InterruptedIOException.

connect method is used for connecting it to a particular remote port and Internet address

→ TCP stream communication -

The API to the TCP protocol provides the abstraction of a stream of bytes to which data may be written and from which data may be read.

→ Characteristics of the network that are hidden by the stream abstraction -

(1) Message Sizes - The application can choose how much data it writes to a stream or reads from it. The underlying implementation of a TCP stream

decides how much data to collect before transmitting it as one or more IP packets.
(2) Host messages - TCP protocol uses an acknowledgement scheme.
(3) Flow Control - TCP protocol attempts to match the speeds of the processes that read from and write to a stream.
(4) Message duplication and ordering - Message identifiers are associated with each IP packet, which enables the recipient to detect and reject duplicates, or to reorder messages that do not arrive in sender order.
(5) Message destination - A pair of communicating processes establish a connection before they can communicate over a stream.

→ Issues related to stream communication -

(1) Matching of data items - Two communicating processes need to agree as to the contents of the data transmitted over a stream.
(2) Blocking - The process that writes data to a stream may be blocked by the TCP flow control mechanism if the socket at the other end is queuing as much data as the protocol allows.
(3) Threads - When a server accepts a connection, it generally creates a new thread in which to communicate with the new client.

→ Failure Model for TCP stream -

To satisfy the integrity property of reliable communication, TCP streams use checksums to detect and reject corrupt packets and sequence numbers to detect and reject duplicate packets. For the sake of the validity property, TCP streams use timeouts and retransmission to deal with lost packets.

When a connection is broken -

(1) the processes using the connection cannot distinguish between network failure and failure of the process at the other end of the connection.
(2) the communicating processes cannot tell whether the messages they sent recently have been received or not.

→ Use of TCP - HTTP, FTP, Telnet and SMTP.

Date ___/___/___
Page _____

→ Java API for TCP streams –

The Java interface to TCP streams is provided in the classes Socket and ServerSocket

ServerSocket – to create a socket at a server port

accept method gets a connect request from the queue, or if the queue is empty, it blocks until one arrives. It gives access to streams for communicating with the client.

Socket –

getInputStream and getOutputStream methods are for accessing the two streams associated with a socket.

Socket can throw an UnknownHostException or IOException.

(2) External data representation and marshalling –

An agreed standard for the representation of data structures and primitive values is called an external data representation.

Marshalling is the process of taking a collection of data items and assembling them into a form suitable for transmission in a message.

Unmarshalling is the process of disassembling them on arrival to produce an equivalent collection of data items at the destination.

Three alternative approaches to external data representation and marshalling are –

(1) CORBA's common data representation (CDR) –

It is concerned with an external representation for the structured and primitive types that can be passed as arguments and results of remote method invocation (RMI) in CORBA.

→ Primitive types –

- Supports both big-endian and little-endian
- transmitted in sender's ordering and the ordering specified.
- receiver translates if needed.
- 15 primitive types – short (16 bit), long (32-bit), unsigned short, unsigned long, float (32 bit), double (64 bit), char, boolean, octet (8 bit) and any

basic & constructed type

- Constructed types are -

(1) sequence - length (unsigned long) followed by elements in order

(2) string - length (unsigned long) followed by characters in order

(3) array - array elements in order (no length specified because it is fixed)

(4) enumerated - unsigned long (the values are specified by the order declared)

(5) union - type tag followed by the selected member

→ CORBA CDR message -

index in sequence of bytes			
0-3	5	length of string	The flattened form represents a Person struct with value: ['Smith', 'London', 1934]
4-7	"Smit"	'Smith'	
8-11	"h___"		
12-15	6	length of string	
16-19	"Lond"	'london'	
20-23	"on__"		
24-27	1934	unsigned long	
← 4 bytes →			

The type of a data item is not given with the data representation in the message because it is assumed that the sender and receiver have common knowledge of the order and types of the data items in a message.

→ Marshalling in CORBA -

Marshalling operations can be generated automatically from the specification of the types of data items to be transmitted in message.

(2) Java object serialization -

It is concerned with the flattening and external data representation of any single object or tree of objects that may need to be transmitted in a message or stored on a disk.

```
public class Person implements Serializable {
    private String name;
    private String place;
    private int year;
    public Person (String aName, String aPlace, int aYear) {
```

Date ____/____/____
Page _____

```
    name = aName;
    place = aPlace;
    year = aYear;
}
// followed by methods for accessing the instance variables
}
```

In Java, the term serialization ~~interface, which~~ refers to the activity of flattening an object or a connected set of objects into a serial form that is suitable for storing on disk or transmitting a message. Deserialization consists of restoring the state of an object or a set of objects from their serialized form.

→ Indication of Java serialized form -

Serialized values				Explanation
Person	8-byte version number		h2	class name, version no.
3	int year	java.lang.String name	java.lang.String place	number, types and name of instance variables
1934	5 smith	6 London	h1	values of instance variables

The true serialized form contains additional type markers; h0 and h1 are handles.

References are serialized as handles. To serialize an object, its class information is written out, followed by the types and names of its instance variables

→ Serialization and deserialization of the arguments and results of remote invocations are generally carried out automatically by the middleware, without any participation by the application programmer.

→ The Use of Reflection -

The Java language supports reflection (the ability to enquire about the properties of a class, such as the names and types of its instance variables and methods. It also enables classes to be created from their names, and a constructor with given argument types to be created for a given class.

It helps in serialization and deserialization in a completely generic manner.

(3) Extensible Markup language (XML) –

It is markup language defined by world wide web consortium (W3C) for general use on the web. In general, the term markup language refers to a textual encoding that represents both a text and details as to its structure or its namespace.

Tag relate to the structure of the text that is enclose. XML data items are tagged with 'markup' strings.

XML is extensible in the sense that users can define their own tags.

The use of textual, rather than a binary representation, together with the use of tags makes the messages much larger, which causes them to acquire longer processing times and transmission times, as well as more space to store but the ability to read XML can be useful when things go wrong.

→ XML elements and attributes –

```
<person id = "123456789">
    <name> Smith </name>          XML definition of the
    <place> London </place>       Person structure.
    <year> 1934 </year>
</person>
```

Element - consists of a portion of character data surrounded by matching start and end tags.

Attributes consists of name and values.

Binary data of XML can be represented in base64 notation

→ Parsing and Well formed documents –

Every start tag has a matching end tag and all tags are correctly nested.

CDATA can be used where the section cannot be parsed or we can use &.

Eg - <place> <![CDATA [King's Cross]]> </place>

(first line) XML Prolog - Specifies the XML version, encoding and standalone status.

Eg - <?XML version = "1.0" encoding = "UTF-8" standalone = "yes"?>

→ XML Namespaces –

It is a set of names for a collection of element types and attributes, that is referenced by a URL.

Date __/__/__
Page _____

XML namespace — refer to the file containing the namespace definitions.

Eg - xmlns:pers = "http://www.cdk4.net/person"

prefix to refer to the elements

→ XML schemas -

It defines the elements and attributes that can appear in a document, how the elements are nested and the order and number of elements, whether an element is empty or can include text. For each element, it defines its types and default values. Eg - XML schema for the Person structure -

```
<xsd:schema xmlns:xsd = URL of XML scheme definitions>
  <xsd:element name = "person" type = "personType" />
    <xsd:complenType name = "personType">
      <xsd:sequence>
        <xsd:element name = "name" type = "ns:string" />
        <xsd:element name = "place" type = "ns:string" />
        <xsd:element name = "year" type = "ns:positiveInteger" />
      </xsd:sequence>
    <xsd:attribute name = "id" type = "ns:positiveInteger" />
    </xsd:complenType>
  </xsd:schema>
```

Document type definitions (DTDs) defines the structure of XML documents

→ APIs for accessing XML -

XML parsers and generators are available for most commonly used programming languages

→ Remote object references - (Applies only to Java & CORBA, not XML)

It is an identifier for a remote object that is valid throughout a distributed system. It is passed in the invocation ~~method~~ message to specify which object is to be invoked. Remote object references must be unique.

Representation of remote object reference -

32 bits	32 bits	32 bits	32 bits	
INTERNET ADDRESS	PORT NUMBER	TIME	OBJECT NUMBER	INTERFACE OF REMOTE OBJECT

③ Group communication -

A multicast operation sends a single message from one process to each of the members of a group of processes, usually in such a way that the membership of the group is transparent to the sender.

Characteristics of multicast messages -

(1) Fault tolerance based on replicated services.
(2) Finding the discovery servers in spontaneous networking.
(3) Better performance through replicated data.
(4) Propagation of event notifications.

→ IP multicast - an implementation of group communication

IP multicast allows the sender to transmit a single IP packet to a set of computers that form a multicast group. The sender is unaware of the identities recipients and of the size of the group.

The membership of multicast group is dynamic.

At the application programming level, IP multicast is available only via UDP. At the IP level, a computer belongs to a multicast group when one or more of its process has sockets that belong to that group.

→ Multicast Routers - Multicast in the Internet which forward single datagrams to routers on other networks with members, where they are again multicast to local members.

→ Multicast address allocation - It may be permanent or temporary assigned by Internet authority from the range 224.0.0.1 to 224.0.0.255.

For temporary →
- Uses time to live (TTL) to limit the number of hops
- Tools like sd (session directory) can help manage multicast addresses and find new ones

→ Failure model for multicast datagrams - same as UDP datagrams that is ~~omission~~ omission failure and ordering problem

→ Java API to IP multicast - through class MulticastSocket, which is a sub class of DatagramSocket

MulticastSocket methods are joinGroup, leaveGroup and setTimeToLive
↳ default is 1.

Date ____/____/____
Page _____

→ Reliability and ordering of multicast -

Effects of reliability and ordering of failure semantics of IP multicast are -

(1) Fault tolerance based on replicated services -

Ordering of the requests might be important, servers can be inconsistent with one another.

(2) Finding the discovery servers in spontaneous networking - Not too problematic

(3) Better performance through replicated data -

non and out-of-order updates could yield inconsistent data, sometimes this may be ~~in~~tolerable.

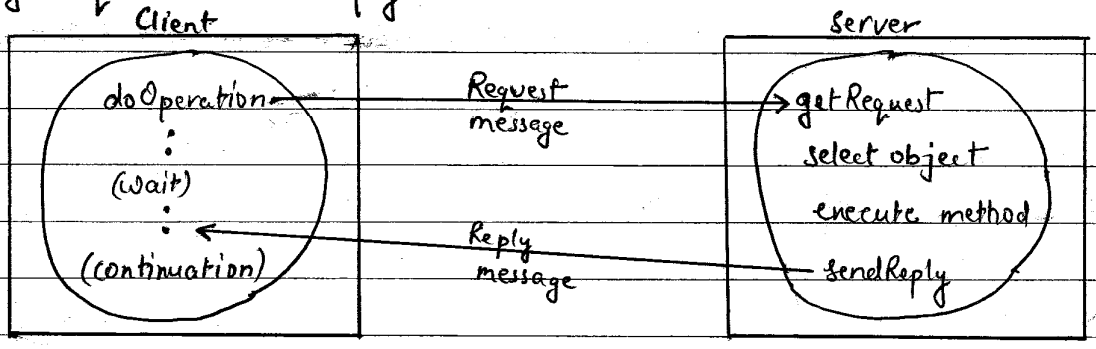
(4) Propagation of event notification - Not too problematic.

④ Client Server communication -

In Synchronous request-reply communication, client waits for a reply whereas Asynchronous request-reply communication, client doesn't wait for a reply.

→ Request-reply protocol -

It is based on a trio of communication primitives : doOperation, getRequest and sendReply.



REQUEST-REPLY COMMUNICATION

→ Operations of the request-reply protocol -

public byte[] doOperation (RemoteObjectRef o, int methodId, byte[] arguments);

public byte[] getRequest();

public void sendReply (byte[] reply, InetAddress clientHost, int clientPort);

Date __/__/__
Page _____

→ Request-reply message structure -

messageType	int (0 = Request, 1 = Reply)
requestId	int
objectReference	RemoteObjectRef
methodId	int or Method
arguments	// array of bytes

→ Message identifiers - Consist of two parts that is a requestId and an identifier like port and Internet address

When the value of the requestId reaches the maximum value for an unsigned integer (Eg → $2^{32}-1$) it is reset to zero. The only restriction here is that the lifetime of a message identifier should be much less than the time taken to exhaust the values in the sequence of integers -

→ Failure model of the request-reply protocol -

If implemented over UDP datagrams, they suffer from omission failures and order problem. In addition, the protocol can suffer from failure of processes.

→ Timeouts - Return immediately from doOperation with an indication to the client that the doOperation is failed. No getting a reply → Timeout and retry

→ Discarding duplicate request messages - Protocol is designed to recognize successive messages with the same request identifier and to filter out duplicates.

→ Idempotent Operation - It is an operation that can be performed repeatedly repeatedly with the same effect as if it had been performed exactly once.

→ History - For servers that require retransmission of replies without re-execution of operations, a history may be used. As clients can make only one request at a time therefore the history need contain only the last reply message sent to each client.

→ RPC Exchange Protocols -

Three protocols, which produce differing behaviours in the presence of communication failures, are used to offer implementing various types of RPC. request (R) protocol, request-reply (RR) protocol & request-reply-acknowledgment reply (RRA) protocol

Date ___/___/___
Page _____

	Messages sent by		
NAME	CLIENT	SERVER	CLIENT
R	Request		
RR	Request	Reply	
RRA	Request	Reply	Acknowledgement Reply

→ Use of TCP streams to implement the request-reply protocol -

(1) Avoid implementing multi-packet protocols.

(2) allows arguments and results of any size to be transmitted.

(3) Reliability increases

→ HTTP - an example of a request-reply protocol. (implemented over TCP)

Hypertext Transfer Protocol (HTTP) used by web browsers clients to make requests to web servers and to receive replies from them.

The protocol allows for content negotiation and password-style authentication.

HTTP 1.1 uses persistent connections - connections that remain open over a series of request-reply exchanges between client and server until the connection is closed by the server or client at any time or by server after timeout

Requests and Replies are marshalled into messages as ASCII text strings.

Resources implemented as data are supplied as MIME-like structures in arguments and results. Multipurpose Internet Mail Extensions (MIME) is a standard for sending multipart data containing, for eg text, images & sounds in email messages.

→ HTTP Methods -

GET → request the resource whose URL is given as argument.

HEAD → Returns all the information about the data.

POST → specifies the URL of a resource that can deal with the data supplied with the request.

PUT → request that the data supplied in the request is stored with the given URL as its identifier, either as a modification of an existing resource or as a new resource.

DELETE → the server deletes the resource identified by the given URL.

OPTIONS - the server supplies the client with a list of methods it allows to be applied to the given URL (eg. GET, HEAD, PUT) and its special requirements.

TRACE - the server sends back the request message.

→ Message Contents -

HTTP request message -

method	URL or pathname	HTTP version	Headers	message body

HTTP reply message -

HTTP version	status code	reason	headers	message body

RPC (Remote Procedure Call) -

① Implementing RPC mechanism -

To achieve semantic transparency, implementation of RPC mechanism is based on the concepts of stubs.

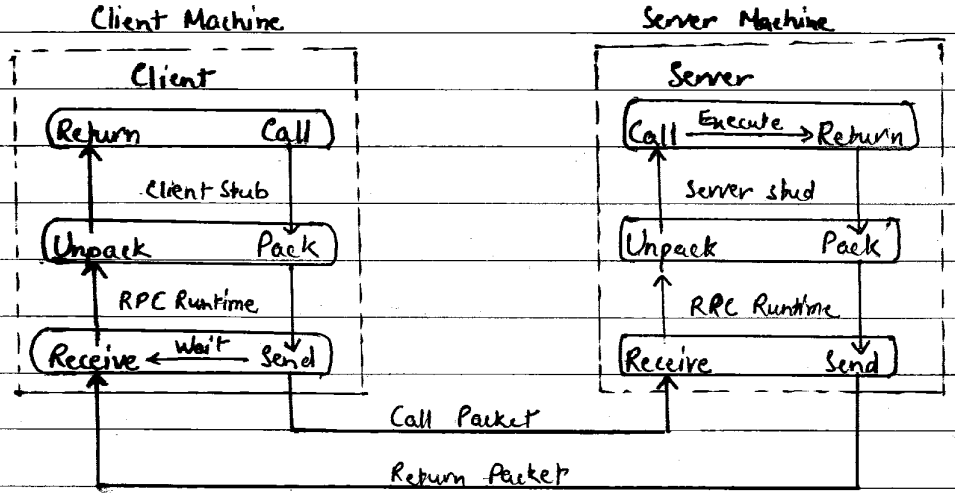
→ Stubs - It provides a normal / local procedure call abstraction by concealing the underlying RPC mechanism.

A separate stub procedure is associated with both the client and server processes.

To hide the underlying communication network, RPC communication package known as RPC Runtime is used on both the sides.

→ Thus implementation of RPC involves the five elements of program - Client, Client Stub, RPC runtime, Server Stub and Server.

The client, the client stub, and one instance of RPC Runtime execute on the client machine. The server, the server stub, and one instance of RPC Runtime execute on the server machine.



IMPLIMENTATION OF RPC MECHANISM

Date ___/___/___
Page _____

→ Client - It call the local procedure, called client hulk for remote service

→ Client Stub - It is responsible for two task -

(1) On receipt of a call request from the client
→ it packs a specifications of the target procedure & the arguments into a message
→ asks the local RPC runtime to send it to the server stub

(2) On receipt of the result of procedure execution, it unpacks the result and passes it to the client.

→ RPC Runtime - It handles transmission of messages across the network between client and server machine. It is responsible for retransmission, acknowledgement, routing and encryption.

→ Server Stub - It is responsible for two task -

(1) On receipt of a call request message from the local RPC Runtime, it unpacks it and makes a perfectly normal call to invoke the appropriate procedure in the server

(2) On receipt of the result of procedure execution from the server, it unpacks the result into a message and then asks the local RPC Runtime to send it to the client stub

→ Server - On receiving a call request from the server stub, the server executes the appropriate procedure and returns of procedure execution to the server hulk.

(2) Stub Generation -

It can be generated in one of the following two ways-

(1) Manually -

The RPC implementor provides a set of translation functions from which a user can construct his or her own ~~stubs~~ stubs. This method is simple to implement and can handle very complex parameter types.

(2) Automatically -

Interface Definition language (IDL) is used here, to define the interface between a client and the server.

→ Interface definition - It is a list of procedure names supported by the interface together with the types of their arguments and results.

It also play role in reducing data storage and controlling amount of data

process to another.

transferred over the ~~internet~~ network.

It has information about type definitions, enumerated types, and defined constants.

Export the interface - A server program that implements procedures in the interface.

Import the interface - A client program that calls procedures from an interface.

The interface definition is compiled by the IDL compiler.

→ IDL compiler generates -

(1) Components that can be combined with client and server programs, without making any changes to the existing compilers

(2) Client stub and server stub procedures

(3) The appropriate marshalling and unmarshaling operations.

(4) A header file that supports the data types.

③ RPC Messages -

Two types of messages involved in the implementation of an RPC system are

(1) Call Messages -

Sent by the client to the server for requesting execution of a particular remote procedure.

RPC call message format

Message Identifier	Message Type	Client Identifier	Remote procedure identifier (Basic component)			Arguments
			Program Number	Version Number	Procedure No	

Message Identifier: identify lost and duplicate messages

Message Type: 0 → call message, 1 → reply message

Client Identifier: for authentication and identification

(2) Reply Messages -

Sent by the server to the client for returning the result of remote procedure execution.

→ Conditions for unsuccessful message sent by the server -

(i) The server finds that the call messages is not intelligible to it.

(ii) Client is not authorized to use the service.

(iii) Remote procedure identifier is missing.

(iv) The remote procedure is not able to decode the supplied arguments

(v) Occurence of exception condition.

RPC reply message format

Message Identifier	Message Type	Reply Status (Successful)	Result

Successful reply message format

Message Identifier	Message Type	Reply Status (Unsuccessful)	Reason for failure

Unsuccessful reply message

SYNCHRONIZATION –

(1) Clock Synchonization –

→ How computer clocks are implemented –

A computer clock usually consists of three components –

(1) A quartz crystal that oscillates at a well-defined frequency.

(2) A constant register is used to store a constant value that is decided based on the frequency of oscillation of the quartz crystal.

(3) A counter register is used to keep track of the oscillations of the quartz crystal.

To make the computer clock function as an ordinary clock –

(1) The value in the constant register is chosen so that 60 clock ticks occur in 1s.

(2) The computer clock is synchronized with real time.

→ Drifting of Clocks –

The difference in the oscillation period between two clocks might be extremely small, but the difference accumulated over many oscillations lead to an obvious computer clock drift from the real-time clock

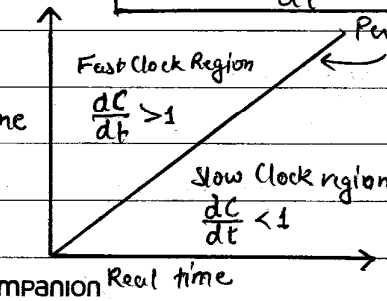
Clock based on a quartz crystal, drift rate is approximately 10^{-6}, giving a difference of 1 second every 1,000,000 seconds or 11.6 days.

A clock is said to non-faulty if the following condition holds for it –

$$1 - \rho \leq \frac{dC}{dt} \leq 1 + \rho$$

where $C \rightarrow$ time value of a clock

$\rho \rightarrow$ maximum drift rate.



time interval between two synchronization

$$\Delta t \leq \frac{\delta}{2\rho}$$

where $\delta \rightarrow$ time diff between two clocks

*** The difference in time values of two clocks is called clock skew.

Date ___/___/___
Page _____

→ Type of clock synchronization -

(1) Synchronization of the computer clock with real-time (or external) clocks.

(2) Mutual (or internal) synchronization of the clocks of different nodes of the system.

→ Clock Synchronization Issues -

(1) Clock synchronization requires each node to read the other nodes' clock values.

(2) Time must never run backwards (in case of fast clock readjusted to actual time).

→ Clock Synchronization Algorithms -

(1) Centralized Algorithms -

One node has a real-time receiver. This node is usually called the time server node, and clock time of this node is regarded as correct and used as the reference time. The goal of the algorithm is to keep the clocks of all other nodes synchronized with the clock time of the time server node. Two types:

(i) Passive time server centralized algorithm -

In this method, each node periodically sends a message to the time server. When the time server receives the message, it quickly responds with a message.

The clock is readjusted to $\boxed{T + (T_1 - T_0)/2}$, where

$T \rightarrow$ current time, $T_0 \rightarrow$ when the client node sends the message.

$T_1 \rightarrow$ when it receives the 'time=T' message.

Two cases -

[1] If additional information is available then, $\boxed{T + (T_1 - T_0 - l)/2}$, where

$l \rightarrow$ time taken by the time server to handle the interrupt and process a time request message.

[2] If additional information is not available then, several measurement of $T_1 - T_0$ are made. Minimum value of $T_1 - T_0$ is considered to be the most accurate one.

then ~~$T + m(T_1 - T_0)$~~ $\boxed{T + [\min(T_1 - T_0)]/2}$

(ii) Active time server centralized algorithm -

The time server periodically broadcasts its clock time. The other nodes receive the broadcast message and use the clock time in the message for correcting their own clocks.

fault-tolerant average -

Time server chooses a subset of all clock values having ~~values~~ do not ~~differ~~ differ from one another by more than a specified amount and then the average is taken.

Date ___/___/___
Page ___

Node's clock is readjusted to the time $T+T_0$ where $T \rightarrow$ current time.
$T_0 \rightarrow$ Prior knowledge of the approximate time (T_0) required for the propagation of the message from the server node to its own node.

Drawback - Not fault tolerant, requires broadcast facility

To remove the above drawback, Berkeley algorithm can be used.

→ Berkeley algorithm -

The time server periodically sends a message to all the computers in the group. On receiving this message, each computer sends back its clock value to the time server.

The time server has a prior knowledge of the approximate time required for the propagation of a message from each node to its own node.

It then takes a fault-tolerant average of the clock values of all the computers (including its own). The calculated average is the current time to which all the clocks should be readjusted.

Server readjusts its own and sends the amount by which each individual's computer's clock's requires adjustment (positive or negative value)

→ Major Drawbacks of Centralized clock synchronization algorithms -

(1) Single-point failure (time server node fails)

(2) No scalability

(2) Distributed Algorithms -

A simple method for clock synchronization may be equip each node of the system with a real-time receiver so that each node's clock can be independently synchronized with real time.

Theoretically, internal synchronization of clocks is not required in this approach.

~~One~~ Two types of distributed algorithms are -

(1) Global Averaging Distributed Algorithms -

In this approach, the clock process at each node broadcasts its local clock time in form of a special "resync" message when its local time equals $T_0 + iR$.

$T_0 \rightarrow$ fixed time in the past agreed upon by all node, $i \rightarrow$ some integer

$R \rightarrow$ system parameter depends on factors like No. of nodes, maximum allowable drift etc.

Broadcasting nodes waits for time T during which it collects "resync" messages by other nodes & record time of receipt according to its own clock.

At the end of waiting time, it estimates the skews of its clock w.r.t other nodes on the basis of times at which it received "resync." messages.

Calculate fault tolerent average of estimated ~~guess~~ skews & use it to correct its own local clock before start of next "resync" interval.

→ Two algorithms used -

(i) Take the average of the estimated skews and use it as the correction for the local clock.

(ii) Each node limits the impact of faulty clocks by first discarding the m highest and m lowest estimated skews and then calculating the average of the remaining skews and then ~~calculating the average~~ used as the correction for the local clock.

(ii) Localized Averaging Distributed Algorithms -

It attempts to overcome these drawbacks of the global averaging algorithms. Nodes are arranged in ring or grid. Periodically, each node exchanges its clock time ~~to the average~~ with its neighbours then sets its clock time to the average of its own clock time & clock times of its neighbours.

→ Case Study: Distributed Time Service (DTS) -

DTS is a component of DCE (Distributed Computing Environment) that is used to synchronize clocks of a network of computers running DCE.

DTS uses the the usual client-server structure: DTS clients, daemon process called DTS clerks, request the correct time from some number of servers, receive responses, and then reset their clocks as necessary to reflect this new knowledge.

Components of DCE DTS are -

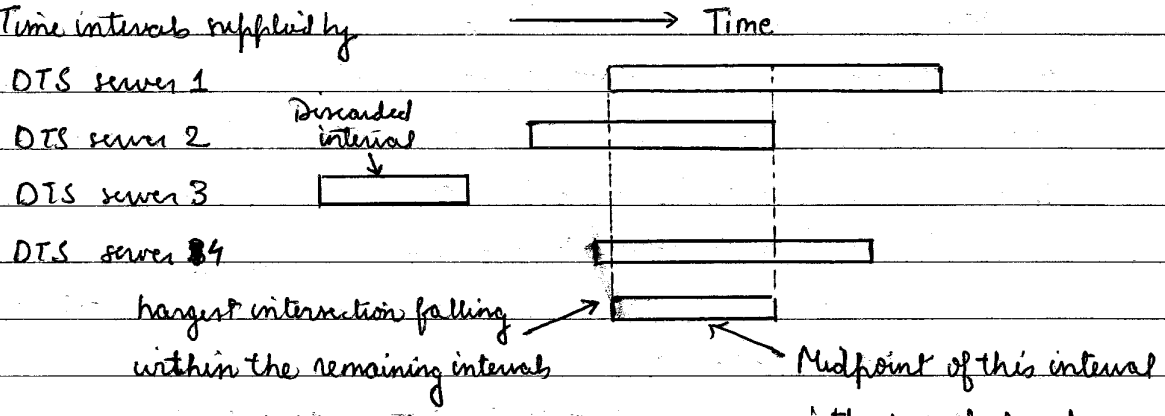
(1) DTS clerks

(2) Time servers - three types -

(a) local time server maintains the time synchronization of a given LAN. The global time server and courier time server are used to synchronize time among interconnected LANs.

(3) DTS API provides an interface where application programs can access time information by the DTS.

→ Computation of new clock value in DTS from obtained time intervals -



(2) Mutual exclusion -

Mutual exclusion are introduced to prevent process from executing concurrently with their associated critical sections.

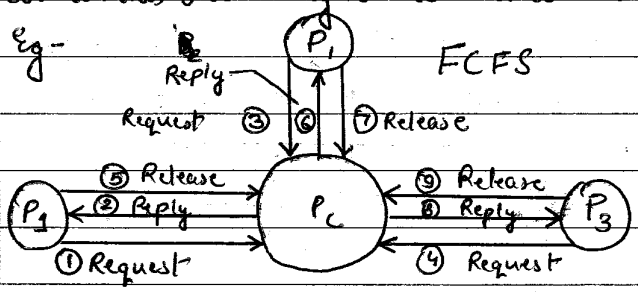
An algorithm for implementing mutual exclusion must satisfy the requirements that is mutual exclusion and No starvation.

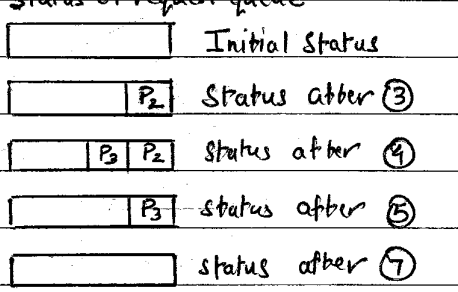
Three approaches for implementing mutual exclusion in distributed systems are -

(1) Centralized Approach -

One of the processes in the system is elected as coordinator and coordinates the entry to the critical sections.

Eg -





Advantages - simple to implement & requires only three messages per critical section.

Drawbacks - single point failure (due to centralized coordinator)

(2) Distributed Approach –

All processes that want to enter the same critical section cooperate with each other before reaching a decision on which process will enter the critical section next.

When a process wants to enter a critical section, it sends a request message to all other processes. Message contains process identifier, name of the critical section and unique timestamp.

On receiving a request message –

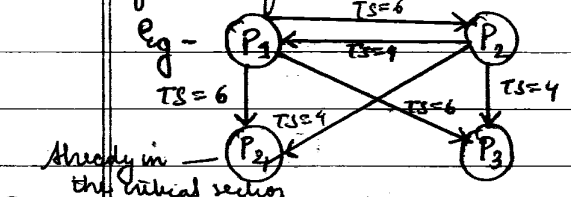
(1) Receiver process defers sending a reply if the receiver itself executing in the critical section and queues the request message.

(2) If the receiver process waiting for its turn to enter the critical section, it compares the timestamp of itself and the received request message. If the timestamp of received request message is lower, the receiver process sends the reply and if the timestamp of receiver process is lower, it defers sending a reply and queues the request message.

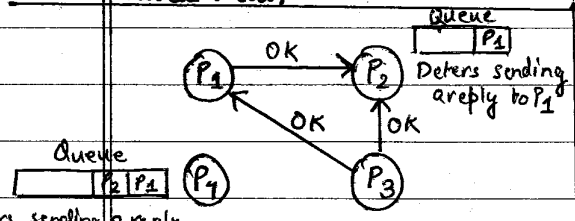
(3) If the receiver process neither in the critical section nor is waiting for turn, then it immediately reply back a reply message.

A process enters the critical section as soon as it has received reply messages from all processes. After it finishes executing in the critical section, it sends reply messages to all process in its queue & deletes them from its queue.

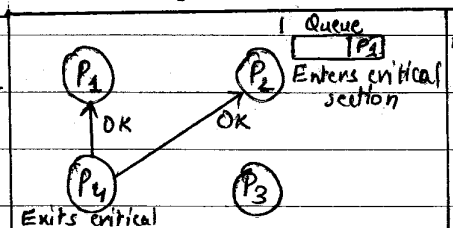
Eg –



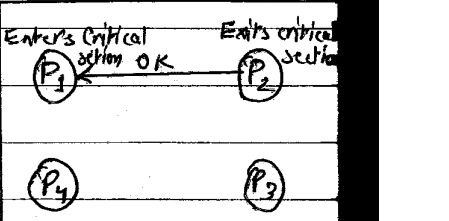
States when processes P_1 and P_2 send request messages to other process while process P_4 is already in the critical section.



States while P_4 is still in critical section



States after process P_4 exists critical section



States after process P_2 exits critical section

2(n-1) messages per critical section entry [n processes, n-1 request messages, n-1 reply messages].

Drawbacks - n points of failure, each process must know the identity of all the processes, waiting time may be large

(3) Token-Passing Approach -

A single token is circulated among the processes in the system (organized in a ring structure - clockwise or anticlockwise). A token is a special type of message that entitles its holder to enter a critical section.

When a process receives a token, if it wants to enter the critical section, it keep the token, enter the critical section and exits from the critical section and then passes the token along the ring (Note → one critical section at a time) or if it does not want to enter the critical section, it just passes the token along the ring.

Drawbacks - Process failure (logical ~~link~~ ring breaks), lost token

③ Election Algorithms -

They are meant for electing a coordinator process from among the currently running process. It based on following assumptions -

(1) Each process in the system has a unique priority number

(2) Whenever an election is held, highest priority number process is elected

(3) On recovery, failed process can rejoin the set of active processes.

Two election algorithms are given as -

(1) The Bully Algorithm -

A process starts an election if it detects the coordinator is failed then →

→ sends an election message to all processes with higher id's & wait for answers (except the failed coordinator / process)

If no answer in time T then, it becomes coordinator and sends coordinator message (with its id) to all processes with lower id's.

else waits for a coordinator message or starts an election if timeout

→ Receiving an election message -
sends an answer back and starts the election if it hasn't started one. Also send election messages to all higher-id process (including the 'failed coordinator' because the coordinator might be up by now)
→ Receiving a coordinator message - set elected, to the new coordinator
→ If failed coordinator recovers then it simply sends a coordinator message to all other processes and bullies the current coordinator into submission.

(2) A Ring Algorithm - (Ring structure)
A process starts the election if it detects the coordinator is failed then -
→ starts ~~the~~ an election message → mark itself as participant → places its id's in an election message and sends the message to its neighbour & neighbours do the same.
→ The election message returns back to the process then -
select highest priority number process as coordinator and inform others about the new coordinator along the ring.
→ When coordinator process recovers from failure then it ~~inform~~ creates an inquiry message and sends along the ring until it reaches the current coordinator which ~~starts sending~~ informing other process of the new coordinator

	Bully Algorithm -	Ring Algorithm -
~~In case of~~ Assigning new coordinator	worst case → $O(n^2)$ messages	worst case → $2(n-1)$ messages
	best case → $n-2$ messages	best case → $2(n-1)$ messages
Recovery of failed coordinator process	worst case → $O(n^2)$ messages	worst case → $n/2$ messages
	best case → $n-1$ messages	best case → 1 message

Ring algorithm is more efficient and easier to element than bully algorithm.

UNIT- 4

Date ___/___/___
Page _____

DISTRIBUTED SCHEDULING AND DEADLOCK

DISTRIBUTED SCHEDULING -

(1) Distributed scheduling refers to the ~~execution of non interactive~~ chaining of different jobs into a coordinated workflow that spans several computers.

(2) **Issues in load Distributing -**

(1) load -

load estimation is ~~called~~ calculated by using resource queue lengths and CPU utilization.

Queue length of waiting task proportional to task response time, hence a good indicator of system loads.

Distributed load = transfer tasks / process among nods

If a task transfer (from another node) takes a long time, the node may accept more tasks during the transfer time causes the node to be highly loaded and affect performance.

Solution - Artificially increment the queue length when a task is accepted for transfer from remote node.

(2) Types of algorithm -

Basic function of a load distributing algorithm is to transfer load (task) from heavily loaded computers to idle or lightly loaded computers. It can be characterized as -

(i) Static load distribution algorithm - Decisions are hard coded into an algorithm with a priori knowledge of system.

(ii) Dynamic load distribution algorithm - Use system state information such as task queue length, processor utilization

(iii) Adaptive load distribution algorithm - adapt the approach based on system state.

Adaptive stop collecting information at high load but dynamic still collects information at high load.

(3) Load Balancing Vs load sharing -

load Balancing algorithm is also ~~co~~ classified as load balancing and load sharing.

Location selection policy -

Another approach is that a task is selected for transfer only if its response time will be improved upon transfer

Date ___/___/___
Page _____

Load balancing - More no. of task transfers → degrade performance

Load balancing algorithm go a step further by attempting to equalize loads at all computers (participating nodes)

Transfer tasks even if a node is not heavily loaded so that queue lengths on all nodes are approximately equal.

Load sharing - less no. of task transfer

Reduce burden of an overloaded node by task transfer to idle or lightly loaded nodes. This task transfer is known as anticipatory task transfer

Transfers tasks only when the queue length exceeds a certain threshold.

(4) Preemptive Vs Nonpreemptive transfers -

Preemptive task transfers involve the transfer of a task that is partially executed which is expensive as it involves collection of task states such as virtual memory image, process control block, IO buffers etc.

Nonpreemptive task transfer involve the transfer of a task that has not begun execution (that means no task states transfer required). It can be considered as task placements. Suitable for load sharing not for load balancing.

③ Components of load distributing algorithms -

(1) Transfer policy -

Determines when a node is ready to participate in a task transfer.

When a load on a node exceeds a threshold T, the node becomes a sender. When it falls below a threshold, it becomes a receiver.

(2) Selection policy -

Determines which task should be transferred.

Approach - Select newly originated tasks because transfer cost is lower as no state information is to be transferred. Non-preemptive transfers are allowed.

Factors of selection - (1) Smaller task have less overhead, small response time.

(2) location-dependent system calls should be minimal.

(3) location Policy -

Determines the receiving node for a task.

Polling is generally used which can be done serially or in parallel (using multicast)

Alternative - broadcasting a query, sort of invitation to share load.

(4) Information Policy -

Responsible for triggering the collection of system state information.

Demand-driven collection - Only when a node is highly or lightly loaded.

- Sender initiated policies → sender looks for receivers to transfer their load.
- Receiver initiated policies → receiver solicit load from senders
- Symmetric initiated policies → combination of sender & receiver initiated policies

Periodic - Do not adapt to system state, are slow to respond, and can make the situation worse by increasing system load.

State-change driven - Only when state changes by certain degree.

(4) Different types of load distributing algorithms -

Four types of load distributing algorithms are -

(1) Sender-initiated algorithms -

load distributing activity is initiated by an overloaded node (sender) that attempt to send a task to an underloaded node (receiver).

→ Transfer Policy - CPU queue threshold T for all nodes. Initiated when a new task arrives.

→ Selection Policy - Once the transfer policy decides that a host is a sender, a selection policy selects a task for transfer.

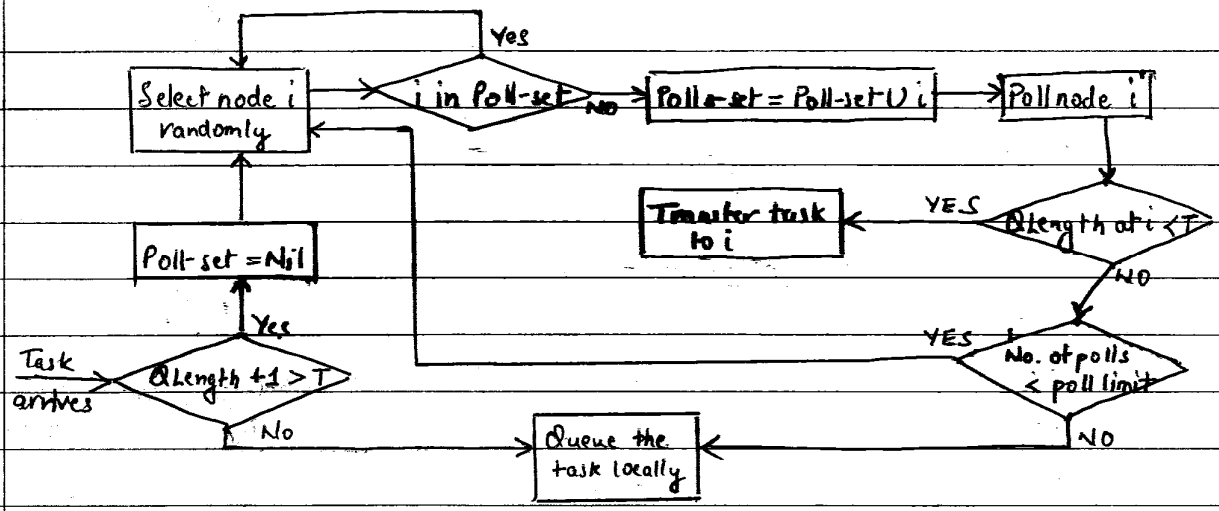
The simplest and popular approach is to select the newly arrived task for transfer that just transforms the host into a sender

→ Location Policy - One of the major tasks of location policy is to check the availability of the service(s) required for proper execution of the migrated and/or re-scheduled task(s) within the selected transfer partner.

(1) Random - Select any node to transfer the task at random. The selected node X may be overloaded. If transferred task is treated as new arrival, then X may transfer the task again. limit the no. of transfers for a task.

It is effective under light-load conditions

(2) Threshold - Poll nodes until a receiver is found. Up to PollLimit nodes are polled. If none is a receiver, then the sender commits to the task.



SENDER INITIATED LOAD SHARING WITH THRESHOLD LOCATION POLICY

(3) Shortest – Among the polled nodes that where found to the receiver, select the one with the shortest queue. Marginal Improvement.

→ Information Policy –

Demand driven policies → Uses decentralized approach

Periodic policies → Either centralized or decentralized approach

State-change driven policies → Either centralized or decentralized approach.

→ Stability – Unstable at high loads (Drawback)

(2) Receiver Initiated Algorithm –

load distributing activity is initiated from an underloaded node (receiver) that is trying to obtain a task from an overloaded node (sender).

→ Transfer Policy – same as in components of load distributing algorithm

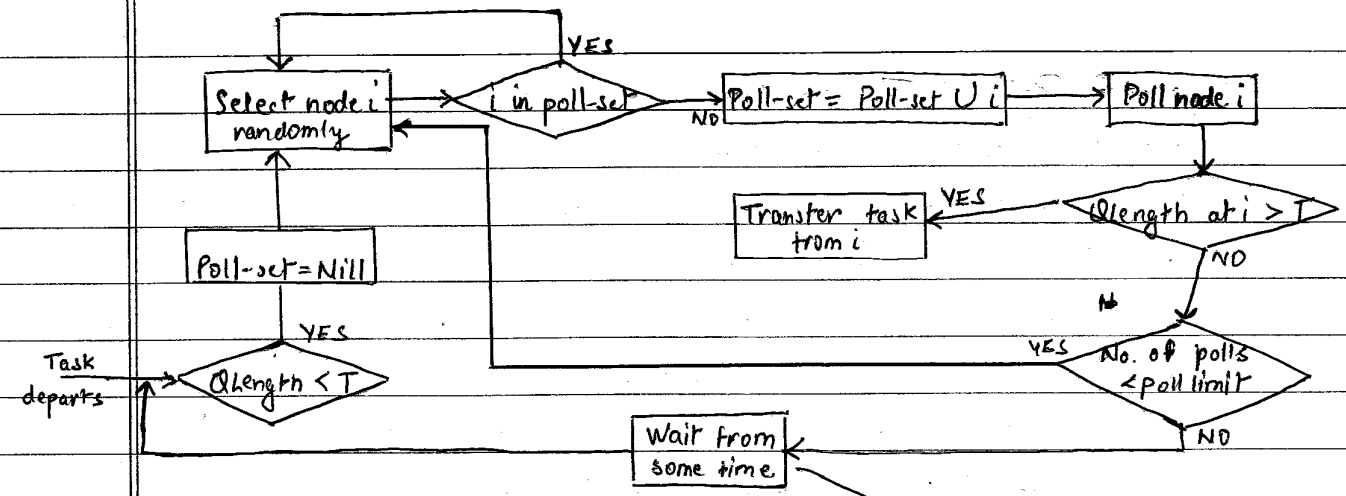
→ Section Selection Policy – Any of the approaches can be used that is described in components for load distributing algorithm.

→ Location Policy – A node selected at random is polled to determine if transfering a task from it would place its queue length below the threshold level. If not, the polled node transfer the task. Remaining process are explained in the given diagram of receiver initiated load sharing.

→ Information Policy – Demand driven because polling activity starts only after a node becomes a receiver.

→ Stability – Stable at high loads as well as low loads. (because of CPU cycles available)

(receiver will find senders with high probability with a small number of polls)



RECEIVER - INITIATED LOAD SHARING

Note that if the search does not start after a predefined period, the extra processing power available at a receiver is completely lost to the system until another task completes, which may not occur soon.

→ Drawbacks - Most ~~task~~ transfers are preemptive and therefore expensive.

(3) Symetrically initiated algorithms -

Combination of sender-initiated and receiver-initiated algorithms. in which senders search for receivers and receivers search for senders.

how loads - Senders can find receivers easily.

High loads - Receivers can find senders easily.

Drawbacks - Unstable at high load and transfers are preemptive (expensive)

A simple symetrically initiated algorithm can be constructed by using both the transfer and location policies of sender & receiver initiated algorithms.

Another symetrically initiated algorithm is given as -

→ Above Average algorithm -

It tries to maintain the load at each node within an acceptable range of the system average.

→ Transfer Policy - Two adaptive thresholds instead of one. If a node's estimated average load is A, a higher threshold [TooHigh > A] and a lower threshold [TooLow < A] is used.

Load < TooLow ⇒ Receiver , Load > TooHigh ⇒ Sender

→ Selection Policy - Any of the approaches can be used decribed in components for load distributing algorithms.

→ Location Policy - Two components -

(i) Sender initiated component -

① → Node with Too High load, broadcasts a Too High message, sets Too High timer, and listen for an accept message

② → A receiver that gets the Too High message sends an accept message, increase its load, and sets Awaiting Task Timer.

③ → If Awaiting Task timer enpires, load is decremented.

④ → On receiving the Accept message :- If the node is still a sender, it chooses the best task to transfer and transfers it to the node.

⑤ → When sender is waiting for Accept, it may receive a Too Low message (receiver initiated). Sender sends Too High to that receiver.

⑥ → On enpiration of Too High timer, if no Accept message is received, system is highly loaded. Sender broadcasts a Change Average message.

(ii) Receiver initiated component -

① → Node with Too low load, broadcasts a Too Low message, sets a Too low timer, and listens for Too High message.

② → If Too High message is received, do step 2 and 3 in sender initiated component

③ → If Too Low Timer enpires before receiving any Too High message, receiver broadcast a Change Average message to decrease the load estimate at the other nodes.

→ Information Policy - Demand driven. Average load is modified based on system load. High loads may have less number of senders frequently.

(4) Adaptive Algorithms -

→ Stable symmetrically initiated algorithms -

It utilizes the information gathered during polling (instead of discarding it as was done by previous algorithms) to classify the nodes in the system as either Sender/overloaded, Receiver/under loaded, or OK.

The knowledge concerning the state of node is maintained by a data structure at each node - a sender list, a receiver list, and an OK list.

Initially, each node assumes that every other node is a receiver.

→ ~~Transfer policy - Same as symmetrically initiated algorithm. OK if load is~~

→ Transfer Policy - LT → lower threshold UT → upper threshold.

~~Adaptive~~ is OK if LT ≤ queues length ≤ UT

Sender if its queue length > UT and receiver if its queue length < LT

It is triggered when a new task originates or when a task departs.

→ Selection Policy -

Sender-initiated component considers only newly arrived tasks for transfer

Receiver-initiated component can make use of any of the approaches of this policy

→ Location Policy - Two components -

(i) Sender-initiated component -

① Sender polls the node at the head of its receivers list to find out whether it is still a receiver

② The polled node removes the sender node ID from the list it is presently in & puts it in the sendersList

③ The polled node returns its status (receiver, sender, ok) to the sender.

④ The sender transfers a task to the node if it is a receiver.

⑤ The sender puts the polled node in appropriate list based on its reply.

⑥ This process may continue.

⑦ This ~~pol~~ polling process stops, if suitable receiver is found or if no. of polls reaches a pollLimit or if receivers list becomes empty

⑧ If receiver is not found, then the task is processed locally

(ii) Receiver initiated component -

① The receiver polls the nodes from the first to the last in the sender's list, then it polls the OK list from last to first and then it polls the receiver's list from last to first

② If the polled node is a sender then it transfers a task and informs the receiver about its status after the task transfer.

③ If the polled node is not sender then it removes the receiver node ID from the list it is presently in and puts it in the receiver's list and informs the receiver about its status.

④ The receiver puts the polled node in appropriate list based on the reply.

⑤ This process may continue.

⑥ Polling process stops, ~~when~~ if sender is found, if receiver is no longer a receiver ~~and~~ or if no. of polls reaches pollLimit

→ Information Policy - Demand driven, as the polling activity starts when a node becomes a sender or a receiver.

→ Stable sender-initiated algorithm -

Two desirable properties -

(1) Does not cause instability

(2) Load sharing is due to non-preemptive transfers only.

Similar to stable symmetrically initiated algorithm with the modified modification of receiver initiated component

In this algorithm, Statevector array is used by each node to keep track of which list (senders, receivers or OK) it belongs to at all the other nodes in the system.

Receiver initiated component modified protocol - When a node becomes a receiver it informs all the nodes ~~whose receiver~~ that are misinformed about its current state with the help of statevector at the receiver to to find the misinformed nod

(5) Task Migration -

Task Migration refers to the transfer of a task that has already begun to a new location and continuing its execution there.

Task placement refers to the transfer of a task that is yet to begin execution to a new location and ~~continue~~ start its execution there.

→ Benefits of task migration - load balancing, reduction in communication overhead resource access and fault tolerance.

→ Steps involved in task migration -

(1) Suspending (freezing) the task on the source.

(2) Extracting and transmitting the state of the task to destination

(3) Reconstructing the state on the destination

(4) ~~Removing~~ Resuming the task's execution on the destination.

(6) Issues in Task Migration -

Three issues in task migration are -

(1) State Transfer, (2) location transparency, (3) Structure of a migration mecha

Date ___/___/___
Page _____

→ State Transfer - Two important issues are -

as little time as possible

(1) The cost to support remote execution, which includes delays due to freezing the ~~do~~ task, obtaining and transferring ~~reduce~~ the state & unfreezing the task.

(2) Residual dependencies - refers to the amount of resources a host of a migrated task continues to dedicate to service requests from the migrated task. They are undesirable for three reasons - reliability, performance and complexity.

→ State transfer mechanisms -

(1) Precopying the State → bulk of the task state is copied to the new host before freezing the task

(2) location-transparent file access mechanism

(3) Copy-on-reference - Just copy what is migrated task need for its execution.

→ location transparency -

Task migration should hide the locations of tasks. location transparency in principle requires that names (process name, file names) be independent of their locations (host names)

Uniform name space throughout the system.

→ Structure of ~~the~~ a migration mechanism -

Typically, there will be interaction between the task migration mechanism, the memory management system, the inter-process communication mechanism and the file system. The ~~of~~ mechanism can be designed to be independent of one another so that if one mechanism's protocol changes, the other's need not the migration mechanism can be turned off without interfering with other mechanisms.

Date ___/___/___
Page _____

DEADLOCK -

① Issues in deadlock detection and resolution -

→ Detection - Two issues -

maintenance of the WFG (wait-for-graph) and search of the WFG for the presence of cycles (or knots).

Depending upon the manner in which WFG information is maintained and the search for cycles is carried out, there are centralized, distributed and hierarchical algorithms for deadlock detection in distributed systems.

A correct deadlock detection algorithm must satisfy two conditions -

(1) Progress - No undetected deadlocks - all detect all existing deadlock in finite time, and progress continuously to find more deadlocks

(2) Safety - No false deadlocks - Should not report deadlocks which are non-existent (phantom deadlock). Due to no global memory or communication, sites may obtain out of date & inconsistent WFGs of the system.

→ Resolution -

Deadlock resolution involves breaking existing wait for dependencies in the system WFG to resolve the deadlock.

It involves rolling back one or more processes that are deadlocked and assigning their resources to blocked processes in the deadlock so that they can resume execution.

When a wait-for dependency is broken, the corresponding information should be immediately cleaned from the system so that it may not result in detection of phantom deadlocks.

② Deadlock Handling Strategies - Three strategies -

(1) Deadlock Prevention -

It is achieved by either having a process acquire all the needed resources simultaneously before it begins execution or by preempting a process that holds the needed resources.

Drawbacks - ① Inefficient, decreases the system concurrency.

② A set of processes can become deadlocked in the resource acquiring phase.

③ Future resource requirements are unpredictable.

→ Heterogeneous – Potentially different DBMS's are used at each node
- Gateways – Simple paths are created to other databases, without the benefits of one logical database
- Systems – Support some or all the functionality of one logical database
  - Full DBMS functionality – Support all of the functionality of distributed database
  - Partial-Multidatabase – Support some features of a distributed database
    - Unfederated – Requires all access to go through a central coordinating module
    - Federated – Supports local databases for unique data requests
      - Loose Integration – Many schemas exists, for each local database, and each local DBMS must communicate with all local schemas
      - Tight Integration – One global ~~the~~ schema exists that defines all the data across all local databases.

DISTRIBUTED MULTIMEDIA –

① Characteristics of Multimedia Data –

(1) Continuous –
- Refers to the user's view of the data
- Video : a image array is replaced 25 times per second.
- Audio : the of amplitude value is replaced 8000 times per second.

(2) Time-based –
- The time at which the values are played or recorded after the validity of the data
- Hence, the timing should be preserved.

② Quality of service management –

The management and allocation of resources to provide such guarantees is referred to as quality of service management.

If there is a system component responsible for the allocation and scheduling of those resources. That component is refered the quality of service (QoS) manager.

Date ____/____/____
Page _____

→ Usage of resource requirement specification –

(1) To describe the characteristics of a multimedia stream in a particular environment. Eg.– Video conference (Bandwidth → 1.5 Mbps, delay → 150ms, loss rate → 1%)

(2) To describe the capabilities of resources to transport a stream. Eg – a network may provide → Bandwidth: 64Kbps, delay: 10ms, loss rate: 1/1000.

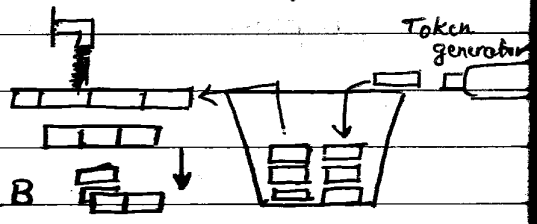
→ Traffic Shaping –

It is a term used to describe the use of output buffering to smooth the flow of data elements. Two algorithms –

(1) Leaky bucket algorithm –

- Completely eliminate burst
- A stream will never flow with a rate higher than R
- B is the size of the buffer
- B bound the time for which an element will remain in buffer

(2) Token bucket algorithm –



- Allows larger burst
- Token is generated at fixed rate of R
- the tokens are collected in a bucket of size B
- Data of size S can be sent only if at least S tokens are in the bucket
- Ensure that over any interval t, the amount of data is not larger than $Rt+B$

→ Flow specification – A collection of QoS parameters

→ RFC 1363 –

(1) Bandwidth – Maximum transmission unit and maximum transmission rate. Token bucket size & rate determine the burstiness of the stream.

(2) Delay – Minimum delay that can application can notice and the maximum jitter it can accept.

(3) loss rate – Total acceptable no. of losses over a certain interval. Maximum no. of consecutive losses.