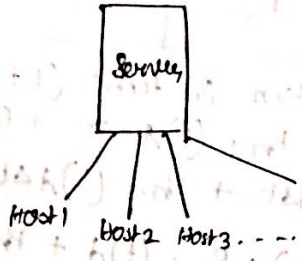
Msc - Sem-1 S.S. 3.8.2017

Distributed Operating System (DOS):

Centralised OS
Distributed OS
} — OS.

Client server —



Monolithic-kernel model.
↓
has a certain type of program that can perform all the jobs and handle all the resources.

Collection of computers. Connected via some network.
They can perform their own local operation.
Each can perform one global operation.
Every computer is represented as virtual uniprocessor.
The kernel is Micro-kernel.

Advantages of Distributed OS over Centralised OS:

	Centralised	Distributed
Speed	less speed	more speed.
Reliability	less reliable	more reliable.
Incremental Growth.	There is a limit.	No limit
		Economically Better.

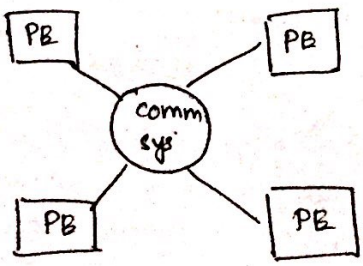
P.K Sinha

Inherent Distributions:

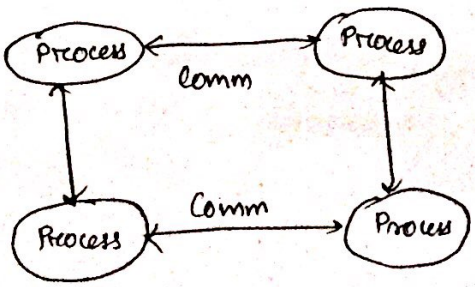
4.08.2017

SM

A distributed system is a collection of independent computers that appear to the users of the system as a single computer.
Distributed systems are 'seamless'. the interfaces among functional units on the network are for the most part invisible to the user.



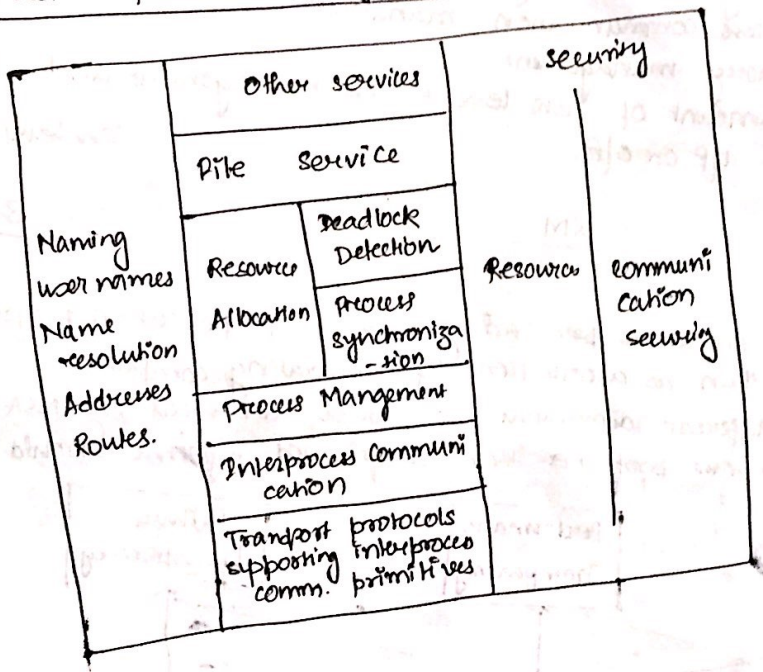
System structure from the physical



logical point of view.

```
class Mcall
{
    static String s[] = {"Java", "is", "fun"};
    static int c;
    public static void main (String args[])
    {
        System.out.println (c);
        System.out.println (s[c]);
        c = c+1;
        if (c == 3)
            System.exit (0);
        main (s);
    }
}
```

Estructure of Distributed System



7.08.2017

SS

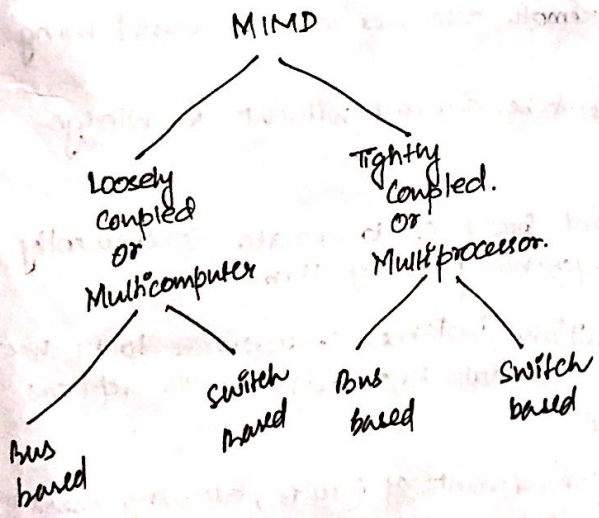
Disadvantages DOS :

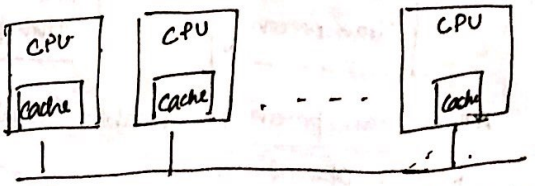
Security, S/W, H/W, Network & problem.

Flynn's Classification:

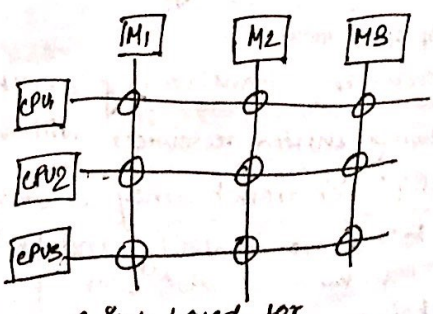
→ Inst stream
→ Data steam

SISD
SIMD
MISD
MIMD.

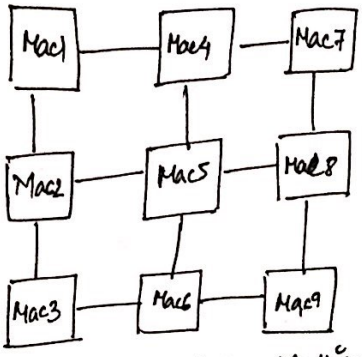




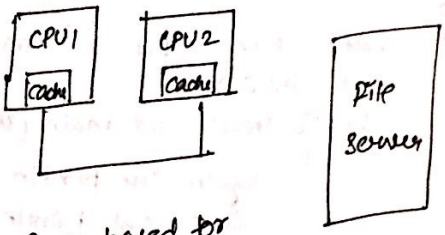
Bus based. Multicomputer



Switch based for Multiprocessor



Switch based for Multicomputer.



Bus based for Multiprocessor.

Micro kernel - The micro kernel is a flexible type of kernel since it does almost nothing. It basically provides four minimal services:

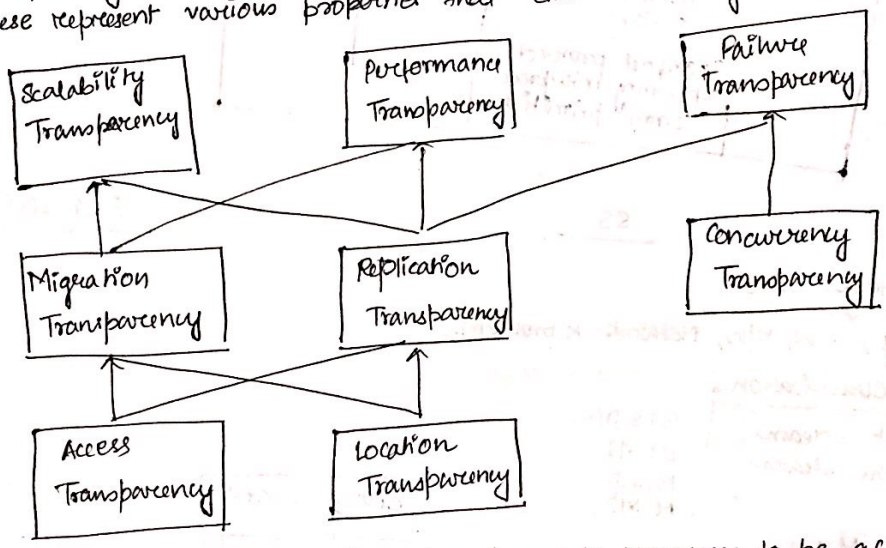
a. An interprocess communication mechanism.
b. some memory management.
c. A small amount of low leve process management and scheduling
d. Low level I/P or O/P. low level func-stub.

8.08.2017

SM

Transparency

Distributed systems should be perceived by users and application programmers as a whole rather than as a collection of cooperating components. Transparency has different dimensions that were identified by ANSA These represent various properties that distributed systems should have.



Access Transparency - enables local and remote resources to be accessed using identical operations.

Location Transparency - enables resources to be accessed without knowledge of their location.

Concurrency transparency - enables several processes to operate concurrently using shared resources without interference between them.

Replication transparency - enables multiple instances of resources to be used to increase reliability and performance without knowledge of the replicas by users or application programmers.

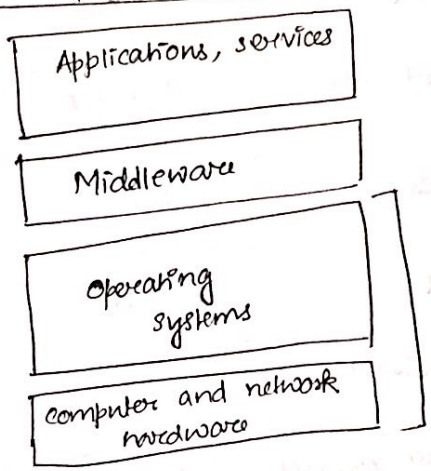
Failure Transparency - enables the concealment of faults, allowing users and application programs to complete their tasks despite the failure of h/w or s/w components.

Mobility Transparency – allows the movement of resources and clients within a system without affecting the operation of users or programs.

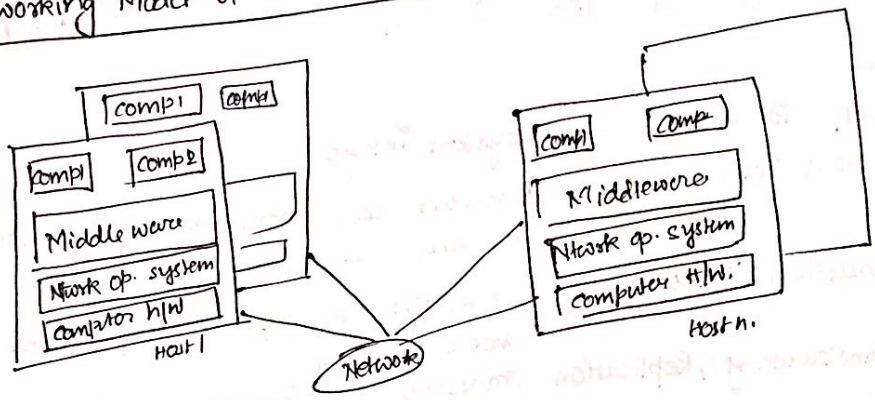
Performance Transparency – allows the system to be reconfigured to improve performance as loads vary.

Scaling Transparency – allows the system and application to expand in scale without change to the system structure or the application algorithm.

S/W & H/W service layers in distributed system



A working Model of a distributed system



Process types in a distributed system:

Filter – data transformer. It receives steams of data values from its input channel performs some computation on those values and sends streams of results to its o/p channels.

Client – It is triggering process. Clients make requests that trigger reactions from servers.

Server – is a reactive process. A server waits for requests to be made, then reacts to them.

Pool: is a collection of identical processes that interact to provide a service to resolve a problem.

There are several process – interaction patterns on distributed system. Each interaction paradigm is an example or model of a communication pattern and associated programming technique that can be used to solve a variety of interesting distributed programming techniques.

10.08.20

ITEM	Network OS	Distributed OS	Multip
Does it look like a virtual Uniprocessor?	No	Yes	Yes.
Do all have to run the same Operating system?	No	Yes	Yes.
How many copies of the OS are there?	No	No	1.
How is communication achieved?	Shared files	Messages	Shared M
Are agreed upon network protocols required?	Yes	Yes	No.
Is there a single run queue? (Job queue).	No	No	Yes.
Does file sharing have well defined semantics?	Usually No	Yes	Yes.
Hardware software relationship	Loosely coupled s/w on Loosecoupled H/W	Tightly coupled s/w on loosely coupled H/W	Tightly c s/w on coupled

Design Issues.

1. Transparency : To acheive single system image.
   a. Migration Transparency – Resources can move at will without changing their names.
   b. Parallelism Transparency – Activities can happen in parallel with users knowing.

Location Transparency, Replication Transparency, Concurrency Transparency

SM.

11.08.2017

IPC- Interprocess Communication is a set of methods used for the exchan of data among multiple threads in one or more processes. The processes r be running on one or more computers connected by a network. The IPC methods are divided into methods for message passing, synchronisation, sha memory and remote procedure calls (RPC). The method of IPC used m vary based on the bandwidth and latency of communication between threads and the type of data being communicated. There are several reasons for providing an environment that allows process cooperation They are:

- Information sharing
- Computational ~~sharing~~ speedup
- Modularity
- convenience
- Privilege separation

The term IPC may also be reffered to as inter-thread communicatio

components.

Different Methods of IPC :

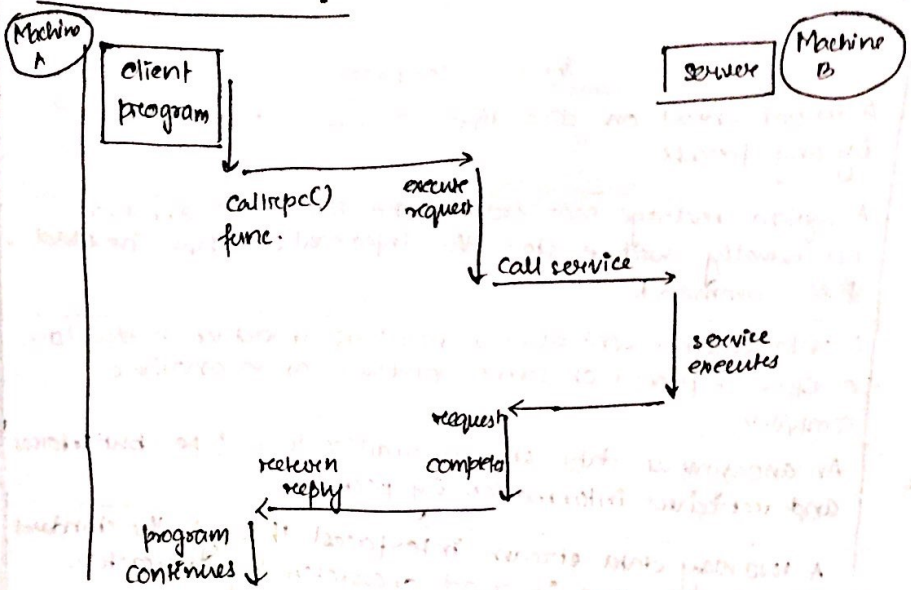
Method	Short Description.
1. File	A record stored on disc that can accessed by the name by any process
2. Signal	A system message sent from one process to another not usually used to store the information but instead gives command.
3. Socket	A data stream sent over a network interface either to a different process on same computer or to another computer.
4. Message Queue	An anonymous data stream similar to a pipe but stores and retrieves information by packets.
5. Pipe	A two way data stream interfaced through the standard I/P and O/P and is read character by character.
6. Named Pipe.	A pipe that is implemented through a file on the file system instead of the standard I/P and O/P.
7. Semaphore	A simple structure that synchronises the threads or processes acting on the shared resources.
8. Shared Memory	Multiple processes give an access to the same memory allowing all the processes to modify it and read change made by other processes.
9. Message Passing (Shared Nothing).	It is similar to the message queue where the messages are not shared to other processes.
10. Memory Mapped File.	A file mapped to the RAM and can be modified by changing the memory addresses directly instead of giving the O/P to a stream sharing the same benefits as a standard file

IPC enables the data communication by allowing processes to use segments semaphores and other methods to share memory and information. IPC facilitates efficient message transfer between processes. The idea of IPC is based on task control architecture (TCA). It is a flexible technique that can send and receive variable length arrays, data structures and lists. It has the capacity of using client server data transfer paradigms while supporting a wide range of OS and languages.

Q. What is RPC? (Remote Procedure Call).

RPC is a powerful technique for constructing distributed client server based application. It is based on extending the notion of conventional or local procedure calling so that the called procedure need not exists in the same address space as the calling procedure. The two processes may be residing on the same system or they may be on different system with a network connecting them. By using RPC the programmers of distributed applications avoid the details of interface with a network. The transport independence of RPC isolates the application from the physical and logical elements of the data communication mechanisms and allows the application to use a variety of transport.

Q. How RPC works?



An RPC is analogous to a function call where the calling arguments are passed to the remote procedure and the caller waits for a response to be returned from the remote procedure. The fig shows the flow of activity that takes place during an RPC call between two networked systems. The thread in this case is blocked from processing until either a reply is received or its times out. When the request arrives the server calls a dispatched routine that performs the requested routine and sends the reply to client. After RPC call is completed the client program continues. RPC specifically supports network applications.

17.08.2017

SS

2. Flexibility: Who could possibly be against them.?

Two schools of thought based kernel:

i) Monolithic kernel – basically centralised OS augmented with networking facilities and the integration of remote services.

ii) Micro kernel – Is more flexible than the other one. *

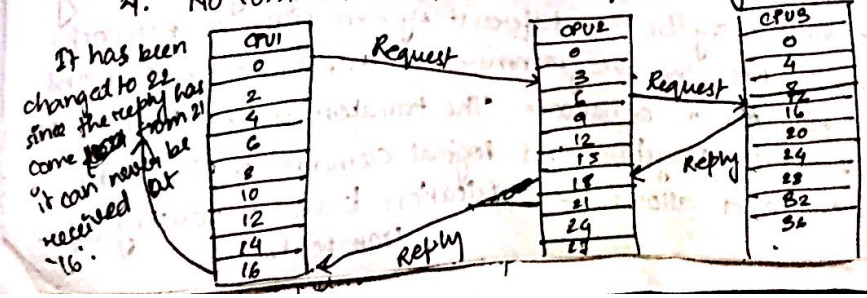
3. Reliability – highly available, security, fault tolerance.
4. Performance
5. Scalability.

* It does almost nothing, provides four minimal services:

a) An inter process communication mechanism.
b) Some memory management.
c) A small amount of low-level process management and scheduling.
d) Low-level input/output

Clock Synchronisation:

1. The relevant information is scattered among multiple machines.
2. Processes make decisions based only on local information.
3. A single point of failure in the system should be avoided.
4. No common clock or other precise global time source exists.



Logical Clock

These are not actually clocks in the usual sense.

Associated with each crystal are two register -

Counter Register - crystal decrements by one.

Holding register - It holds the counter's initial value, when an interrupt occur.

Clock Tick - Each interrupt.

Clock Skew - When a system has n computers, all n crystals will run at slightly different rates causing the clocks gradually to get out of synchronization and give different values when read out. The difference in time is called clock skew.

Physical Clock

With logical clock, if we add the additional constraint is present that the clocks must not only be the same, but also must not deviate from the real time by more than a certain amount, the clocks are physical clocks.

18.08.2017

SM

Fault Tolerance in Distributed System:

A system ability to tolerate failure - 1.

- Reliability - the likelihood that a system will remain operational for the duration of a mission.
  - requirement might be stated as 0.999999 availability for a 10 hr mission
  → the probability of failure during the mission must be at most 10^{-6}
  - Very high reliability is most important in critical applications - space shuttle, industrial control, in which failure mean loss of life.
- Availability - expresses the fraction of time a system is operational
  - a 0.999999 availability means the system is not operational at most one hour in a million hours.
  - a system with high availability may in fact fail its recovery time and failure frequency must be small enough to achieve the desired availability
  - high availability is important in airline reservations, telephone switching etc, in which every minute of downtime translates into revenue loss.

Importance of Design

A good fault-tolerant system design requires a careful study of failures, causes of failures, and system responses to failures

Such a study should be carried out in detail before the design begins and must remain part of the design process.

Requirement specification:

- Planning to avoid failures is most important.
- A designer must analyze the environment and determine the failures that must be tolerated to achieve the desired level of reliability.
- To optimize fault tolerance, it is important to estimate actual failure rate for each possible failure.

Three types of ~~failures~~ Faults - Transient failure - which may come and disappear in 2nd implementation

Intermittent failure - appear/disappear/appear.

* Permanent failure (until and unless rectified the errors will persist).

Design

- Design of systems that tolerate faults that occur while system is in use.
- Basic Principle - Redundancy.
  - Spatial - redundant hardware.
  - Informational - redundant data structures
  - Temporal - redundant computation.
- Redundancy costs money and time.
  - One must optimize the design by trading off amount of redundancy used against the desired level of fault tolerance.
  - Temporal redundancy usually requires recomputation and it results in a slower recovery from failure.
  - Spatial has faster recovery but increases hardware costs, space, power etc requirements.
- Commonly used Techniques for Redundancy.
  - Modular Redundancy
    - Uses multiple, identical replicas of hardware modules and a voter mechanism.
    - The outputs from the replicas are compared, and correct output is determined - majority vote.
    - Can tolerate most hardware faults that can affect the minority of the hardware modules.
- N-version Programming.
  - Write multiple versions of s/w module.
  - Outputs from these versions are received and correct output is determined via voting mechanism.
  - Each version is written by different team with the hope that they will not contain the same bugs.
  - Can tolerate s/w bugs that affect a minority of versions.
  - Can tolerate correlated fault - reason for failure is common to two (or more) modules eg two modules share a single power power supply, failure of which causes both to fail.

SS

21.08.2017

Happens-Before Relation

To synchronize logical clock, Lamport define this relation. The expression $a \rightarrow b$ is read as "a happens before b" and means that all processes agree that first event a occurs, then afterward event b occur.

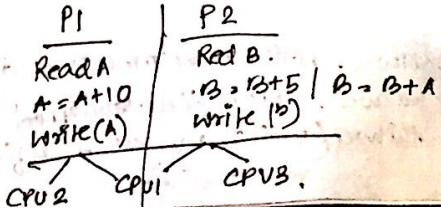
If a and b are events in the same process, and a occurs before b, then $a \rightarrow b$ is true.

If a is the event of a message being sent by one process and b is the event of the message being received by another process then $a \rightarrow b$ is true.

If two events a, b do not exchange messages then $a \rightarrow b$ is not true.

This is transitive relation $a \rightarrow b$, $b \rightarrow c$ then $a \rightarrow c$.

1. If a happens before b is the same process, $c(a) < c(b)$.
2. If a is the event of a message being sent by one process and b is the event of the message being received by another process then $c(a) < c(b)$
3. If two events a, b do not exchange messages then $c(a) \neq c(b)$.



Mutual Exclusion:

1. Centralised Algorithm: a) One process is elected as the coordinator (eg. the one running on the machine with the highest network address).
b) Whenever a process wants to enter a critical section, it sends a request message to the coordinator stating which critical region it wants to enter and asking for permission.
c) If no other process is cwrcently in that critical region, the coordinator sends back a reply granting permission.
d) When reply arrives, the requesting process enters the critical region.
e) If more than one process wants to enter critical region, then one process come first gets the permission and others request insert into a queue maintained by the coordinator.
f) When a cwrcent process gets out from a critical section, one of the process's message will be selected from the queue depend on their timestamp value.

Advantages – 1. No process ever waits forever (no starvation)
2. The scheme is easy to implement.

Disadvantages – 1. The coordinator is a single point of failure, so if it crashes, the entire system may go down.
2. If processes normally block after making a request, they cannot distinguish a dead coordinator from "permission denied" since in both cases no message comes back.
3. In a large system, a single coordinator can become a performance bottleneck.

22.08.2017

SM

* Failure type – Some are more probable than others
Some are transient, other permanent
some occur in h/w, other in s/w.

Error-Control coding

- Replication is expensive
- For certain applications - RAM, Buses, error correcting codes can be used.
  - Hamming or other codes.
- Checkpoints and rollbacks
- A checkpoint is a copy of an application's state saved in some storage that is immune to the failures under consideration.
- A rollback restarts the execution from a previously saved checkpoint.
- When a failure occurs, the application's state is rolled back to the previous checkpoint and restarted from there.
- Can be used to recover from transient as well as permanent h/w failure
- can be used for uniprocessors and distributed applications.

Recovery blocks

- Uses multiple alternates to perform the same function.
- One module is primary others are secondary.
- When primary completes execution, its outcome is checked by an acceptance test.
- If the o/p is not acceptable, a secondary module executes and so on until either an acceptable o/p is obtained or alternates are exhausted.
- This method can tolerate s/w failures, because alternates are usually implemented with different approaches (s/w algorithms).

Dependability Evaluation

- Once a system has been designed, it must be evaluated to determine if it meets reliability and dependability objectives.

- Two dependability approaches:
  - Use an Analytical Model
    - Can help developers to determine a system's possible states and probabilities of transitions among them.
    - can be difficult to analyze models accurately.
  - Injecting faults
    - various types of faults can be injected to determine various dependability metrics.
- In distributed system a transaction based service can accept occasional failures followed by a lengthy recovery procedure.
- A real time service - Process control.
  - may have inputs that are readings taken from sensors
  - may have o/p to actuators that are used to control a process directly or to activate alarms so that humans can intervene in the process.
  - due to strict timing requirements, recovery must be acheived within a very small time limit eg. air traffic control, monitoring patients, controlling reactors.
- A Fault-Tolerance service
  - For a service to perform correctly both the effect on a server's resources and the response sent to the client must be correct
  - Correct behaviour must be specified.
  - Failure semantics - the ways in which the service can fail, must be specified.
  - can detect faults, thus
    - > fails predictably
    - > masks faults from its users
    - > operates in the presence of faults in services on which it depends.

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

1. Robustness - refers to the powers of a file to survive crashes of storage device and decays of the storage medium on which it is stored.
2. Recoverability - refers to the ability to be rolled back to the earlier consistent state when an operation of the file fails or aborted by the client.
3. Availability - refers to the fraction of time for which the file is available for use. (there should not be any connection failure when connected to a remote machine).

Fault Models

- Omission Failure
  - A server omits to respond to a request or receive request.
- Response Failure
  - value failure - returns wrong value
  - State transition failure - has wrong effect on resources
- Timing Failure - any response that is not available to a client within a specified real time interval.
- Server Crash Failure - a server repeatedly fails to respond to requests until it is restarted.
  - Amnesia-Crash - a server starts in its initial state, having forgotten its state at the time of the crash, i.e loses the values of the data items.
  - Pause crash - a server restarts in the state before the crash.
  - Halting crash - server never restarts.

Distributed Algorithm

1. When a process wants to enter a critical region, it builds a message containing name of the critical region it wants to enter, its process number and the current time.
2. It then sends the message to all other process, conceptually including itself. The sending of messages is assumed to be reliable.
3. When a process receives a request message from another process, the action it takes depends on its state wrt the critical region named in the message. The cases have to be distinguished:-
   a. If the receiver is not in the critical region and does not want to enter it it sends back an OK message to the sender.
   b. If the receiver is already in the critical region, it does not reply. Instead it queues the request.
   c. If the receiver wants to enter the critical region but not yet done so it compares the timestamp in the incoming message with the one contained in the message that it has sent everyone. The lowest one wins.
4. After sending out requests asking permission to enter a critical region, a process sits back and waits until everyone else has given permission. As soon as all permissions are in, it may enter the critical region.
5. When it exists the critical region, it sends OK messages to all processes on its queue and deletes them all from queue.

Disadvantages - 1. The num of messages required per entry is $2(n-1)$ where the total num of processes in the system n.
2. If any process crashes, it will fail to respond to requests. This silence will be interpreted as detail of permission, thus blocking all subsequent attempts by all processes to enter all critical regions.
3. This method is good for all small groups.
4. Load is another issue

Token Ring Algorithm

1. In software, a logical ring is constructed in which each process is assigned a position in the ring.
2. The ring positions may be allocated in numerical order of network addresses or some other means.
3. It does not matter what the ordering is, but each process must know who is next in line after itself.
4. When the ring is initialized, process k is given a TOKEN, that circulates around the ring from k to process k+1
5. When a process aquires the token from its neighbour, it checks to see if it is attempting to enter a critical region.
6. When it has exited, it passes the token to the next one, so that the next process waiting to enter critical region can enter.

Disadvantages - 1. If the token is ever lost, it must be regenerated. In fact, detecting that it is lost is difficult, since the amount of time bw successive appearances of the token on the network is unbounded
2. If a process crashes, but recovery is easier than in the other cases.

Algo	Message per entry / exit	Delay before entry	Problems
Centralized	3	2	Coordinator Crash
Distributed	$2(n-1)$	$2(n-1)$	Crash of any process
Token Ring	1 to ∞	0 to $n-1$	Lost token, Process Crash.

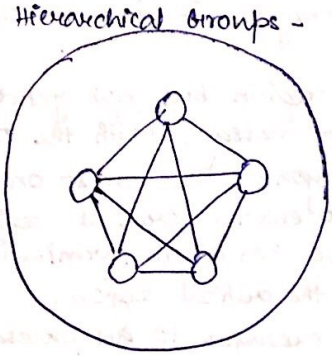
Fault Example

- UDP service
  → has omission failures because it occasionally loses messages
  - does not have value failures because it does not transmit corrupt messages.
    > UDP uses checksums to mask the value failures of the underlying IP by converting them to omission failures.

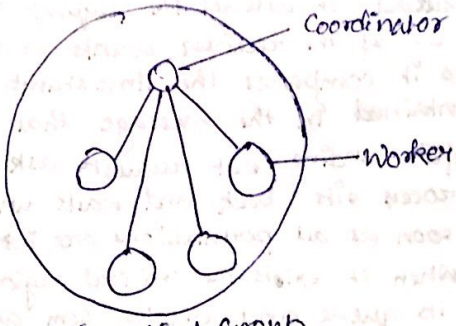
Process Resilience

Protection against process failures can be achieved through process replication into groups.

→ Flat Groups – All the members are equal.
→ Hierarchical Groups – There is a coordinator.



Flat Group



Hierarchical Group

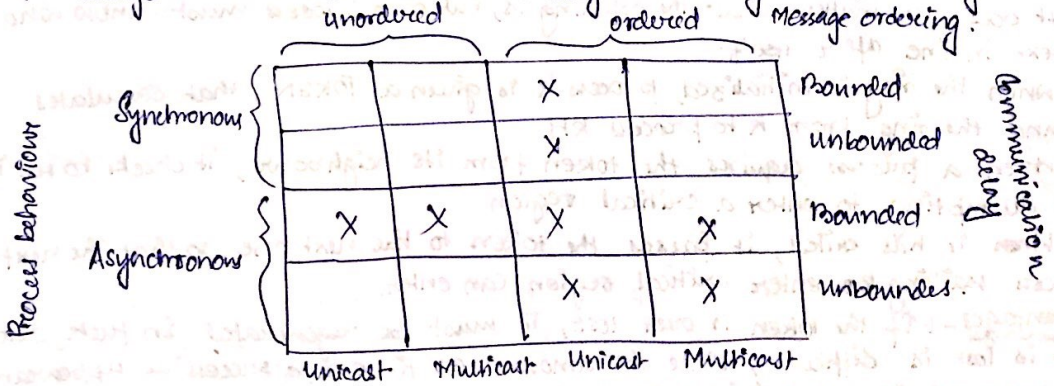
Failure Masking

- In most simple fault case, k+1 processes provide k fault tolerance.
- If the faulty processes continue to run, providing faulty response but do not team up to give wrong response, then to have k tolerant system we need 2k+1 processes.
- Assume all the messages arrive at all nodes at the same time.

Agreement issue in Faulty Systems

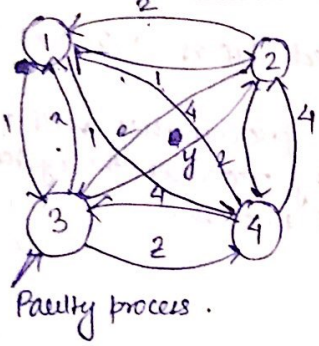
Possible assumptions about the underlying system:

1. Synchronous versus asynchronous systems.
2. Communication delay is bounded or not.
3. Message delivery is ordered or not.
4. Message transmission is done through unicasting or multicasting.



Byzantine Agreement Problem: Lamport et al. 1982

- Assume reliable synchronous ordered unicast based message system. There are N processes, k of which may act as faulty or even malicious. A faulty process may send different values of different processes.
- These arbitrary failures may be loosely categorized as follows:
  - a failure to take another step in the algorithm, also known as a crash failure.
  - a failure to correctly execute a step of the algorithm.
  - arbitrary execution of a step other than the one indicated by the algorithm.



1 Got (1,2,x,4)
2 Got (1,2,y,4)
3 Got (1,2,3,4)
4 Got (1,2,2,4).

1 Got	2 Got
(1,2,y,4)	(1,2,x,4)
(a,b,cd)	(e,f,g,h)
(1,2,3,4)	

SS

Election Algorithm

28-08-2017

In general, it does not not matter which process takes on this special responsibility but one of them has to do it.

1. Bully Algorithm
2. Ring Algorithm.

The goal of an election algorithm is to ensure that when an election starts, it concludes with all processes agreeing on who the new coordinator is to be.

Bully Algorithm

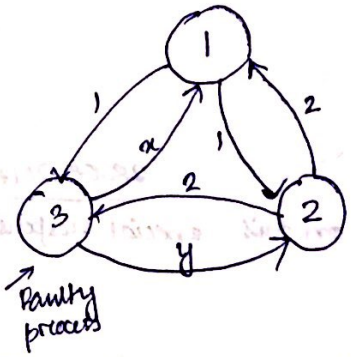
1. When a process notices that the coordinator is no longer responding to requests it initiates an election. A process P holds an election as follows:
   a) P sends an Election message to all processes with higher numbers.
   b) If no one responds, P wins the election and becomes coordinator.
   c) If one of higher-ups answers, it takes over. P's job is done.
2. At any moment, a process can get an election message from one of its lower numbered colleagues. When such a message arrives, the receiver sends OK message back to the sender to indicate that he is alive and will take over. The receiver then holds an election, unless it is already holding one.
3. Eventually all processes give up but one, and that one is the new coordinator. It announces its victory by sending all process a message telling them that starting immediately it is the new coordinator.
4. Thus the biggest one in town always win, hence the name BULLY ALGORITHM.

Ring Algorithm

1. This one based on the use of a ring, but without a token. We assume that the process physically or logically ordered, so each process knows who is successor of it.
2. When any process notices that the coordinator is not functioning, it builds an election message containing its own process number and sends the message to its successor. If the successor is down, the sender skip over the successor and goes to the next member along the ring or the one after that, until a running process is located. At each step, the sender adds its own process number to the list in the message.
3. Eventually the message gets back to the process that started it all. That process recognizes this event when it receives an incoming message containing its process number.
4. At that point, the message type is changed to Coordinator and circulated once again, this time to inform everyone else who the coordinator is and who are the members of the new ring.
5. When this message has circulated once, it is removed and everyone goes back to work.
6. If more than one message circulates within the ring, when all have removed again, both will be removed. It does no harm to have extra messag... a little bandwidth.

Byzantine Agreement Problem:

- Three processes can agree on the values received from 1, 2 and 4. so that malicious process 3 value is irrelevant....
- If N=3 and k=1, that is only two non-faulty process this will not work!
- Lamport proved that, with 2k+1 non faulty processes, the system will survive k faulty processes, which makes a total of 3k+1 process...



1 Got (1, 2, x)
2 Got (1, 2, y)
3 Got (1, 2, 3).

1 Got
(1, 2, z)
(a, b, c)

2 Got
(1, 2, z)
(d, e, f).

Client-Server communication

- TCP - Transport Control Protocol as a connection oriented end-to-end communication protocol is a reliable protocol. But it does not prevent connection crash failures, which requires searching for new connections.

RPC semantics in the Presence of Failures:

- Five different classes of failures that can occur in RPC systems.
  - The client is unable to locate the server
  - The request message from the client to the server is lost.
  - The server crashes after receiving a request. This can be handled with principles such as:-
    - At least once
    - At most once
    - The preferred principle is exactly once, which is virtually impossible to implement.
  - The reply message from the server to the client is lost.
  - The client crashes after sending a request.
- Each case needs to be resolved properly to mask the failures.

Failure Examples Reliable Multicasting

- Use negative acknowledgement, known as scalable reliable multicasting - SRM.
- Non-hierarchical and hierarchical solutions are possible.
- Atomic multicasting requires all the replicas reaching agreement on the success or failure of multicast. This is known as distributed commit: two phase or three-phase commit protocols can be used.

Recovery from a failure

- When and how the state of a distributed system be recorded and recovered by means of check-pointing and logging.
- To be able to recover to a stable state, it is important that the state is safely stored.

Stable storage

- stable storage is an example of group masking at the disk block level.
- Designed to ensure permanent data is recoverable after a system failure, a disk write operation or after a disk block has been damaged.
- Provided by a Careful storage service.
  - Unit of storage is the stable block.
  - Each stable block is represented by two normal

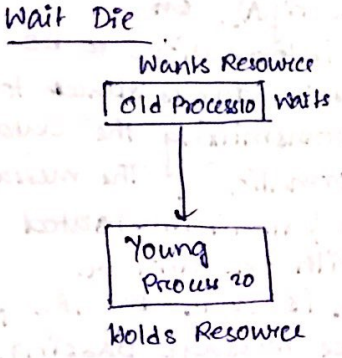
- Write operation writes one careful block ensuring it is correct before writing the second block.
- Careful blocks are disk blocks ~~stored~~ stored with a checksum to mask values failures, the blocks are located on different disk drives with independent failure modes.
- Value failures are converted to ommision failures.
- The Read operation reads ~~to~~ one of the pair of careful blocks, if an ommision failure occurs then it reads the other, thus masking the ommision failures of the careful storage service.

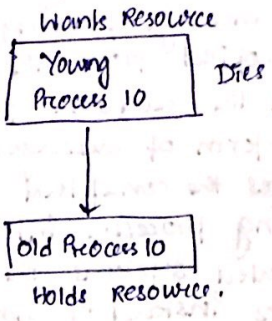
Q1. What are the characteristics of good fault tolerance system?
Q2. What are the principle of redundancy in fault tolerance design?
Q3. Difference between Flat Group and Hierarchical group.
Q4. Agreement issues in faulty system.
Q5. Byzantine failure (3 or 4 processes).

31.08.2017

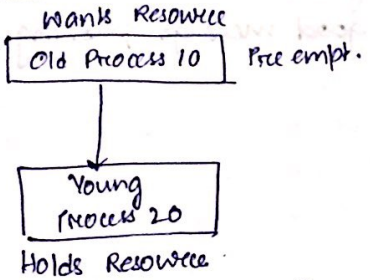
SS

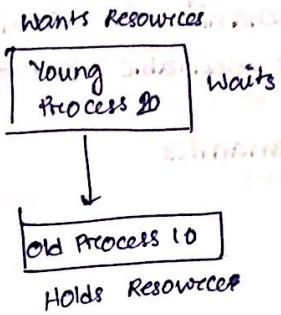
Wait Die





Wound Wait





Distributed Deadlock Detection:
- Centralised Detection
- Distributed Detection.

Centralised Deadlock Detection:

1. Each machine maintains the resource graphs for its own processes and resources.
2. There must be a central coordinator which also maintains the resource graphs for the entire system: whenever an arc is added or deleted from the resource graph a message can be sent to the coordinator providing the update.
3. Periodically every process can send a list of arcs added or deleted since the previous update.
4. When the coordinator detects a cycle, it kills off one process to break the deadlock.

Drawback - False Deadlock.

How to break the deadlock?

There are number of ways in which the deadlock can be broken -

1. The process that initiated the probe commit suicide. [Problem: If several processes do it simultaneously, i.e overkill].
2. Each process add its identity to the end of the probe message so that when it returned to the initial sender, the complete cycle would be listed. The sender can [illegible] has the highest number and kill that one or send it a

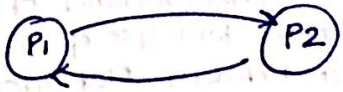
SM.

Message Passing

IPC requires information sharing among two or more processes. Two basic methods of information sharing are:

1. Original sharing or shared data approach.
2. copy sharing of message passing approach.

(A)



(B).

In shared data approach the information to be shared is placed in a common memory area accessible to all the processes in an IPC. Fig(A). On the other hand in message passing approach the information to be shared is physically copied from the sender's process's address spaces to the adress space of all the receiver's processes. By transmitting the data to be copied in the form of messages. (block of information). The message passing paradigm gives the conceptual communication pattern illustrated in Fig B - Communicating processes interact directly with one another. Normally in distributed systems the communication is done by the is. exchange of messages instead of shared data. Hence message passing the basic IPC mechanism in distributed system.

Q. What are the desirable features of a good message passing system?

- Simplicity
- Uniform semantics
- Efficiency
- Reliability
- Correctness
- Flexibility
- Security
- Portability.

1. A message passing system should be simple and easy to use. It must be straightforward to construct new applications and to communicate with existing ones by using the primitives provided by message passing. The system should be modular which helps to send and receive messages between the modules in a simpler way. Without worrying about the network aspects of the system.

2. A message passing system is used for two types of IPC
   - Local Communication - Communicating processes residing on same node.
   - Remote Communication - Communicating processes are on different node.

The uniform semantics should be very close to local communications for ensuring that message passing system is easy to use.

3. Efficiency is a critical issue for message passing system to be acceptable by the user. If message passing system is inefficient the IPC may become so expensive that the application designers will try to avoid its use in their applications. As a result the developed abblication

of message exchanges as far as practical during the communication. Some important aspects normally adopted for efficiency are.
- Avoiding costs of establishing and terminating connection between the same pair of processes for each and every message exchange between them.
- Minimizing the cost of maintaining the connections.
- Piggy backing of acknowledgement for ~~p~~ of previous messages with next messages during the connection b/w sender and receiver that involves several message exchanges.

4. Distributed systems are prone to catastrophic events such as node crashes or communication link failures. Such events may interrupt a communication that was in progress between two processes resulting in the loss of message. A reliable IPC protocol can cope with failure problem and guarantees the delivery of a message. Handling of a lost message involves acknowledgement and retransmission on the basis of timeouts. Another issue related to reliability is that of duplicate messages. Duplicate messages may be sent in the event of failures or because of timeouts. A reliable IPC protocol is ~~eat~~ also capable of detecting and handling the duplicate that involves generating and assigning appropriate sequence no. to the messages

5. A message passing system has IPC protocols for group communication that allows a sender to send the message to a group of receiver and a receiver to receive the messages from a group of senders. Correctness is a feature related to IPC protocol for group communication. Correctness sometimes may be useful for some applications related to -
a. Atomicity b. Ordereddelivery c. Survivability.

a. Atomicity ensures that every message sent to a group of receivers will be delivered to either all of them or none of them.

b. Ordered Delivery ensures that messages arrived at all the receiver in an order acceptable to the applicant.

c. Survivability - guarantees the messages correct delivery despite partial failures of process, machines or communication links. It is a difficult property to achieve

6. Different application require different degree of reliability and correctness for the IPC protocols. A broadcast ~~message~~ protocol can be used ~~in~~ for adaptive routing for queuing delays in different parts of networks. In this case the broadcast message may be a bit late due to communication failures. But it might be outdated by a more recent protocol. Atomicity of the message delivery is not required in some cases but becomes essential for other communications. So, IPC protocols of message passing system must be flexible enough to cater the various needs of different applications and gives the users the choice of specific types and levels of reliability and correctness requirements of their applications

7. A good message passing system must be capable of providing a secure end to end communication. This is a message in transit on the network should not be accessible to any user other than those to whom it is addressed. 3 steps necessary for ~~sec~~ secured communication -
- Authentication of receivers of a message by sender.
- Authentication of sender of a message by receiver.
- Encryption of message before sending it over the network.

1. The message passing system should itself be portable i.e. it should be possible to easily construct a new IPC facility on another system by reusing the basic design of the existing message passing system.

2. Applications to written by using the primitives of IPC protocols of message passing system should be portable. This requires heterogeneity considered while designing a message passing system.

04.09.2017

SS

Distributed Deadlock Detection (Chandy – Misra Haas Algo).

1. Processes are allowed to request multiple resources at once, instead one at a time.
2. No need to maintain a centralized coordinator.
3. When a process waits for a resource, at that point the process generate a special PROBE message and sent to the process holding the needed resources. The message consists of three numbers: the process that just blocked, the process sending the message and the process to whom it is being sent.
4. When the message arrives, the recipient checks to see if it itself is waiting for any processes. If so, the message is updated, keeping the first field but, replacing the second field by its own process number and the third one by the number of the process it is waiting for.
5. The message is then sent to the process on which it is blocked. If it is blocked on multiple processes all of them are sent messages.
6. If a message goes all the way around and come back to the original sender, that is, the process listed in the first field, a cycle exists and system is deadlock.

Processor Allocation

These can be divided into two different classes –

1. Non migratory: When a process is created, a decision is made about where to put it. Once placed on a machine, the process stays there until it terminates.
2. Migratory: A process can be moved even if it has already started execution.

- Response Time.
- Response Ratio – It is defined as the amount of time it takes to run a process on some machine, divided by how long it would take on some unloaded benchmark processor.

SM

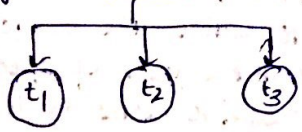
5.09.2017

THREAD

It is lightweight process. Multiple threads comprise of one single process.

Related to java; thread is a class.

Runable interface

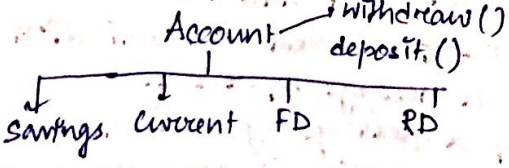


(java.)

Interface holds only method declarations (all methods must be abstract)

Class have method ~~declar~~ definition and declaration

Abstract class – must have one abstract method. (at least one method abstract)



abstract void withdraw (doubl

Runable interface –

```
MyThread t1 = new MyThread();
t1.start();
}
```

MAX_PRIORITY – 10
MIN – PRIORITY – 1
NORM_PRIORITY – 5

t1.setPriority(5). get priority.

~~resumeAll~~ → resumeAll → to resume all the threads which are suspended.

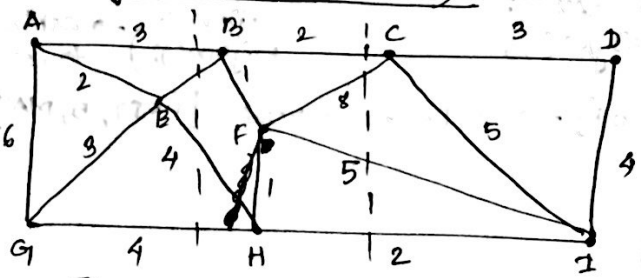
SS 7.09.2017

Q. Consider there are 2 diff processor & 2 processes. Processor 1 runs at 10 MIPS Processor 2 runs at 100 MIPS. But it has a waiting list of backlog processes that will take 5sec to finish. Process A has 100 Million & Process B has 800 million inst. Find the response time for each processes on each processor.

Processor 1	Processor 2
10 mips	100 mips. + 5 sec.
Process A 100 million inst.	3+5 = 8 sec } 18
∴ 10 sec taken by it	1+5 = 6 sec } 36.
P(B) = 800 → 30.	

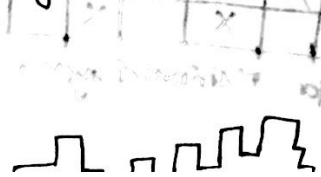
2017
-1995
22

Graph Based algorithm – (Deterministic)

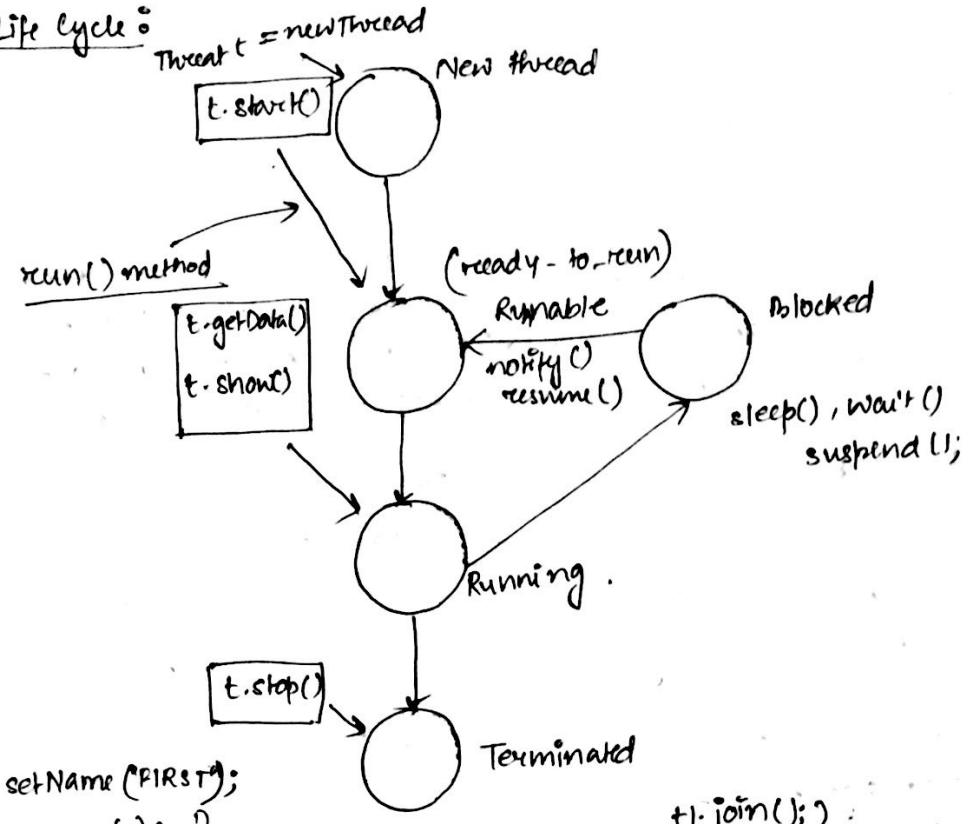


Edges ~~are~~ weights represents the ~~com~~ number of communication reqd between two vertices.

A heuristic centralised algorithm known as Up Down Algorithm. In this algo a coordinator maintain a usage table with one entry per computer initially zero. When a process is to created and the machine it is created on decides that the process should be run elsewhere it asks the usage table coordinator to allocate it a processor. If there is one available and no one else wants it the request is granted. If no processors are free the request is temporarily denied and a note is made of the request. When a machine is running processes on other's machines it accumulates penalty points, a fixed number per sec. These points are added to it usage table entry when it has unsatisfied requests pending penalty points are subtracted from the usage table. When requests are pending and no processors are being used the usage table entry is moved a certain number of points closer to 0 until it gets there. Usage table entry can be of positive negative or zero. A positive score indicates that the workstation is a net user of system resources, whereas a negative score means it need resources and a zero score is neutral. When a processor becomes free the pending request whose owner has the lowest score wins. As a consequence a user who is occupying no processors and who has had a request pending for a long time will always beat someone who is using many processors.



Thread Life cycle :



t1.setName(FIRST);
t1.resume(); } resumeAll();
t1.notify(); } notifyAll();

t1.join();
t2.join(); } → asking the thread to complete the task.
t3.join();

[Threadname, Priority, Parent] → To print a thread. [FIRST, 5, MAIN].

FIRST
↓
A1 → [A1, 5, FIRST]

class A1 extends FIRST implements Runnable, ~~Runnable~~ A, B.
{
}.

Daemon Thread → Thread t1 = new Thread();
t1.setDaemon();
t1.isDaemon();

When all threads complete the jobs its the job of Daemon Thread to ~~look~~ verify that every thread have completed the task successfully.

18.09.2017

8s

Sender and receiver initiated algorithm

Time slot	0	1
0	A	C
1	B	D
2	A	C
3	B	D
4	A	C
5	B	D

Processor

Time slot	0	1	2	3	4	5	6	7
0	X				X			
1			X			X		
2		X			X		X	
3	X					X		
4		X		X				X
5			X		X			

On chip Memory - 8155 memory unit.

Bus base Multiprocessor - with cache
without cache

Snoopy Cache - Type of cache that snoops the cache when there is write back or write through operation ~~command~~ in the bus. which updates the value in file server.

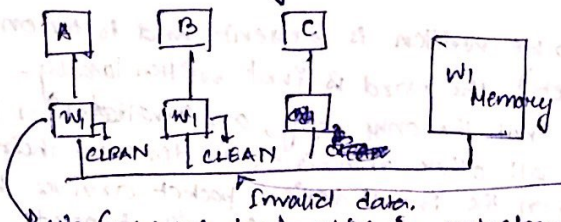
Write Through

Write back/one

Cache Ownership Protocol.

This protocol manages cache blocks, each of which can be in one of the following three states:

1. INVALID - This cache block does not contain valid data.
2. CLEAN - Memory is up-to-date; the block may be in other caches.
3. DIRTY - Memory is incorrect; no other cache holds the block.



W_2 (updated data) which in not clean data but dirty data.

Very much effective for small set of processes.

21.09.2017

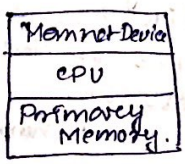
SS

Ring Based Multiprocessor - Memnet

→ A single address space is divided into a private part and a 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 for all machines and is kept consistent by a h/w protocol roughly similar to those used in bus-based multiprocessor systems.
→ Shared memory divided into 32 byte blocks.
→ All the machines are connected together in a token passing ring. The ring consists of 20 parallel wires, which together allow 16 data bits and 4 control bits.
→ The ring interface with the MEMNET device of the machine which consists of MMU, cache, Local Memory and Block table.

Architecture.

Each 32 byte block in the shared address space has a home machine on which physical memory is always reserve for it.



A block may be cached on a machine other than its home machine.
A read only block may be present on multiple m/c, where as write only block may be present on only m/c.

The Memnet device on each m/c contains a table, contain the following info:

	Valid	Exclusive	Home	Interrupt.	Location
Block	This bit telling whether the block is present in the cache and up	specifying whether the local copy if any is the only one.	which is set only if this is the block's home m/c	used for forcing interrupts	This tells where the block is located in the cache if it is present and valid.

Protocol

Read:

When the CPU wants to read a word from shared memory, the memory address to be read is passed to the Memnet device, which checks the block table to see if the block is present.

1. If so, the request is satisfied immediately.
2. If not, the Memnet device waits until it captures the circulating token,
3. then puts request packet onto the ring and suspends the CPU.
3. The request packet contains the desired address and a 32 byte dummy field.

As the packet passes around the ring each device checks that it has the block or not. If so, it puts the block in the dummy field and modifies the packet header. If the block's Exclusive bit is set, it is cleared.

Prob: If the requesting m/c does not has no free space in its cache to hold the incoming block, so it picks randomly a cached block and sends it to home. But blocks whose Home bits are set are never chosen since they are already at home.

Write Protocol

1. If the block containing the word to be written is present and is the only copy in the system (i.e, the Exclusive bit is set) the word is just written locally.
2. If the block is present but it is not the only copy, an invalidation packet is first sent around the ring to force all other machines to discard their copies of the block about to be written. When the invalidation packet arrives back at the sender, the Exclusive bit is set for that block and the write proceeds locally.
3. If the block is not present, a packet is sent out that combines a read request and an invalidation request. The first machine that has the block copies it into the packet and discards its own copy. All subsequent machines just discard the block from their caches. When the packet comes back to the sender, it is stored there and written.

Bus based protocol - Tightly coupled protocol

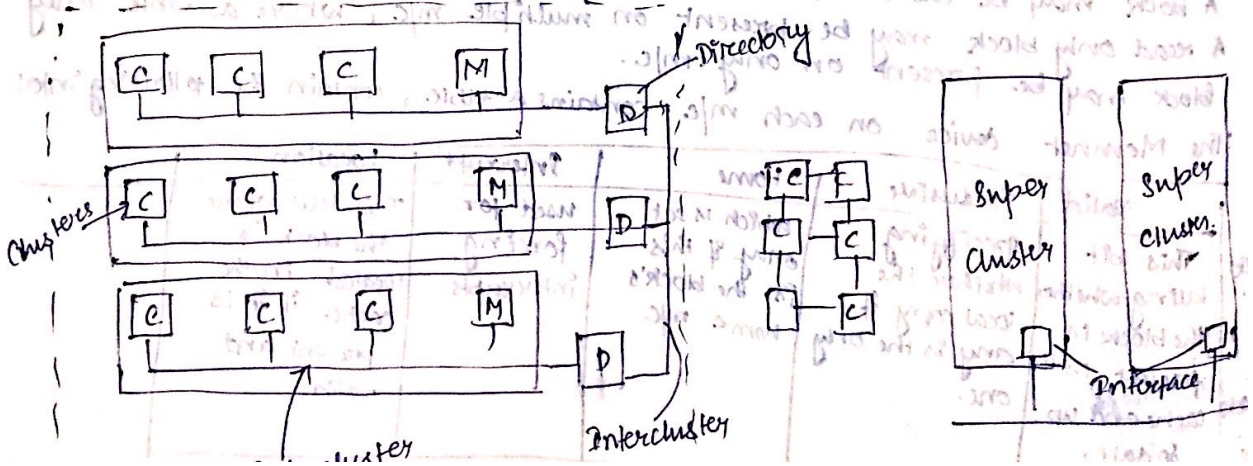
Ring based protocol - Loosely coupled protocol.

Switched Multiprocessor Protocol

26.10.17

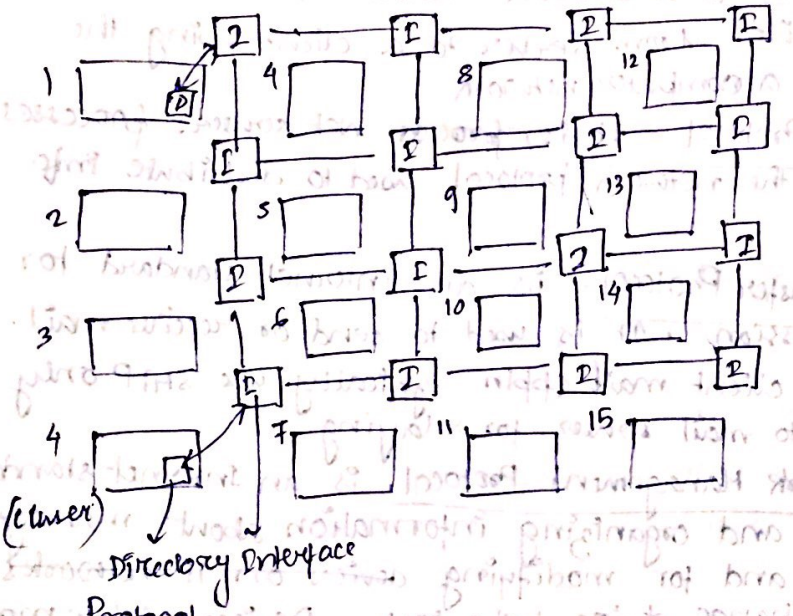
This is basically a Hierarchical method:

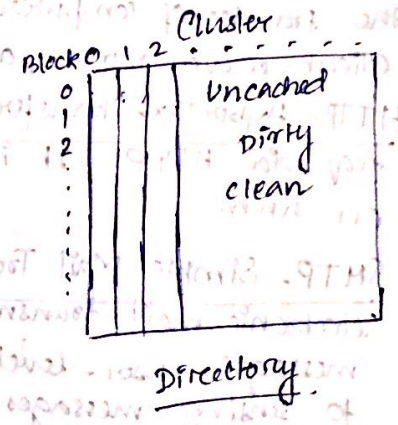
1. Some number of CPU on a single bus, but now regard this entire unit as a CLUSTER.
2. Build the system as multiple clusters and connect the cluster using an INTERCLUSTER BUS...
3. If number of clusters are connected via Supercluster bus which create a SUPERCLUSTER.
4. If there are N number of clusters, then the shared memory is divided among the clusters equally
5. This machine are known as DASH - Directory Architecture for shared Mem



Directories:

1. Each cluster has a DIRECTORY that keeps track of which clusters currently have copies of its blocks.
2. If each cluster owns 1M Memory blocks, then the directory has 1M entries.
3. Each entry holds a bit map with one bit per cluster telling whether or not that cluster has the block currently cached.
4. The entry also has a 2-bit state field telling state of the block.
   a) UNCACHED - The only copy of the block is in this memory
   b) CLEAN - Memory is up-to-date; the block may be in several caches.
   c) DIRTY - Memory is incorrect; only one cache holds the block.
5. This list also tell which clusters holds the corresponding cache block.
6. The inter-cluster link systems are WORMHOLE routing which means that the first part of a packet can be forwarded even before the entire packet has been received, thus reducing the delay at each hop.





Dash Protocol - Read

Protocol

1. The protocol based on ownership and invalidation.
2. At every instant, each cache block has a unique owner.
3. For uncached or clean blocks, the block's home cluster is the owner.
4. For Dirty blocks, the cluster holding the one and only copy is the owner.
5. Writing on a clean block requires first finding and invalidating all existing copies.

VRML - Virtual Reality Modelling Language
AIFF - Audio Interchange File Format.
MIDI - Musical Instrument Digital Interface.
CGI - Common Gateway Interface.
OSI - open systems Interconnection.
BITMAP - Bitmap.
TIFF - Tagged Image File Format.
JPBG - Joint Photographic experts group.
GIF - Graphic Interchange Format.
PNG - Portable Network Graphics.
ASP - Active server Pages
JSP - Java Sewer Pages.

FTP - File Transfer Protocol is a standard network protocol used for the transfer of computer files from server to a client using the client-server model on a computer network.

HTTP - Hyper Text Transfer Protocol. ~~It sits process~~. Web servers processes req via HTTP It is the network protocol used to distribute info on WWW.

SMTP - Simple Mail Transfer Protocol - is an Internet standard for electronic mail. transmission . It is used to send or receive mail. messages, user-level client mail appln typically use SMTP only to sending messages to mail server for relaying.

SNMP - Simple Network Management Protocol. is an Internet standard protocol for collecting and organizing information about managed devices on IP networks and for modifying ~~devices on IP networks and~~ that info to change device behaviour. Devices - cable modem router, switches etc.

It is widely used in network management for network monitoring.

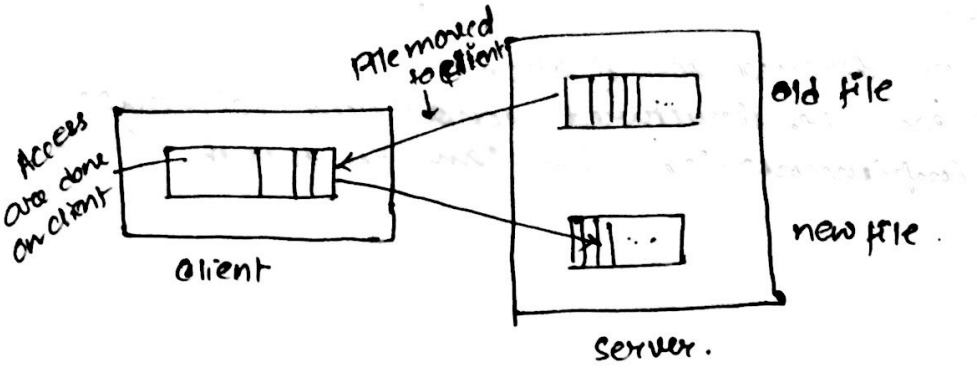
Pop - Post Office Protocol is an appln layer Internet standard protocol used by local email clients to retrieve email from a remote server over a TCP/IP connection. POP has been developed through several versions with POP3 version.

SM 24.10.2017

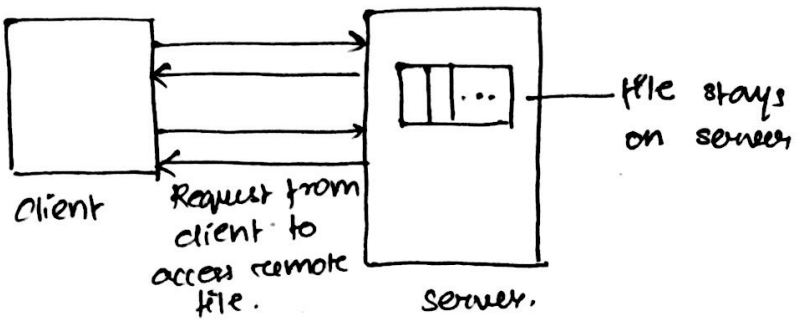
Implementing runnable is advantageous because there is a probability of inheriting multiple threads.
Exception is the topmost hierarchy of exception class.
notify is used with wait() method.

SS

30.10.2017



Upload Download Model



Remote Access Model

Semantics of File Sharing :

When two or more users share the same file it is necessary to define the semantics of reading and writing precisely to avoid problems. When a read operation follows a write operation the read must return the value just written by the system. Similarly when two writes happen in quick succession followed by a read the value read is the value stored by the last write. In effect the system enforces an absolute time ordering on all operations and always returns the most recent values. We will refer to this model as Unix semantics.

In a distributed environment unix semantics can be achieved easily as long as there is one file server and clients do not cache files. All reads and writes go directly to the file server. But in general the files are cached to maintain local copies of heavily used files. One way out of this difficulty is to propagate all changes to cached files back to server immediately

requiring a read to see the effects of all previous writes one can have a new rule says that 'changes to an open file are initially visible only to the processes that modify the file. Only when the file is closed are the changes made visible to other processes'.

Distributed file systems often provide file replication as a service to their clients. In other words multiple copies of selected file is maintained with each copy on a seperate file server. The reason for offering such a service may varey but among the major reasons are ① to increase reliability ② to allow file access even when one file server is down ③ To split the work load over multiple servers. There are 3 different way file can be replicated into some other server:

1. Explicit file replication - When a process makes a file it does so on one specific server. Then it can make additional copies on other server if desired. If the directory server permits multiple copies of a file the network addresses of all copies can then be associated with the file name so that when the name is looked up all copies will be found.

2. Lazy Replication - Only one copy of each file is created on some server. Later the server itself makes replicas on other server automatically without the programmers knowledge. The system must be smart enough to be able retrieve any of copies if required.

3. Group communication - In is scheme all write system calls are simultaneously transmitted to all the servers so extra copies made at the same time the original is made. There are two principle difference between Lazy replication and Group communication

→ with lazy replication one server is addressed rather than a group.

→ Lazy replication happens in the background whereas all copies are made at same time of group communication

SM

31.10.2017

To retrieving the priority getPriority (thread)
" give " " setPriority (p).

```
class PriorityDemo
{ psvm(String args[])
  { ChildThread obj1 = new ChildThread (Thread.NORM_PRIORITY-2);
    ChildThread obj2 = new    "        (   "    "  PRIORITY+2);
                 obj3 =  "    "        (   "    "  "  +3);
    obj1.start();
```

```
obj2.t.start();
obj3.t.start();
try
{
    soln("Main Thread waiting");
        obj1.t.join();
        obj2.t.join();
        obj3.t.join();
}
catch(InterruptedException e)
        { soln("Main thread is interrupted");
}
Soln(obj1.t + " is alive ?: " + obj1.t.isAlive());
"    (obj2.t + " " " ?: " + obj2.t. " ());
"    (obj3.t + " is alive ?: " + obj3.t ".());
soln("Main thread is ~~waiting~~ exiting");
}
}
```

Daemon Thread:

```
public class JavaDaemonThread {
    psvm(String[] args) throws InterruptedException {
Thread dt = new Thread(new DaemonThread(), "dt");
dt.setDaemon(true); → converting particular thread to daemon thread.
dt.start();
// continue program
Thread.sleep(30000);
Sopn("Finishing program");
}
}
class DaemonThread implements Runnable {
    public void run() {
        while(true){
            processSomething();
        }
    }
    private void processSomething() {
        try {
            sopl("Processing daemon thread");
            Thread.sleep(5000);
        } catch(InterruptedException e) {
            e.printStackTrace();
        }
    }
}
```