30-08-2019

3 Processes

* Threads:

Threads are the building blocks of processes. Threads are also known as small part of process because of multithreading technique the efficiency of processing becomes smooth and the better performance can be attended or acheved.

Multiple threads helps to gain fine granularity (simplicity).

A process is often defined as program in execution, i.e the program which is currently being executed on operating system.

The fact that multiple process may be concurrently sharing the same CPU and other hardware resources.

It becomes hard to acheave better performance. Like a process, a thread executes it's own piece of code, independently from other threads, thats why high degree of performance can be acheved.

Page No.
Date

Threads are often provided in a form of thread package. Such a package contains operation to create and destroy thread. There are basically two approches to impliment thread package.

First approach is to construct a thread library that is executed entirely in user ~~guide~~ mode.

and second approch is to have the kernel be aware of threads and and schedule them.

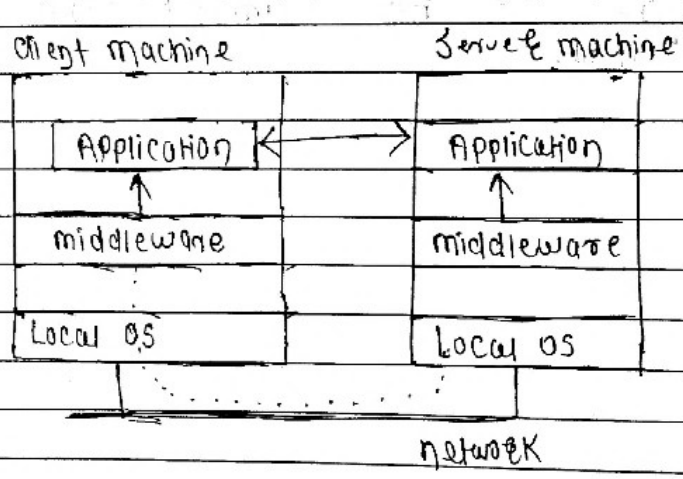
The better solution for thread library execution at user node is that implimenting threads in operating system's kernel. because here every thread operation will have to be carried out by the kernel.

Page No.
Date

## * Clients *

A major task of client machine is to provide the means for users to interact with remote servers.

There are generally two ways in which this interaction can be supported. The first is where interaction takesplace with the use of application level protocol and the second where the communication takesplace without application level protocol. These two approches are properly shown in below diagram




ⓐ A networked application with its own Protocol.

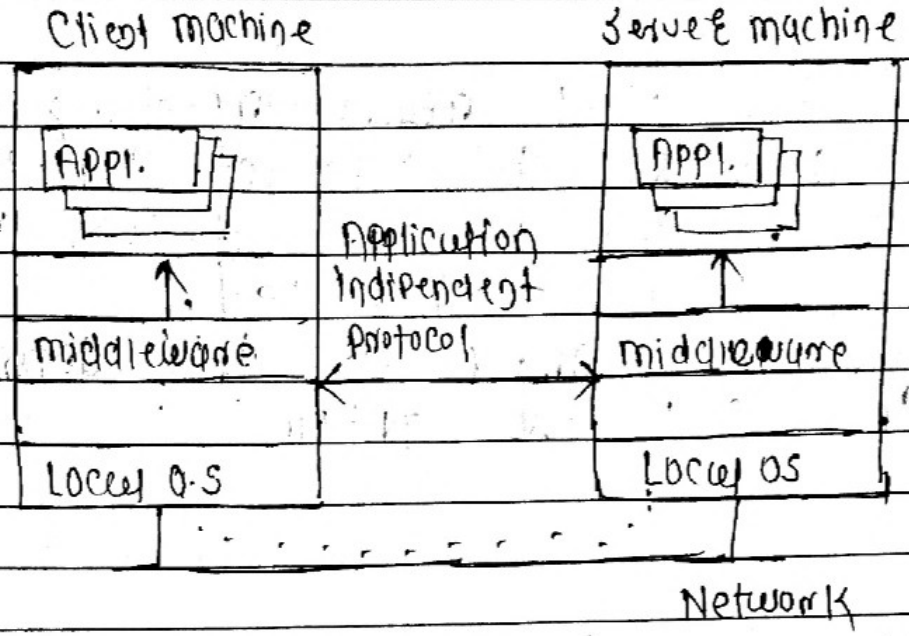



fig. A general solution to allow access to remote application

As we can see in above diagram (सोचिये) the second approach is more convinient. where solution is to provide direct access to remote services by only offering a convienient user interface. because of this approach more applications can be supported Hence more services can be accessed.

this means that the client machine is used only as a terminal with no need for local storage.

In the case of networked user interfaces everything is processed and stored at the server.

This approach is popularly known as "thin client" approach.

This thin client ~~reach~~ receiving most attention as internet connectivity increses. and handheld devices are becoming more sophisticated.

The example of this approach is the X window system.

## * Servers *

A server is a process implimenting a specific service on behalf of a collection of clients. In essence, each server is organized in the same way: it waits for an incoming request from a client and subsequently ensures that the request is taken care of, after which it waits for the next incoming request.

There are several ways to organized servers. In the case of iterative server, the server itself handles the request and if nessary a responce to the requesting client is provided.

Iterative Ser - Automatically manage
User request & give response
Concurrent Ser - It opposite of Iterative server
It only pass the request from one
other to another

Page No.
Date

A Concurrent server does not handle the request itself. infact it's job is to pass the request to the separate thread or another process after which it's immidiety waits for next incoming request.

An example of concurrent server is multithreaded server.

## * ~~Server cluster~~ :

A server cluster is nothing else but a collection of machines connected through a network, where each machine runs one or more servers. The server-cluster that we consider here, are the one in which the machines are connected through a local area network. Often offering high bandwidth and low latency.

below diagram shows the functionality of server cluster in simplified manner.

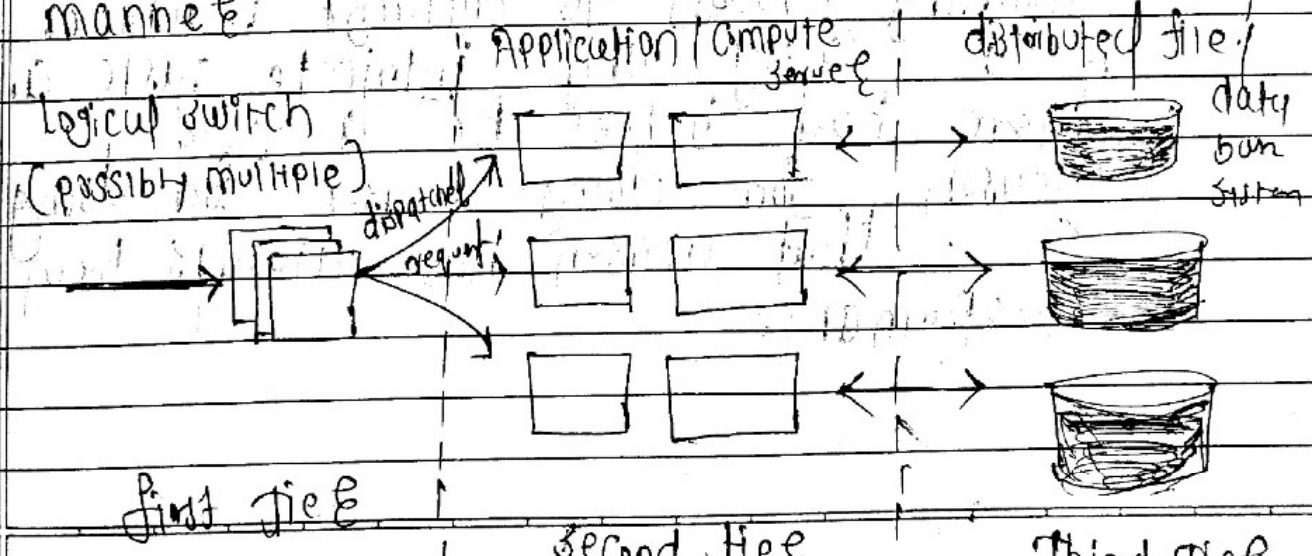



Fig: The general organization of three tiered server cluster

As we can in above fig there is a three tier logical organization of server cluster.

In the first tier there is logical switch which decides the client request root. in the second tier there are compute servers which processes the request and passes to the respective database system in the third tier.

## Code migration

Traditionally code migration is distributed system took place in the form of process migration in which an entire process was moved from one machine to another moving a running process to different machines is a costly and intricate task and there had better be a good reason for doing reason for doing so. that reason has always been performance

The basic idey is that overall system performance can be improved is process are moved from heavily loaded to lightely loaded machines.

Consider as an example a client server system in which the server manage a huge database. If a client application lead to perform many database operations involving large quantities of data it may be better to shift part of the client application to the server and send only the result across the network.

Model for Code Migration

① Weak mobility :-
The bare minimum for code migration is to provide only weak mobility in this model it is considered to transfer only the code segment along with perhaps some initialization data

② Strong mobility :-
In contrast to weak mobility in system that supports

Strong mobility the execution segment can be transferred as well.
Clearly strong mobility is much more general than weak mobility but also much harder to implement.

Irrespective of whether mobility is weak or strong a further distinction can be made between sender initiated and receiver initiated.

a) Sender initiated migration :-
In sender initiated migration, migration initiated at the machine where the code currently resides are is being executed.

b) Receiver initiated migration :-
In this migration the initiative for code migration is taken by the target machine.

New chapter

Page No.
Date

4. Communication :

* Fundamental *

To make it easier to deal with the numerous level and issues involve in communication. The international standard organization (ISO) developed a reference model that clearly identify the various involved gives them standard names and points out which level should do which job.

This model is called the open system interconnection reference model usually abbreviated as ISO-OSI some times just the OSI model.

The rules of communication are called protocol there are two types of protocol are used.

First is connectionless protocol for ex. HTTP where connection is not established in advance connection remain for a viewer time. The second is connection oriented protocol.

for ex. FTP / TCP / IP, or we can considered any four communication is available for code time a consideration with process.

Page No.
Date

# * Types of Communication *

## ① Persistent Communication :

for deliver a message

Where Communication remain as long as the deliverer is not provided to the receiver.

## ② Tranisient Communication :-

sending and receiving application executed

It is an contrast to persistent where Communication took place when sending and receiving application are executive.

## ③ Asynchronous Communication :-

Or besides before transient Communication Or Synchronous

like whatsapp mobile

Page No.
Date

B blocked @ untill it request B accepted

There are essentially three point where synchronization can takeplace.

first the sender may be blocked untill the middleware notice that it will take over transmission of the request.

second the sender may synchronize untill it request has been delivered to the intended receipient

Third synchronization may takeplace by letting the sender wait untill it's request has been fully processed, that is up the time that the receipient returns a response.

Page No.
Date

## * Remote procedure call *

Remote procedure calls are used for communication purpose. We can say that RPC's are calls or messages which send and receives by machines from below example we can understand the functionality of RPC.

for ex :- When a process on machine A calls a procedure on machine B, the calling process on A is suspended, and execution of the called procedure takes place on B. Information can be transported from the caller to callee in the parameter and come back in the procedure result.

No message passing at all is visible to the programmer. This method is known as Remote procedure calls or RPC.

## * Basic RPC operation *

Initially we have to understand about conventional procedure calls then we can understand how the call itself can be split into client and the server part that are each executed on different

machines.

To understand How RPC works it is important first to fully understand how a Conventional (i.e single machine) procedure Call works. Consider a Call in C like.

Count = read (fd, buf, nbytes);

Where fd is an Integer Indicating a file, bef is an array of character into which data are read, ~~and nbytes~~ And nbytes is another Integer telling how many bytes to read.

After the read procedure has finished running, it puts the return value in a register, removes the return addresses and transform Control back to the Caller.

One Other parameter passing mechanism also exist. Although it is not used in C. It is Called Call by Copy / restore. It Consist of having the variable Copied to the stack by the Caller, as in Call by value, and then Copied back after the Calls, overwriting the Caller Original value.

From the below fig we can be said that the functionality of RPC in between a client and server program.

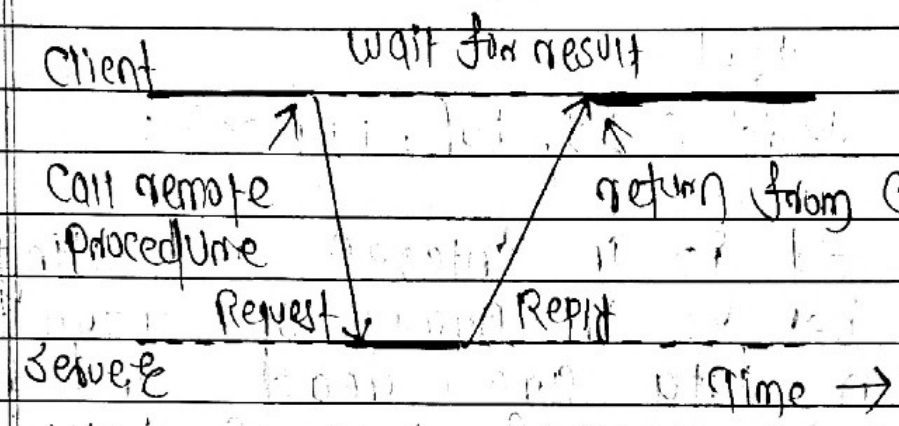



fig. Principle of RPC between a client and server program.

## * Parameter Passing *

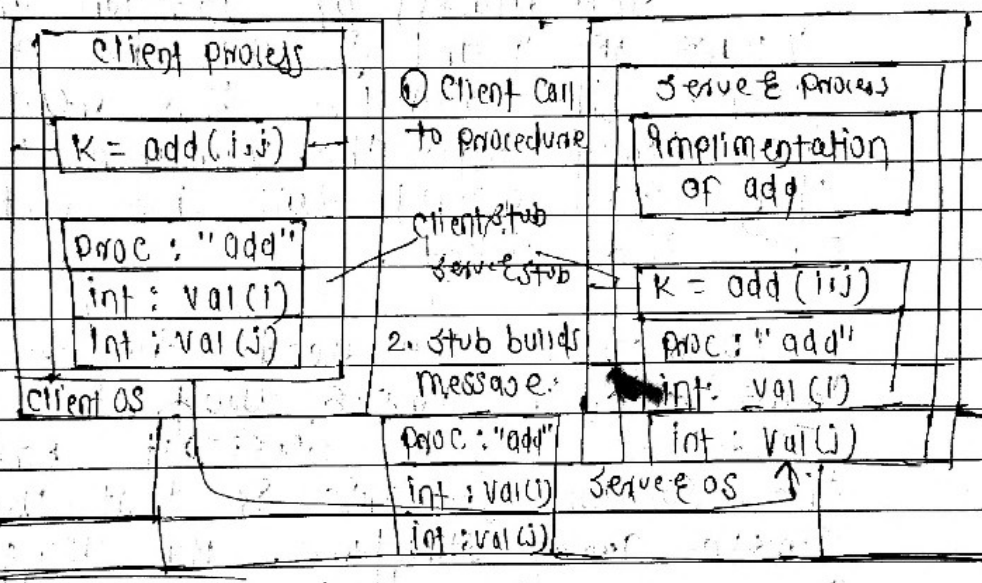
The function of the client stub is to take its parameters, packs them into a message and them to the server stub.

"Packing parameter into a message is called parameter marshaling."

In the below fig we have a very simple example.

Page No
Date

Consider a remote procedure calls add(i,j) that takes two integer parameter i and j and return their arithmetic sum as a result. ~~(Fig a~~




③ message is sent across the network

④ server O.S hands message to server stub

⑤ stub unpack message

⑥ stub makes local calls to "add"

fig The steps involved in doing

As per above fig when message arrive at server the parameters inside the RPC gets opened then the implimentation of process takes place.

for ex :-

The process of addition of values of i and j will be done after the addition the result will be stored into the variable k and then this result will be returned to the client.

Parameters are needs to be transfered in proper way. This is considered as a difficult problem, the solution for this is that use of simple array structure. By using array also one difficulty remains i.e to handle pointers, because of this data structure may become complex.

Page No.
Date

## Asynchronous RPC :-

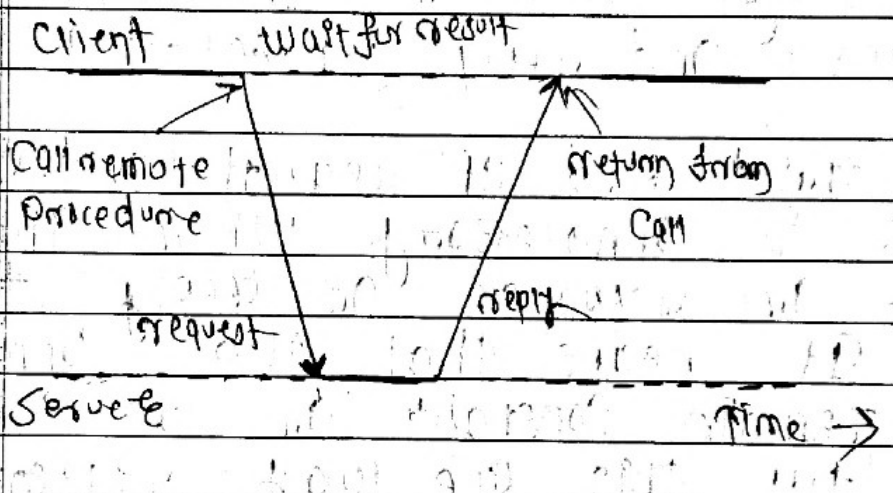
As we have seen in Conventional RPC there is a request and reply structure is used for this client server communication in which a client sends a request to the server at that time after this the connection ~~we~~ remains blocked till the processing over request done by the server and reply generated.

This strict kind of request and reply behaviour is unnecessary where there is result or responce or reply to return. It means that the connection should remain constant or we can say unblocked till this one way request doesn't completes successfully.

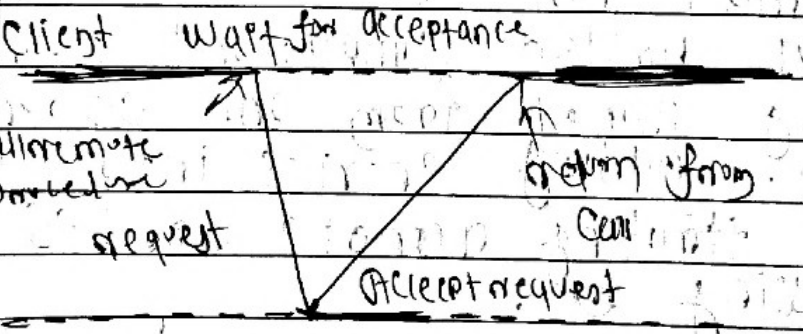
There are so many cases where connection must not be blocked untill the successful process completion such as transfering money from one account to another, adding entries into a database, strating remote services, batch processing etc.

To support such situation, RPC system may provide facilities for what are called "Asynchronous" RPC.

where client remains in connection until the complete execution of process. It means that client will continue without further blocking as soon as it has received server's acknowledgement.




(a). Call local procedure and return results




(b)

Fig. The interaction between client & server in traditional RPC

# * Message Orientation *

Message-Oriented transient Communication :-

{ Berkeley Sockets } :-

- Conceptually Sockets ~~to~~ is a Communication end point to which an application can write data that are to be send out over the underlying network, and from which incoming data can be read.

→ Next page

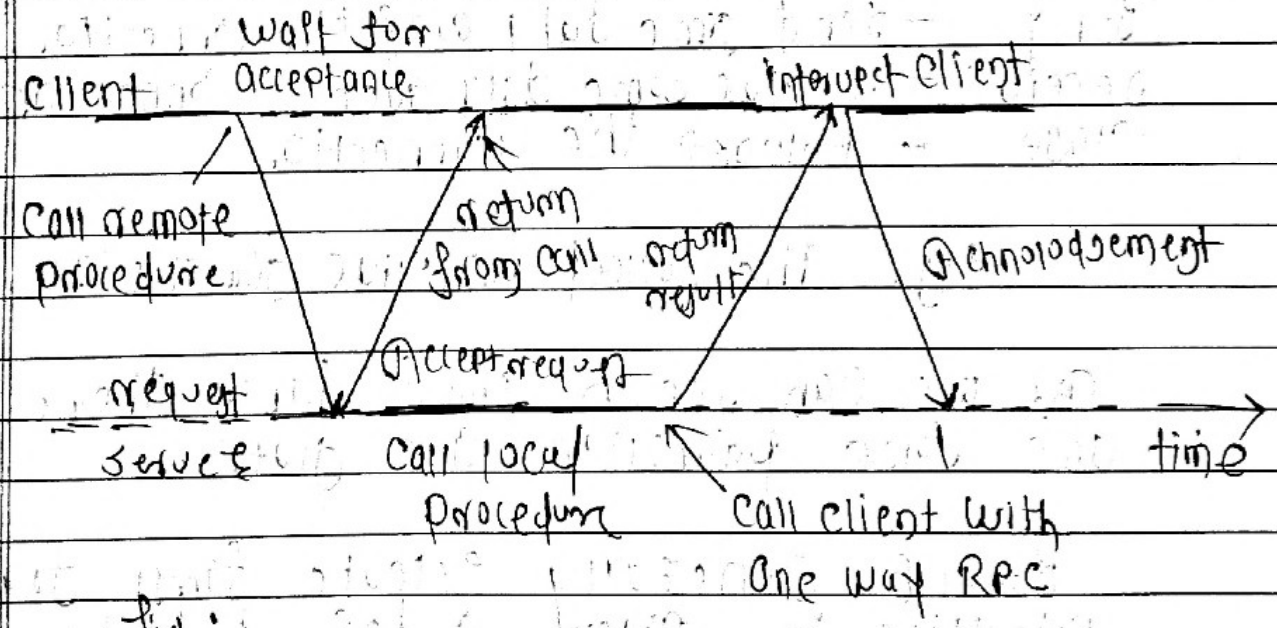



Fig:
A client & server interacting through two asynchronous RPCs.

Sockets forms hidden layer over the actual communication end point which is used by local operating system for specific transport protocol.

In below table we can see the messages handle by the Berkeley sockets-

| Primitive | — meaning |
|---|---|
| Socket | — Create new comm end point |
| Bind | — Attach a local address to a socket |
| Listen | — Announce willingness to accept connection |
| Accept | — Block caller until a connection request arrives |
| Connect | — actually attempt to establish a connection |
| Send | — send some data over the connection |
| receive | — receive some data over a connection |
| Close | — Release the connection |

Fig. The socket primitive for TCP/IP

As we can see in above table there are some primitive are given.

Server generally excute first four primitive in given order and in the above table the job of each primitive is given properly.

The need to be hardware and platform independent message passing i.e the standard defination of message passing interface or MPI.
MPI is designed for parallel application.

* Message-Oriented persistent (strong) Communication *

(message queuing model)

- message oriented Communication is also known as message oriented middleware (MOM)
The essence of these system is that they offer intermidiate-term storage capacity for message, without requiring either the sender or receiver to be active during message transmission.

The basic idea behind a message queuing system is that application communicate by inserting message in specific queues.
These messages are forwarded over a series of communication server and are eventually delivered to the destination even if it was down when the message was sent. In practice most communication server are directly connected to each other.

In other words, A message is generally transferred directly ~~connected~~ to a destination server.

In principle, each communication has its own private queue to which other application can send message.

An important aspect of message-queing system is that a sender is generally given only the guarantees that its message will eventually be inserted in the recipient's queue.

No guarantee are given about when or even if the message will actually be read, which is completely determined by a behaviour of recepient.

As like every communication system here in message queuing system there are some also some primitive are used these messages are nothing but communication sending & receiving calls as shown in table below.

| Primitive | meaning |
|---|---|
| put | - Append a message to a specific queue |
| Get | - Block until the specified queue is nonempty, and |
| ~~put~~ | - remove the first message. |
| ~~Notify~~ | - |

Page No
Date

Poll - Check a specified queue for messages and remove the first never block.

Notify - Install a handler to be called when a message is put into the specified queue.

Stream
* ~~Stating~~ Oriented Communication *

Support for the exchange of time-independent information is often formulated as support for continuous media.

Every media has some encoding technique.

for ex. text is generally encoded as ASCII or Unicode. Image can be represented in different formats such as GIF or JPEG. Audio stream can be encoded in a computer system by, for example taking 16 bit samples using PCM (pulse code modulation).

A data stream is nothing but a sequence of data unit. data stream can be apply to discrit as well as continuous media.

Page No.
Date

In synchronous transmission mode there is a maximum end to end delay defined for each unit in a data stream.

In Isochronous transmission mode, it is necessary that data unit are transfered on time.

Stream can be simple or complex. A simple stream consist of only a single sequence of data whereas a complex stream consist of several related simple stream called substream.

Timing (and other nonfunctional) requirements are generally expressed as quality of service (QOS) requirement. QOS for continuous data stream mainly concern timeliness, volume and reliability.

Here in data stream transport some important properties are bit rate, end to end delay etc

Page No.
Date

# * Multicast Communication *

An important topic in communication in distributed system is the support for sending data to multiple receivers, also known a multicast communication.

- There are several types of multicast communication are used such as ~~Ap~~ Application-level multicasting.

## * Application level multicasting *

The basic idea in application-level multicasting is that node organized into overlay network, which is then used to disseminate information to a members.

lot of complication can be solved by using network level routing.

A crucial issue is the construction of the overlay network. In essence there are two approaches. First node may organize themselves directly into a tree, meaning that there is a unique (overlay) path between every pair of nodes. ~~An alternative approach is that no~~

The main diffrence betn the two approaches is that the second

approach is more robust as compar to first approach because in this second approch because of mesh network communication or transformation still become possible even though any node fails or communication brakes.

To make this approach more concreat/ strong DHT-based pair to pair to system is used.

To simplify matters some what assume that a multicast group has an associated and well known node that keeps track of that node that have joined the tree. When a new node sends request to join this network, it has to contact the main node where all the communication links are jethered.

The goal is to select the best member that can operate as the new node parent in the tree.

END CHAPTOR

16-Sept-2019

Page No
Date

# 5. Naming

## * Names, identifires, And addresses *

A name in a distributed system is a string of bits or characters that is used to an entity.

An entity in a distributed system can be practically anything. Typical examples includes resources such as hosts, printers, disks, and files. Other well-known examples of entities that are often explicitly named are processors, users, mailboxes, newsgroups, web pages, graphical windows, messages, network connection, and so on.

To operate on an entity, it is necessary to access it, for which we need an access point. An access point is yet another, but special kind of entity in distributed system. The name of access point is called as address. The address of an access point of an entity is also simply called an address of that entity.

In addition to addresses, there are other types of names that deserve special treatment, such as names that are used to uniquely identify an entity.

A true identifires is a name that has the following properties.

(i) An identifier refers to at most on entity

(ii) Each entity is refers to by at most identifier.

(iii) An identifier always refers to the same entity. (i.e it is never reused).

## * Flat Naming *

In flat naming there are various issues are resolved regarding naming, addressing and identification. for this perpose there are two simple solutions are provided to locate an entity.

These two solutions are broadcasting and multicasting. Both are applicable only on local area network.

In LAN these broadcasting and multicasting techniques works effectively efficiently and easily.

Page No
Date

Cosider a distributed system built on a computer network that offers efficient broadcasting facilities. Such facilities are offered by local area network in which all machines are connected to single cable or the logical equivalent thereof. Also local area wireless network fall into this category.

locating an entity ~~is used in the~~ in such an enviroment is simple.

Broadcasting become inefficient when the network grows.

One possible solution is to switch to multicasting by which only a restricted group of hosts receives the request. for example, ethernet network support data link level multicasting directly in language.

multicasting can also be used to locate entities in point to point network.

- multicasting forwarding pointers :-

— Another popular approach to locating mobile entities is to make use of forwarding pointers. The principle is simple:

When an entity moves from A to B it leaves behind A as reference to it's new location at B.

The main advantage of this it's simplicity. as soon as an entity has been located for example by using a traditional naming service, a client can look up the current address by following the chain of forwarding pointers.

There are also a number of important drawbacks. First if no special measure are taken, a chain for a highly mobile entity can become so long that locating that entity is prohibitively expensive. Second, all intermediate location in a chain will have to maintain their part of the chain of forwarding pointers as long as needed. A third drawback is the vulnerability to broken links. As soon as any forwarding pointer is lost the entity can no longer be reached. An important issue is therefore to keep chains relatively short, and to ensure that forwarding pointers are robust

Page No.
Date

Human readable

# * Structured naming *

Flat names are good for machine, but are generally not very convenient for humen to use. As an alternative, naming systems generally support structured names that are composed from simple, humen readable names. Not only file naming but also host naming on the internet follow this approach.

Names are commonly organized into what is called a name space. Name spaces for structured name can be represented as a labeled, directed graph with two types of nodes.

A leaf node represents a named entity and has the property that it has no outgoing edges. A leaf node generally stores information on the entity it is representing for example, its address so that a client can access it.

In contrast to a leaf node, a directory node has a number of outgoing edges each labeled with a name as shown in fig below.

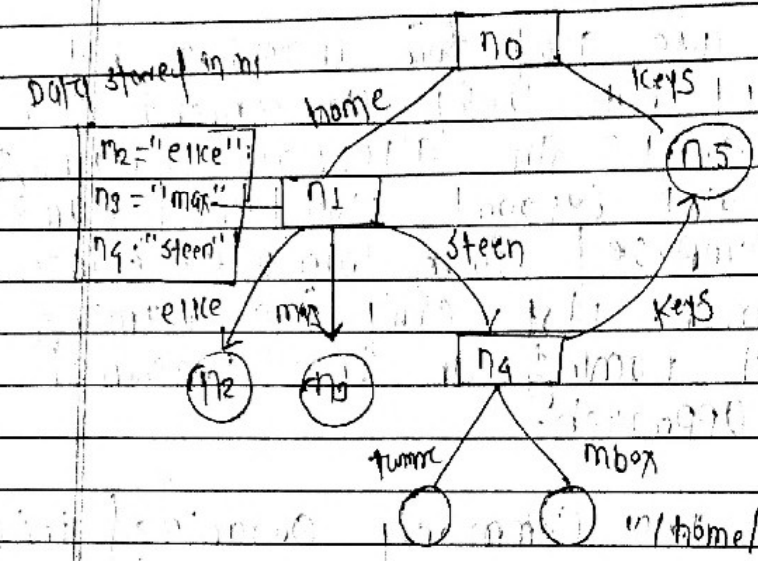



Fig. A general naming graph with a single root node

A directory node stores a table in which an outgoing edge is represented as a pair.

The naming graph shown in fig 5.9 has one node namely n0, which has only outgoing and no incoming edges. Such node is called as root node.

Page No
Date

## * Attribute - Based Naming *

Flat and structured names provide a unique and location independent way of to entity. Moreover, structured names have been partly designed to provide a human friendly way to name entities. so that they can be conventiently accessed.

However location independence and human friendliness are not the only criterion for naming entities.

In particular, as more information is being made available it become important to effectully search for entities. This approach requires that a user can provide meaney a description of what he is looking for.

There are many ways in which description can be provided, but a popular one in distributed system is to describe an entity in terms of (attribute, value) pairs, generally refered to as attribute-based naming. In this approach, an entity ty is assumed to have an associated collection of attributes.

Attribute - based naming system are also known as directory service, where as system that support structured naming are generally called naming system.

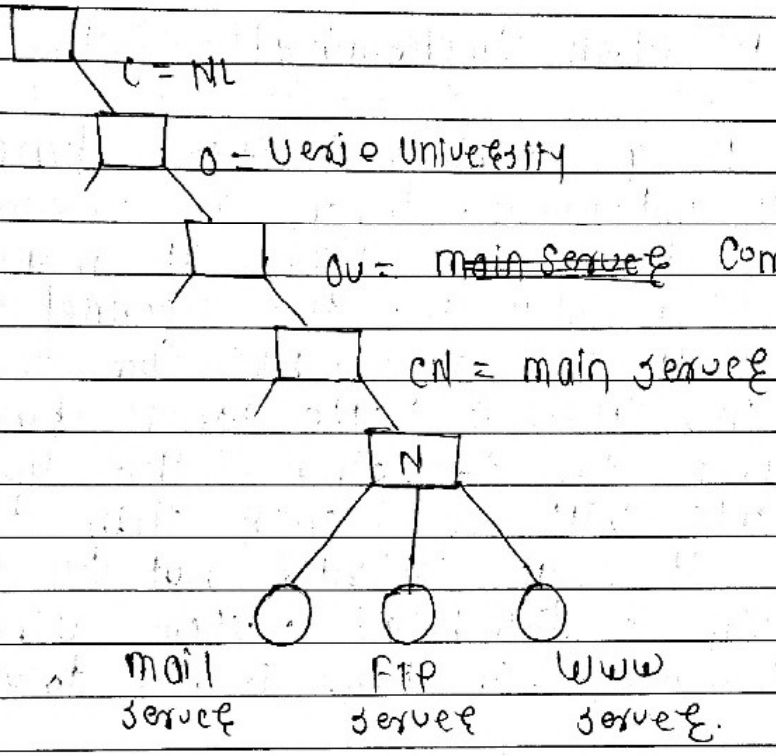
A common approach to tackling distributed services is to combine structured naming with attribute-based naming.

This approach has been widely adoped for example, in Microsoft's active directory service and other systems. Many of these system use, or rely on the lightweight directory access protocol commonly referred simply as LDAP.

As an example, a simply directory entry identifying the network address of some general servers from below fig.

| Attribute | Abbr | Value |
|---|---|---|
| Country | C | NL |
| Locality | L | Amsterdam |
| Organization | O | Vrije University |
| Organization unit | OU | Comp. Sci |
| Common name | CN | Main server |
| mail server | - | 137.37.20.3, 130.37.24.6, 137.37.20.10 |
| FTP server | - | 130.37.20.20 |
| WWW server | - | 130.37.20.20 |

Fig. Simple example of an LDAP directory entry using LDAP naming conventions.




Above is directory information tree i.e DIT which essentially forms the naming graph of an LDAP directory service in which each node represent a directory entry.

New chapter

# Synchronization

## * Clock Synchronization *

In a centralized system time is unambiguous. When a process want to know the time, it makes a system call and the kernel tell it.
If process A asks for the time, and then a little later process B asks for the time, the value of B gets will be higher than the value A.
It will certenly not be lower.
In a distributed system, achequing agreement on time is not trivial.

## * Physical clocks *

Nearly all computer have a circuit for keeping track of time. Despite the widespread use of the word "clock" to refer to these devices, they are not actually clock in the usual sense.
Timer is perhaps a better word.
Oscillation with each crystal are two registers:

(i) Counter
(ii) Holding register

Page No.
Date

Each oscillation of the crystal moves the counter by one, then counter is reloaded in the register from the holding memory.

In this way, it is possibly to program a timer to generate an interept 60 times a second, or at any other desired frequency. Each interupt is called one click tick.

When the system is booted, it usually asks the user to enter the date and time, which is then converted to the number of ticks after some known starting date and stored in a memory. Most computer have a special battery-backup CMOS RAM so that the date and time need not be entered on subsequently boots.

At every clock tick, the interept service procedure add one to the time stored in memory. In this way, the clock is kept up to date.

Page No.
Date

# * Global positioning System *

As a step toward actual clock Synchronization problems. We first consider a related problems, namely determining one's geographical position anywhere on earth. This positioning problems is by itself solved through a highly specific dedicated distributed system, namely GPS, which is an acronym for global positioning system. GPS is a satelite based distributed system that was launched in 1978.

Although it has been mainly for military application in recent years it has found it's way to many civilian applications, notably for traphic navigation.

However many more application domain exist. For example, GPS phones now allow to let caller ~~tracc~~ track other's position, a feature which may show it be extremly handy when you are lost or in trouble.

This principle can easily be applied to tracking other things as well, including pets, children, cars, boats, and so on.

Page No.
Date

Gps uses 29 satelites, each circulating in an orbit at a height of approximatly 20,000 km. Each satelite has up to four automic clocks, which are regularly calibrated from special station on Earth. A satelites continuously broadcast it's positions, and time stamps each message with it's local time.
This broadcasting allows every receiver on earth to accurately computes it's own position using, in principle, only three satellites.

## * Clock Synchronization algorithm *

If one machine has a WWV (world wide vacuum) receiver, the goal becomes keeping all the other machines synchronized to it. If no machine has WWV receiver, each machine keep truck of it's own time, and the goal is to keep all the machines, together as well as possible.

All the algorithm have the same underlaying model of the system.
Each machine is assumed to have a timer that causes an interept H times a second. When this timmer goes off, the interept handler add 1 to

Page No.
Date

Software clock that keeps track of the number of ~~tricks~~ ticks since some agreed upon time in the past. let us call the value of this clock C. more specifically, when the UTC time is t the value of the clock on machine p is Cpu. In perfect world, we would have Cpu = t for all p and all t.

Network time protocol plays an important role in time management over the network. Here in a client server model, client asks time to the server periodically, to make system up to date so server remains active here.

In wireless network reference broadcast synchronization (RBS) is a clock synchronization protocol is used to maintain the times for the machines which are wirelessly connected.

Page No.
Date

# * Logical Clocks *

## * Lamport's Logical Clocks

- To synchronize logical clocks, Lamport defined a relation called happens-before. The expression "a ~ b" read "a happens before b" and means that all processes agree that first event a occurs, then afterward, event b occurs. The happens-before relation can be observed directly in two situation.

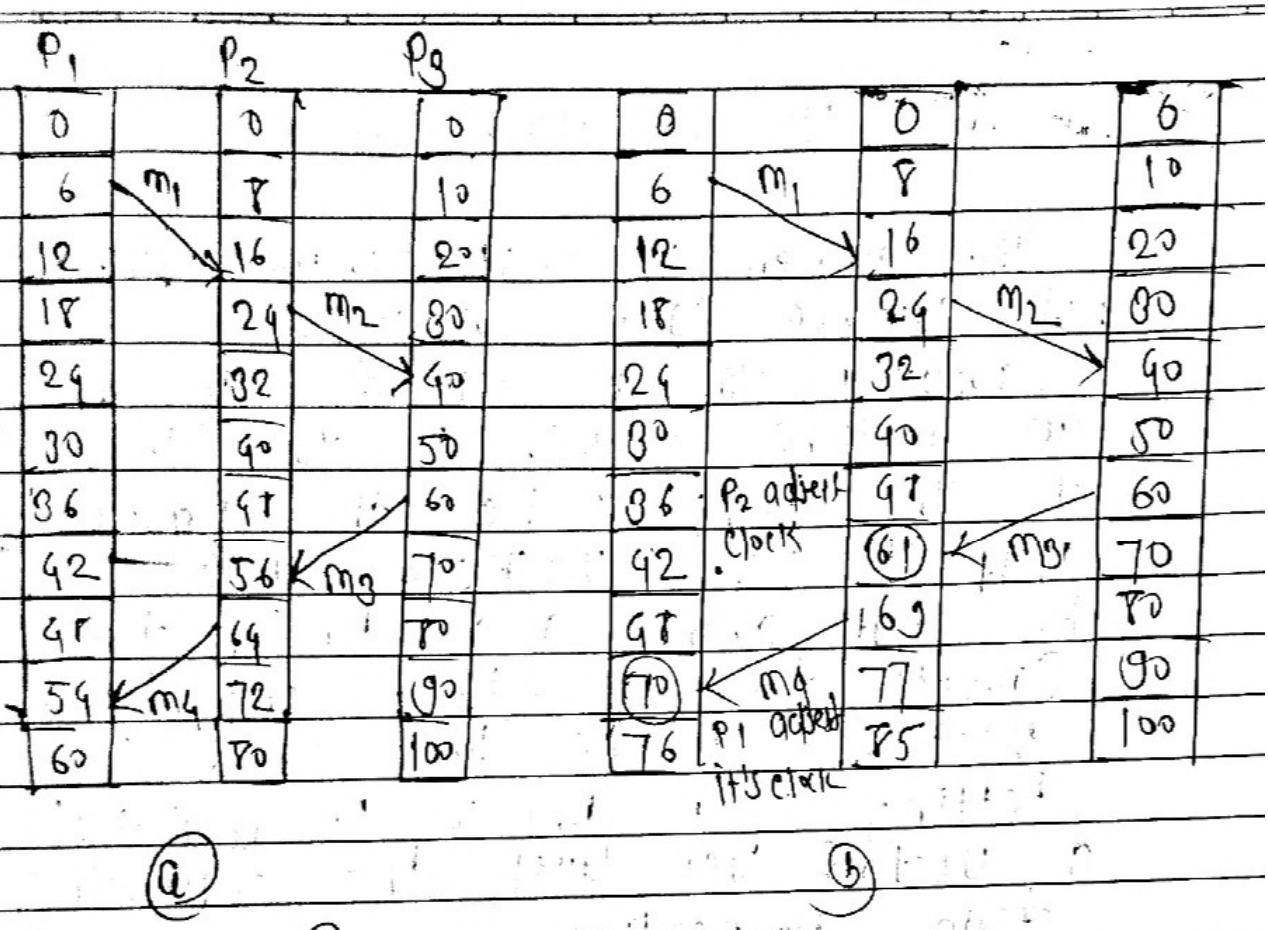
(i) If a & b are events in the same process, and a occurs before b, then a ~ b is true.

(ii) If a is the event of a message being send by one process, and b is the event of the message being received by another process, then a ~ b is also true. A message can not be received before it is sent, or even at the same time it is sent, since it takes a finite, nonzero amount of time to arrive.

Page No.
Date

Happens-before is a transitive relation so if $a \sim b$ and $b \sim c$, then $a \sim b$. If two event $x$ and $y$, happens in a different processes that do not exchange messages (not even indirectly via third parties), then $x \sim y$ is not true, but neither is $y \sim x$. These event are said to Concurrent, which ~~sup~~ simply means that nothing can be said about ~~what~~ when the event happend or which event happend first.

Consider the three processes as depicted in fig 9. The processes run on different machine, each ~~three~~, with its own clock, running at its own speed. As can be seen from the figure, when the clock has ticked 6 times in process $P_1$, it has ticked 8 times in process $P_2$ and 10 times $P_3$. Each clock runs at a constant rate but the rates are different in the crystal.




(a)

fig. Three processes, each with it's own clock. The clocks run at different at different rates.

(b). ~~The clocks~~ Lamport's algorithm correct the clock.

Page No.
Date

# * Vector clocks *

The problem is that lamport clock do not capture ~~cau~~ causality.
Causality can be captured by means of vector clocks. A clock vc assigned to an event a has the property that if vc(a) < vc(b) for some event ~~be~~, b- that event a is known to causely to precede event b.
vector clocks are constructed by letting each process $p_i$ maintain a vector vc with the following two properties.

① $vc_i$ is the number of event that have occoured so far at $p_i$. In other words $vc_i$ is the local logical clock at process $p_i$.

② If $vc_j(v) = k$ then $p_i$ known that k events have occoure at $p_j$. It is thus $p_i$'s knowledge of the local time at $p_j$.

Using vector clocks it is now possible to ensure that a message is delivered only if all messages that causally precede it have also been received as well.

To enable such a scheme, we will assume that messages are multicast within a group of processes.

Note that this causally-ordered multicasting is weaker than the totally-ordered multicasting we discussed earlier.

Specially if two messages are not in any way related to each other, we donot care in which order they are delivered to application, they may even be delivered in different order at different location.

Furthermore, we assume that clocks are only adjusted when sending and receiving messages. In particular upon sending a message, process $P_i$ will only increment $VC_i[i]$ by 1. When it receives a message m with timestamp ts(m). It only adjust $VC_i[k]$ to $\max\{VC[k], ts$ for each k.

Page No.
Date

## * Mutual Exclusion *

Fundametel to distributed system is the Concurrency and Collaboration ~~to simultaneously~~. among multiple processes. In many cases, this also means that processes will need to simultenously access the same resources.

To prevent that such Concurent access corrupt the resources, or make it inconsistent, solution are needed to grant mutual exclusion access by processes.

In this section, we take a look at some of the more important distributed algorithm that have been proposed.

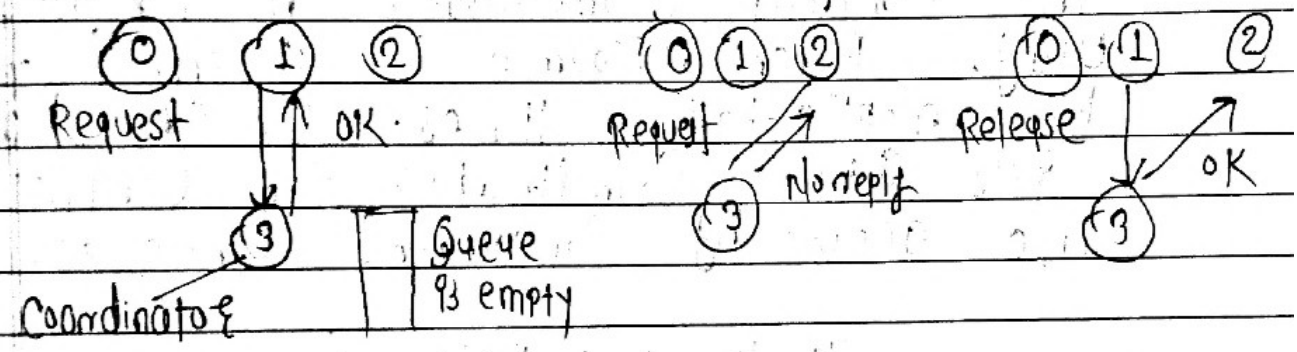
## * Centralized Algorithm *

The most straightforward way to achieve mutual exclusion in a distributed system is the simulate how it is done in a one-procedure system.

One process is elected as the Coordinator. whenever a process want to access a shared ~~location~~ resource it send a request message

to the Coordinator stating which resource it want to access and asking for permission. If no other process is currently accessing that resource, the Coordinator sends back a reply granting permission as shown in fig.

When a reply arrives, the requesting process can go ahead.




(a) (b) (c)

(a) Process 1 asks the Cordinator for permission to access a shared resource permission is granted

(b) Process 2 then ask permission to access the same resource the Coordinator does not reply

(c) When Process 1 release the resource. It tells the Coordinator, which then replies to 2

Page No
Date

# * A Distributed Algorithm *

The algorithm works as follows, When a Process want to access a shared resource it builds a message Containing the name of the resources, it's Process number, and the current logical time. It then sends the message to all other Process, Conceptually, Including Including itself. The sending of message is assumed to be reliable, that is no message is lost.

When a Process receives a request message from another Process, the action it takes depends on it's own state with respect to the resources named in the message. Three different cases have to be clearly distinguish.

① If the receiver is not accessing the resources and does not want to access it, it sends back an OK message to the sender.

② If the receiver already has access to the resources, it simply does not reply. Insted it queues the request.

Page No.
Date

(a) If the receiver want to access the resource as well but has not yet done so, it compares the timestamp of the incoming message with the one contained in the message that it has sent everyone. The lowest one wins. If the incoming message has a lower timestamp the receiver sends back an OK message. If it's own message has a lower timestamp the receiver queues the incoming request and sends nothing.

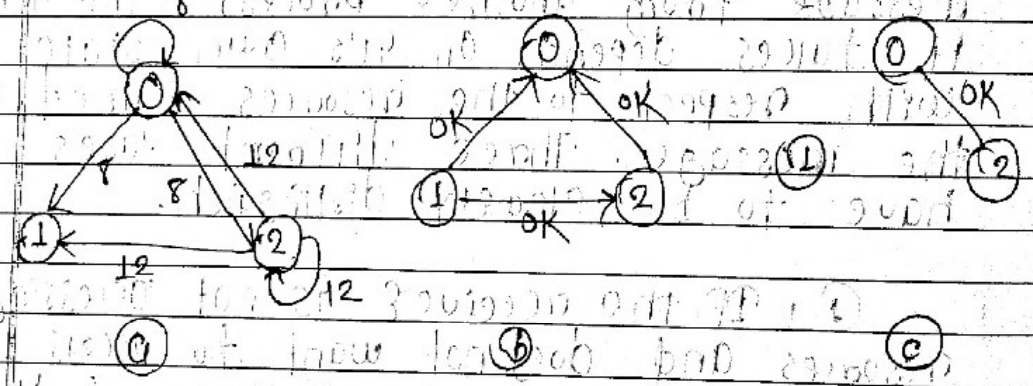



fig. (a) Two processes want to access a shared resource at the same moment

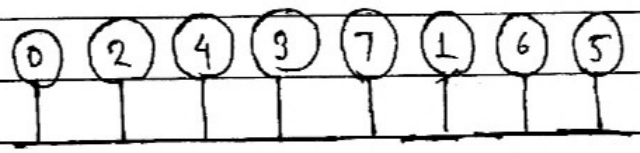
(b) Process 0 has the lowest timestamp. So it wins

(c) When a 0 is done. It sends an OK also, so 2 can go ahead.

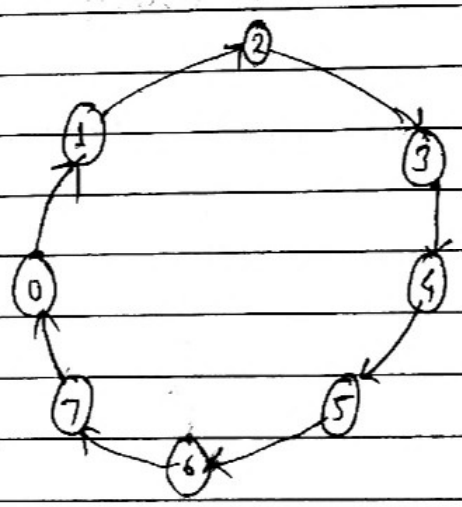
Page No.
Date

## * A token ring algorithm *

In this algorithm with no inherent ordering of the processes are used. In software, a logical ring is constructed in which each process is assigned a position in a ring as shown in fig. The ring positions may be allocated in numerical order of the network addresses or some other means. It does not matter what the ordering is. All that matters is that each process know who is next in line after itself.




(a)
An unordered group of processes on a network




(b)
A logical ring constructed in a software

When the ring is initialized, process 0 is given a token. The token circulates around the ring. It is passed from processes K to process K+1 in point to point message.

When a process acquires the token from it's neighbor it checks to see if it need to access the shared resource.

If so, the process goes ahead it ~~passes~~ does all the work it need to, ~~access~~ and release the resources.

After it has finished it passes the token along the ring. It is not permited to immidiety enter the resource again using the same token.

Page No.
Date

# * Election Algorithm *

## * Traditional election Algorithm *
(The bully Algorithm)

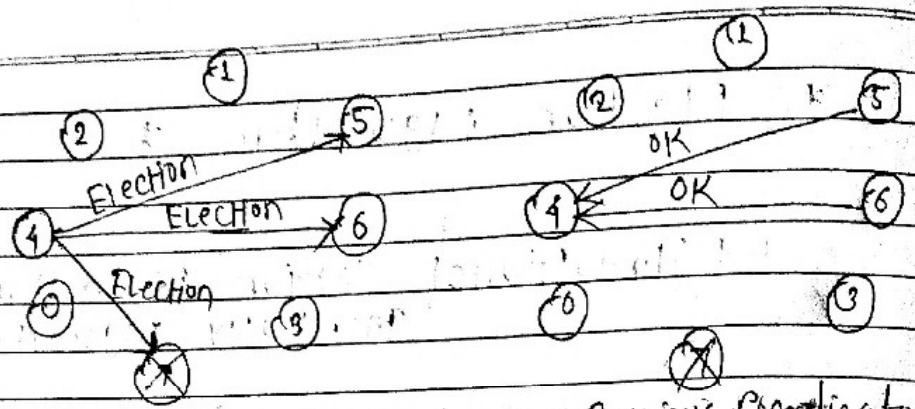
- As a first example, Consider the bully algorithm devised by Garcia-Molina. When any process notice that the Coordinator is no longer responding to requests, it initiates an election.
A Process, P, holds an election as follows.

1. P sends an ELECTION message to all processes with higher number.

2. If no one responds, P wins the election and become Coordinator

3. If one of the higher-ups answers, it takes over, P's job is done

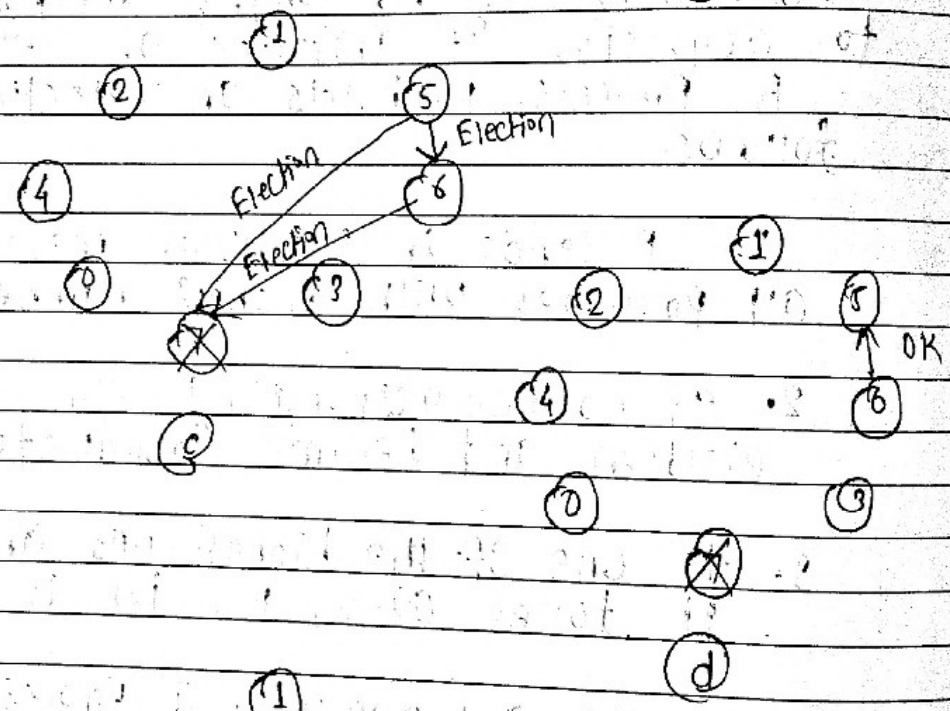
At any moment, a process can get an ELECTION message from from one of its OK message-numbered Colleagues. When such a message arrives, the receiver send an OK message back to the sender to indicate that he is alive and will take over.




Previous Coordinator has crashed.

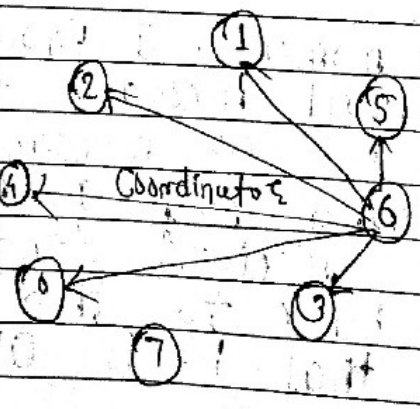
a

b




c

d




e

Fig.

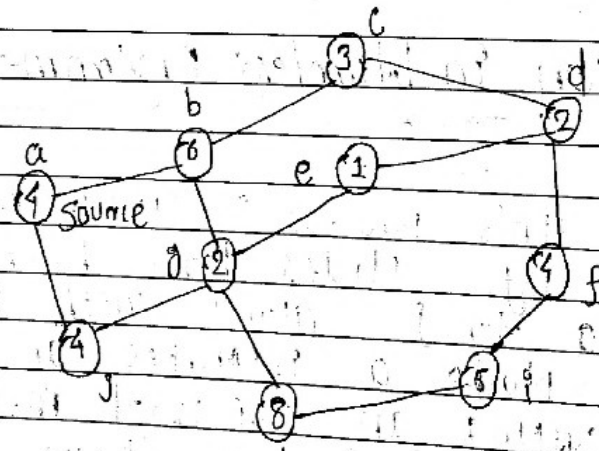
The bully election algorithm.

(a) Process 4 holds an election.
(b) Process 5 & 6 respond, telling 4 to stop
(c) Now 5 & 6 each hold an elections
(d) Process 6 tells 5 to stop
(e) Process 6 wins and tell everyone

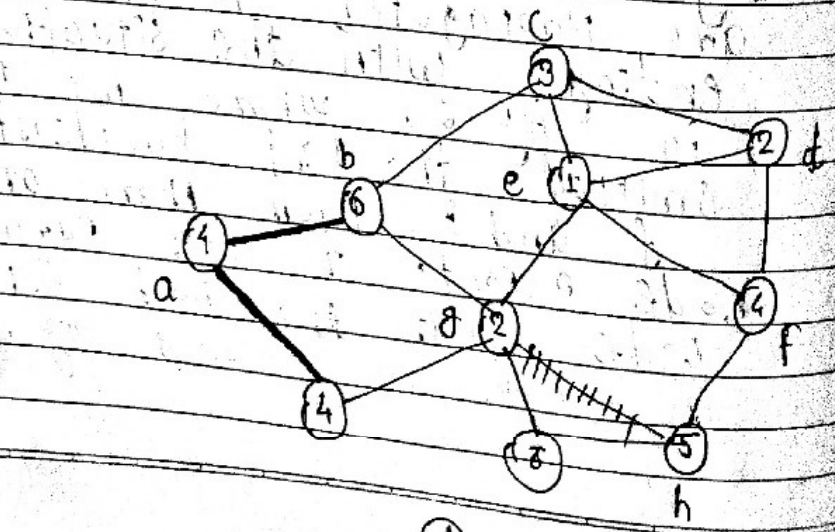
## * Election in Wireless Environment *

~~Cons~~ This process is illustrated in given fig. Nodes have been labeled a to j, along with their capacity. Node a initiates an election by broadcasting an ELECTION message to nodes b and j, as shown in fig. After that step, ELECTION message are propagated to all nodes, ending with the situation shown in fig 6, where we have a smitted the last broadcast by nodes f and i. From there on each node reports to its parents the node with the best capacity as shown in following fig.

For example when g node g receives the acknowledement from it's children e and h, it will notice that h is the best node, propagating (h,r) to its own parent, node b. In the end, the source will note that h is the best leader and will broadcast this information to all other nodes.

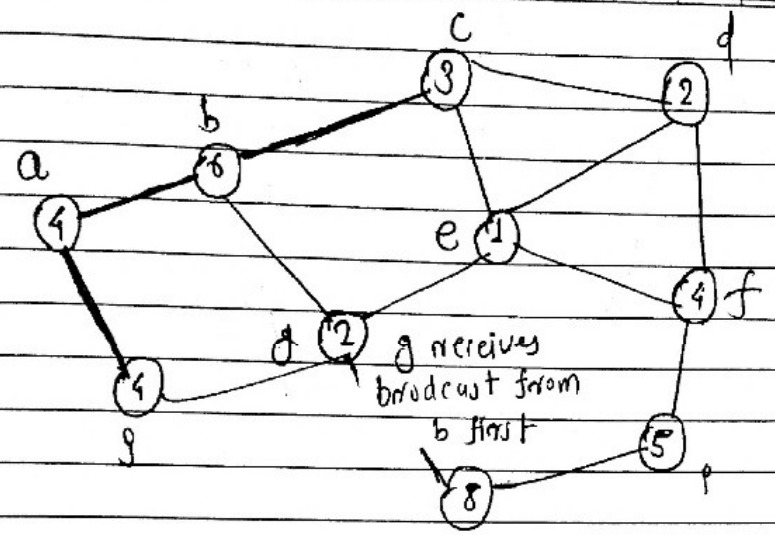



(a)

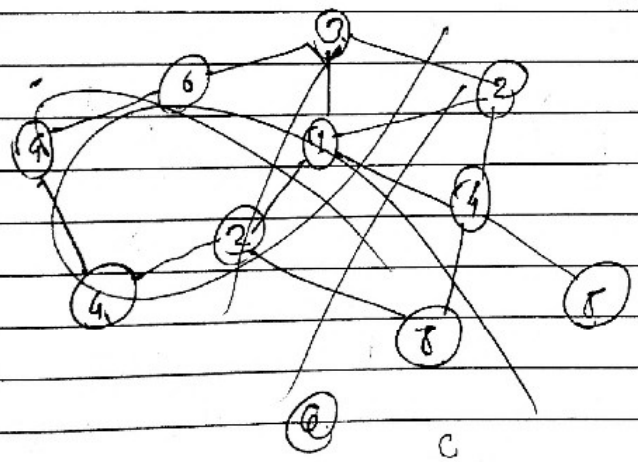



(b)

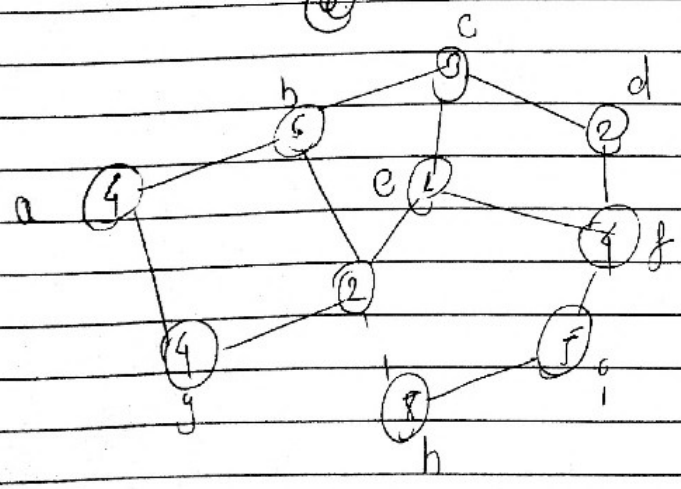
Page No.
Date




(c)



(c)




(c)