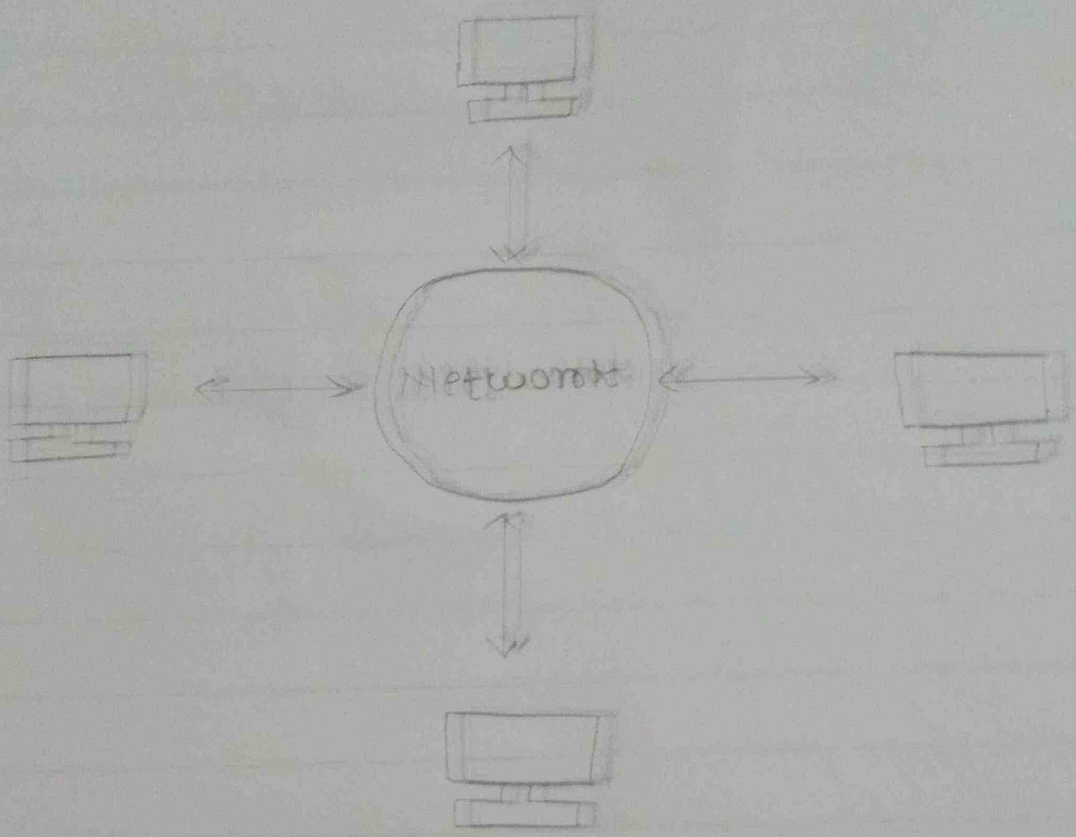


distributed system

Date / /20
Page No. 1

Aim :- Study of distributed system

Introduction :- Computer systems are undergoing a revolution. From 1945, when the modern computer era began, until about 1985, computers were large and expensive. Even minicomputers normally cost tens of thousands of dollars each. As a result, most organizations had only a handful of computers, and for lack of a way to connect them these operated independently from one another.

The amount of improvement that has occurred in computer technology in the past half century is truely staggering and totally unprecedented in other industries. From a machine that cost 10 million dollars and executed 1 instruction per second, we have come to machines that cost 1000 dollars and execute 10 million instructions per second a price gain of 10^{10}. If cars had improved at this rate in the same time period, a Rolls Royce would now cost 10 dollars and get a billion miles per gallon.

The results of these technologies is that it is now not only feasible, but easy, to put together computing systems composed of large numbers of CPUs connected by a high-speed network. They are usually called distributed systems in contrast to the previous centralized systems

consisting of a single cpu, its memory, peripherals, and some terminals.

There is only one fly in the ointment; software. Distributed systems need radically different software than centralized system do

Distrubuted System :-

Various definitions of distributed systems have been given in the literature, none of them satisfactory and none of them in agreement with any of the others for our purposes it is sufficient to give a loose characterization:

A distributed system is a collection of independent computers that appear to the users of the system as a single computer this definitions has two aspects. The first one deals with hardware: the machines are autonomous the second one deals with software: the users think of the system as a single computer. Both are essential we will come back to these points later in this chapter after going over some background material on both the hardware and the software.

Rather than going further with definitions, it is probably more helpful to give several examples of distributed systems, As a first example, Consider a network of workstations in a university or company department. In addition to

each user's personal workstations there might be a pool of processors in the machine room that are not assigned to specific users but are allocated dynamically as needed. Such a system might have a single file system, with all files accessible from all machines in the same way and using the same path name. Furthermore, when a user typed a command, the system could look for the best place to execute that command, possibly on an idle workstation belonging to someone else and possibly on one of the unassigned processors in the machine room. If the system as a whole looked and acted like a classical single processor time sharing system, it would qualify as a distributed system.

As a final example, think about a large bank with hundreds of branch offices all over the world. Each office has a master computer to store local accounts and handle local transactions. In addition each computer has the ability to talk to all other branch computers and with a central computer at headquarters.

Goals of distributed system:

Just because it is possible to build distributed systems does not necessarily mean that it is a good idea. After all, with current technology it is possible to put four floppy disk drives on a personal computer. It is just that doing so would be pointless In this section we will discuss the motivation and goals of typical distributed systems and look at their advantages and disadvantages compared to traditional centralized systems.

Advantages of distributed systems

1) The most cost-effective solution is frequently to harness a large number of cheap CPUs together in a system. Thus the leading reason for the trend toward distributed systems is that these systems potentially have a much better price ratio than a single large centralized system would have.

2) A slight variation on this theme is the observation that a collection of microprocessors cannot only give better price ratio than a single mainframe, but may yield an absolute performance that no mainframe can achieve at any price.

3) No existing machine even comes close to this, and both theoretical and engineering

Considerations make it unlikely that any machine ever will.

4) As an aside, some authors make a distinction between distributed systems, which are designed to allow many users to work together, and parallel systems,

5) A next reason for building distributed system is that some applications are inherentently distributed.

Disadvantages of distributed system:

1) Although distributed systems have their strengths, they also have their weakness.

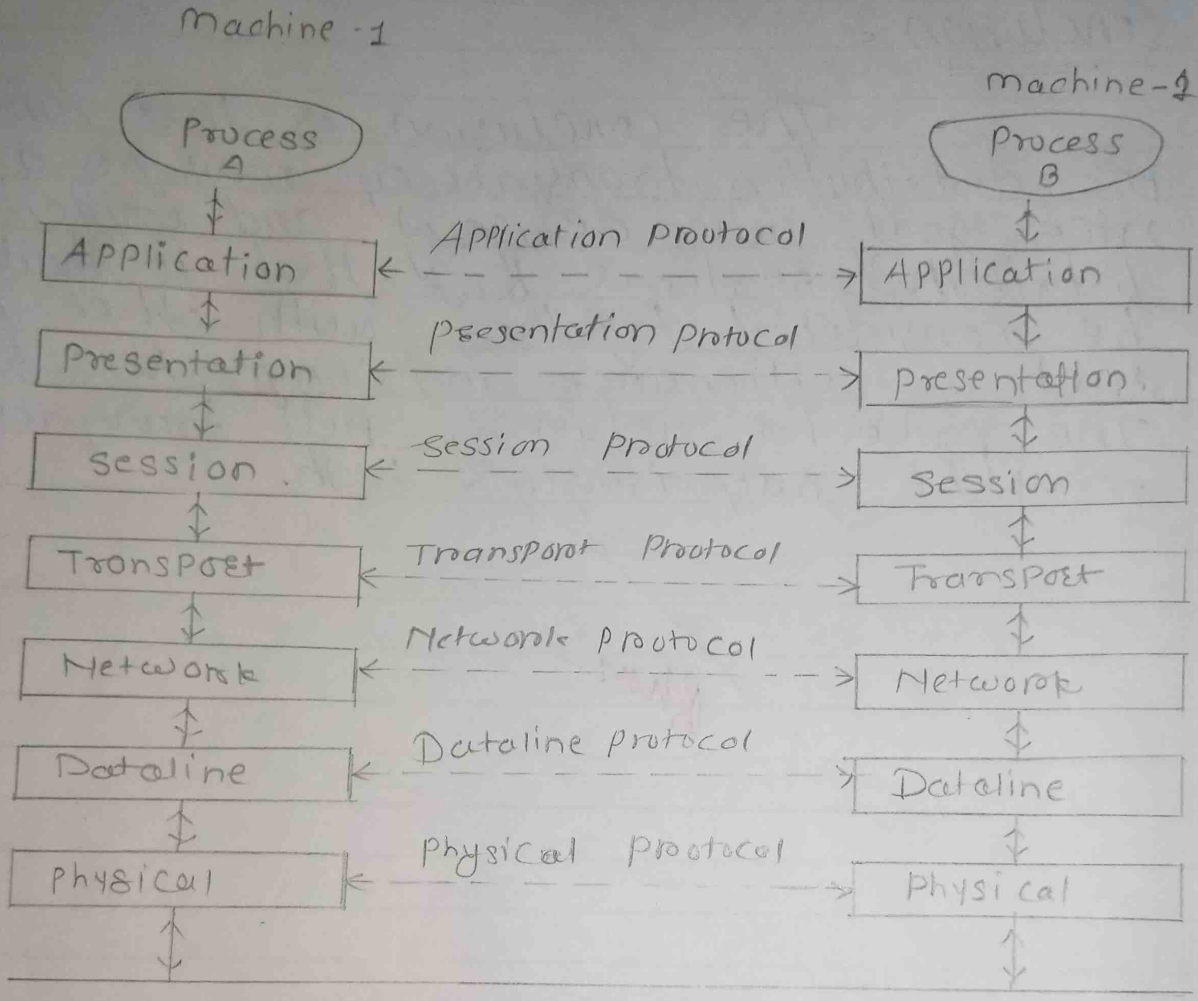
2) A second potential problems is due to the communication network. It can lose messages which required special software to be able to recover, and it can become overloaded

3) Once the system comes to depend on the network its loss or saturation can negate most of the advantages the distributed system was built to achieve.

4) If people can conveniently access data all over the system they may equally be able to conveniently access data that their they have no business looking at.

Conclusion :

The conclusion is that aiming for distribution transparency may be a nice goal when designing and implementing distributed systems, but that it should be considered together with other issues such as performance and comprehensibility. The price for achieving full transparency may be surprisingly high.


Machine -1
Machine -2
Process A
Process B
Application
Application Protocol
Application
Presentation
Presentation protocol
Presentation
Session
Session Protocol
Session
Transport
Transport Protocol
Transport
Network
Network protocol
Network
Dataline
Dataline protocol
Dataline
Physical
Physical protocol
Physical

Date / /20
Page No. 7

Aim :- Study of layer protocal in distributed system.

Layered protocols : Due to the absence of shared memory, all communication in distributed systems is based on message passing ; when process A wants to communicate with process B. it first builds a message in its own address space. then it executed a over the network to B. Although this basic idea sounds simple enough, in order A sends a brilliant new novel written in french and encoded in IBM's EBCOIC character code, and B expects the inventory of a supermarket written in English and encoded in ASCII Communication will be less than optimal.

many different agreements are needed. How many volts should be used to signal a 0-bit and how many volts for a 1-bit How does the receiver know which is the last bit of the message? How can it detect if a message has been damaged or lost, and what should it do if it finds out? How long are numbers string and other data items, and how are they represented? In short, agreements are needed at a variety of levels, varying from the low-level details of bit transmission to the high-level details of how information is to be expressed.

1) The physical Layer :-

The physical layer is concerned with transmitting the 0s and 1s. How many volts to use for 0 and 1, how many bits per second can be sent, and whether transmission can take place in both directions simultaneously are key issues in the physical layer. In addition, the size and shape of the network connector as well as the number of pins and meaning of each are of concern here.

The physical layer protocol deals with standardizing the electrical, mechanical and signaling interfaces so that when one machine sends a 0 bit it is actually received as a 0 bit and not a 1 bit.

2) The Data Link Layer :-

The physical layer just sends bits. As long as errors occur, all is well. However, real communication networks are subjects to errors, so some mechanism is needed to detect and correct them. This mechanism is the main task of the data link layer. what it does is to group the bits into units, sometimes called frames and see that each frame is correctly received. The data link layer does its work.

3) The network layer:
On a LAN, there is usually no need for the sender to locate the receiver. It just puts the message out on the network and the receiver takes it off. A wide-area network, however, consists of a large number of machines, each with some number of lines to other machines, rather like a large-scale map showing major cities and roads connecting them.
Two network-layer protocols are in widespread use, one connection oriented and one connectionless. The connection-oriented one is called x.25 and is favored by the operators of public networks such as telephone companies and the European PTTs.

4] The transport layer:
Packets can be lost on the way from the sender to the receiver. Although some applications can handle their own error recovery, others prefer a reliable connection. The job of the transport layer is to provide this service. The idea is that the session layer should be able to deliver a message to the transport layer with the expectation that it will be delivered without loss.

5] The Session layer :-

The Session layer is essentially an enhanced version of the transport layer. It provides dialog control, to keep track of which party is currently talking, and it provides synchronization facilities. The latter are useful to allow users to insert checkpoints into long transfers, so that in the event of a crash it is only necessary to go back to the last checkpoint, rather than all the way back to the beginning.

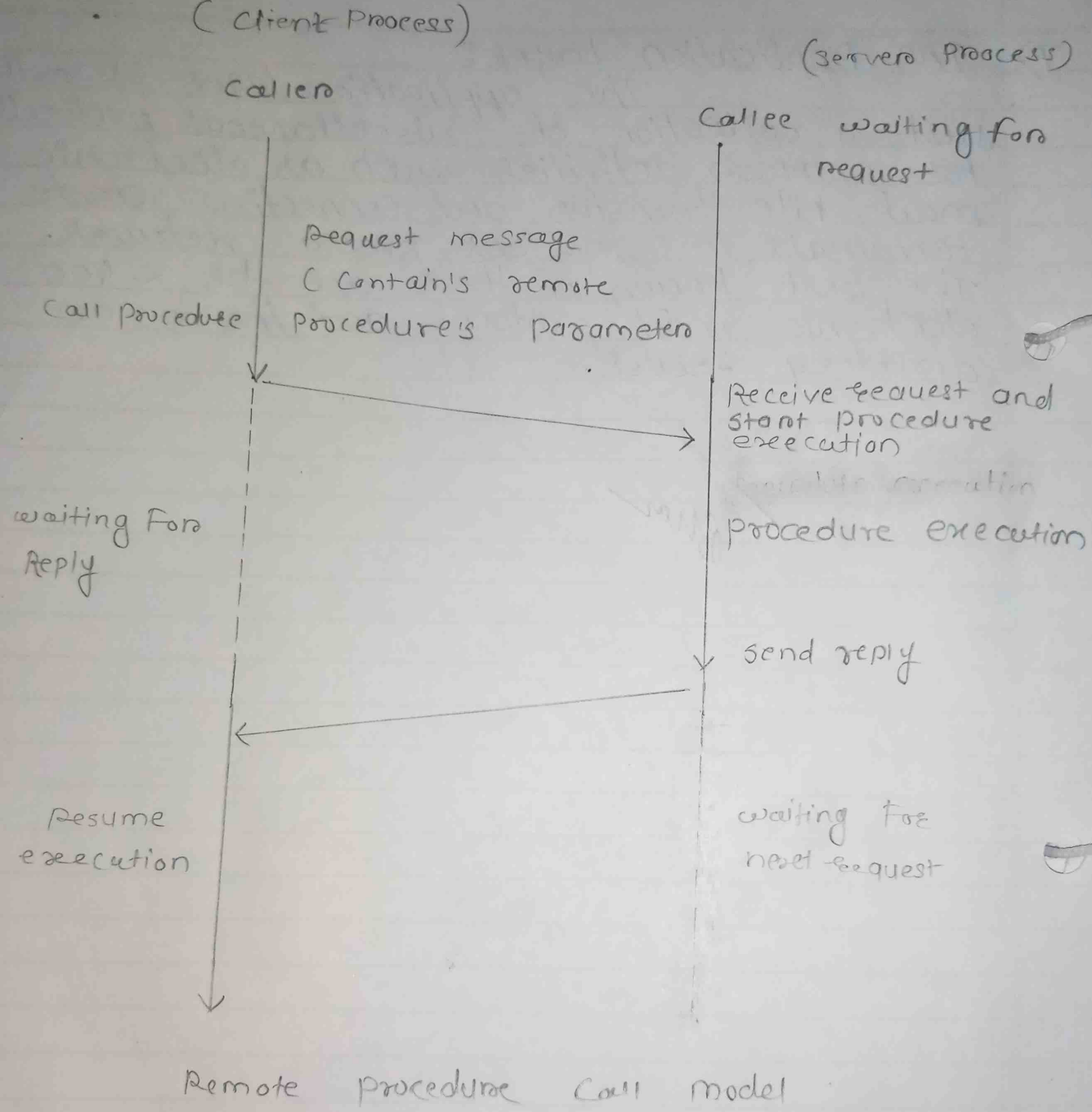
6] The presentation layer :-

Unlike the lower layers, which are concerned with getting the bits from the sender to the receiver reliably and efficiently, the presentation layer is concerned with the meaning of the bits. most messages do not consists of random bit strings, but more structured information such as peoples names, addresses, amounts of money and so on. In the presentation layer it is possible to define records containing fields like these and then have the sender notify the receiver that a message contains a particular record in a certain format.

7) The Application Layer:

The application layer is really just a collection of miscellaneous protocols for common activities such as electronic mail, file transfer and connecting remote terminals to computers over a network. The best known of these are the X.400 electronic mail protocol and the X.500 directory server.

Bhan



Remote procedure Call model

Aim: Study of RPC and group communication:

Remote procedure call (RPC) :- Although the client-server model provides a convicient way to structure a distributed operating system, it suffers from one incurable flow: the basic paradigm around which all communication is built is input/output. The procedures send and receive are fundamentally engaged in doing I/O. Since I/O is not one of the key concepts of centralized systems, making it the basis for distributed computing has struct many workers in the field as a mistake.

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 the callee in the parameters and can come back in the procedure result. No message passing or I/O at all is visible to the programmer.

Basic RPC operation :-

To understand how RPC works, it is important first to fully understand how a conventional procedure call works. Consider a call like count = read (fd, buf, nbytes); where fd is an integer, buf is an array of characters and nbytes is another

integer. If the call is made from the main program, the stack will be as shown in before the call. To make the call, the caller pushes the parameters onto the stack in order, last one first. After read and finished running it puts the return value in a register removes the return address and transfers control back to the caller. The caller then removes the parameter from the stack, returning it to the original state.

RPC Protocol :-

The first issue is the choice of the RPC protocol. Theoretically, any old protocol will do as long as it gets the bits from the client's kernel to the several major decisions to be made here, and the choices made can have a major impact on the performance. The first decision is between a connection oriented protocol and a connection oriented protocol and a connectionless protocol. With at the time the client is bound to the server, a connection is established between them. All traffic in both directions, uses this connection.

Advantages of RPC :

1] Server independent
2] Process-oriented and thread oriented models supported by RPC.
3] The development of distributed system is simple because it uses straight forward semantics and easier.
4] The code re-writing effort is minimized.

Disadvantages of RPC :

1] Content switching increases scheduling costs.
2] RPC is not a standard it is an idea that can be implemented in many ways
3] RPC does not solve the most of the distribution creation problems.
4] RPC is only interaction based. This does not offer any flexibility in terms of hardware architecture.

Group Communication :-

An underlying assumption intrinsic to RPC is that communication involves only two parties, the client and the server. Sometimes there are circumstances in which communication involves multiple processes. not just two for example consider a group of file servers cooperating to offer a single fault-tolerant file service. In such a system, it might be desirable for a client to send a

message to all servers, to make sure that the request could be carried out even if one of them crashed.

Introduction to group Communication:

A group is a collection of process that act together in some system or user-specified way. The key property that all groups have is that when a message is sent to the group itself, all members of the group receive it. It is the form of one-to-many communication. and is constrated with point-to-point communications.

Design Issues of Group Communication:

Group communication has many of the same design possibilities as regular message passing such as buffered versus unbuffered blocking versus non blocking, and so forth.

However there are also a large number of additional choices that must be made because sending to a group is inherently different from sending to a single process: furthermore groups can be organized in various ways internally. In this section we will look at some of the most important design issues and point out the various alternatives.

Group Membership :

When group communication is present, some method is needed for creating and deleting groups, as well as for allowing processes to join and leave group. One possible approach is to have a group server to which all these requests can be sent. The group server can then maintain a complete data base of all groups and their exact membership. This method is straightforward, efficient and easy to implement. Unfortunately it shares with all centralized technique, a major disadvantage a single point of failure.

Group Addressing :

In order to send a message to a group, a process must have some way of specifying which group it means. In other words, groups need to be addressed, just as processes do. One way is to give each group a unique address. If the network supports multicast, the group address can be associated with a multicast address, so that every message sent to the group address can be multicast. In this way, the message will be sent to all those machines that need it and no others.

Date / /20
Page No. 17

Aim :- Study of Clock Synchronization in distributed OS.

Clock Synchronization :

Synchronization is distributed systems is more complicated than in centralized ones because the former have to use distributed algorithms. It is usually not possible to collect all the information about the system in one place and then let some process examine it and make a decision as it done in the centralized case. In general, distributed algorithms have the following properties.

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.

When a 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 called physical clocks.

In Synchronization there are two types of clocks.

1] Physical clock :-
Time isn't a big issue in traditional centralized systems, where one or more cpus share a common bus the entire system shares the same understanding of time, right or wrong it is consistent.
In distributed systems, this is not the case. Every system, through, has its own timer that keeps the clock running.

2] Logical clock :
Logical clocks mean creating a protocol on all computers in a distributed system so that the computers can keep a uniform ordering of happenings inside some virtual time range.
In a distributed system, a logical clock is a technique for recording temporal and causative links.

Issues in clock synchronization :
A basic technique of clock synchronization is for each node to submit a time query message with the value of time 't'. This method has the following issues:
- Every node's capacity to read the clock value of another node. This can raise errors due to delays in message communication b/t nodes. The time required to prepare, deliver and get a blank message because of the lack of transmission

problems and system load can be used to calculate delay.

Clock Synchronization Algorithms

1] Cristian Algorithm :-

Cristian algorithm is a centralized clock synchronization algorithm used to synchronize time with a time server by client processes. This algorithm works well with a low latency network. where the round-trip time - time duration between the start of request and end of corresponding response is short as compared to the accuracy.

Berkeley Algorithm :-

The Berkeley Algorithm is a centralized clock synchronization mechanism is a distributed system that implies no computer has a precise timing source. The algorithm was developed by Gusella and Zatti at the University of California Berkeley in 1989.

This algorithm is an example of an active time server approach: the time server periodically sends a message to all the computers in the group. when the message is received each computer then sends back its own clock value to the time server.

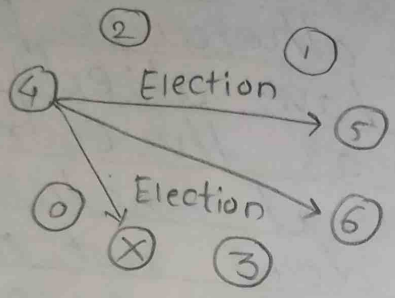
Date / /20
Page No. 20

Aim :- Study of Election Algorithm

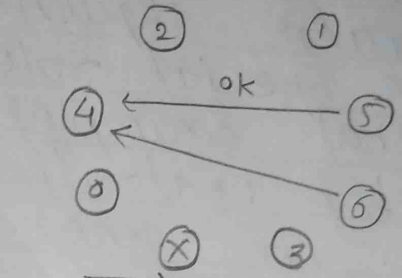
Election Algorithm :-

many distributed algorithms require one process to act as coordinator, initiator, sequencer or otherwise perform some special role. We have already seen several examples such as the coordinator in the centralized mutual exclusion algorithm. In general, it does not matter which process takes on this special responsibility but one of them has to do it. In this section we will look at algorithms for electing a coordinator. If all processes are exactly the same with no distinguishing characteristics there is no way to select one of them to be special.

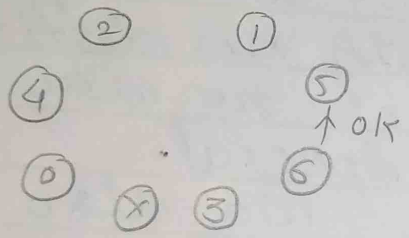
Furthermore, we also assume that every process knows the process number of every other process. What the processes do not know is which one are currently up and which ones are currently down. The goal of an election algorithm is to ensure that when an election starts it concludes with all processes agreeing on who the new coordinate is to be. various algorithms are known for example.



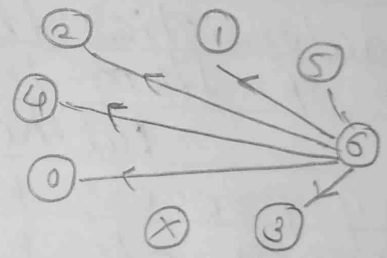
(a)



Previous Coordinator has crashed (b)



(d)



(e)

The bully algorithm

The Bully Algorithm:

As a first example consider the bully algorithm devised by Garcia-Molina (1982). When a process notices that the coordinator is no longer responding to request, it initiates an election. A process P, holds an election as follows:

1] P sends an ELECTION message to all processes with higher numbers.
2] If no one responds, P wins the election and becomes 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 one of its lower-numbered colleagues. When such a message arrives, the receiver sends an 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.

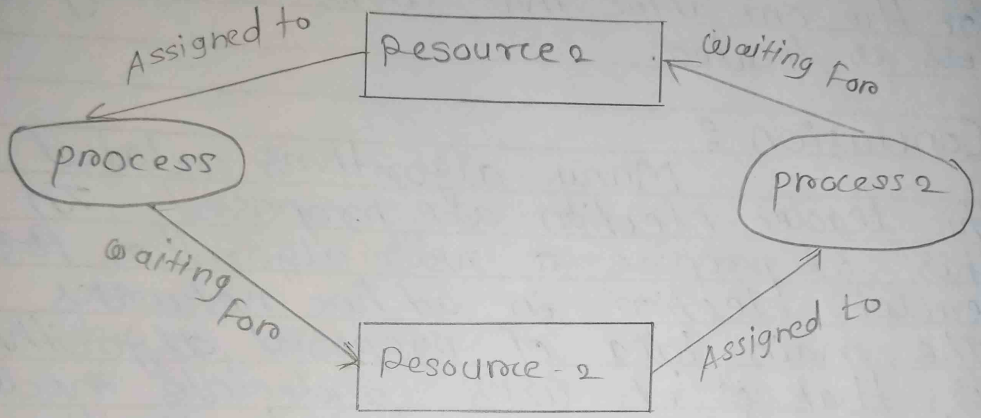
A Ring algorithm:

Another election algorithm is based on the use of a ring but without a token. We assume that the processes are physically or logically ordered, so that each process knows who its successor is. 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 skips over the successors and goes to the next member along the ring or the one after that, until a running process is located.

Conclusion :

Many algorithms related to the leader election are proposed. In this, we propose a new algorithm for leader election in ad hoc networks. The main idea of proposed algorithm is that is it uses candidate nodes. The complexity of proposed algorithm is o(n) and computer ~~simult~~ simulation shows its number of messages is smaller than other important algorithms.



Deadlock Distributed system

Date / /20
Page No. 23

Aim: Study of deadlock in distributed system.

Deadlocks in distributed System:

Deadlocks in distributed systems are similar to deadlocks in single processor systems, only worse. They are harder to avoid, prevent, or even detect, and harder to cure when tracked down because all the relevant information is scattered over many machines. In some system, such as distributed database systems, they can be extremely serious, so it is important to understand how they differ from ordinary deadlocks and what can be done about them.

Some people make a distinction between two kinds of distributed deadlocks occurs, communication deadlocks and resource deadlocks. A communication deadlock occurs, communication for example, when process A is trying to send a message to process B, which in turn is trying to send one to process C. which is trying to send one to A.

Various strategies are used to handle deadlocks. Four of the best known ones are listed and discussed below.

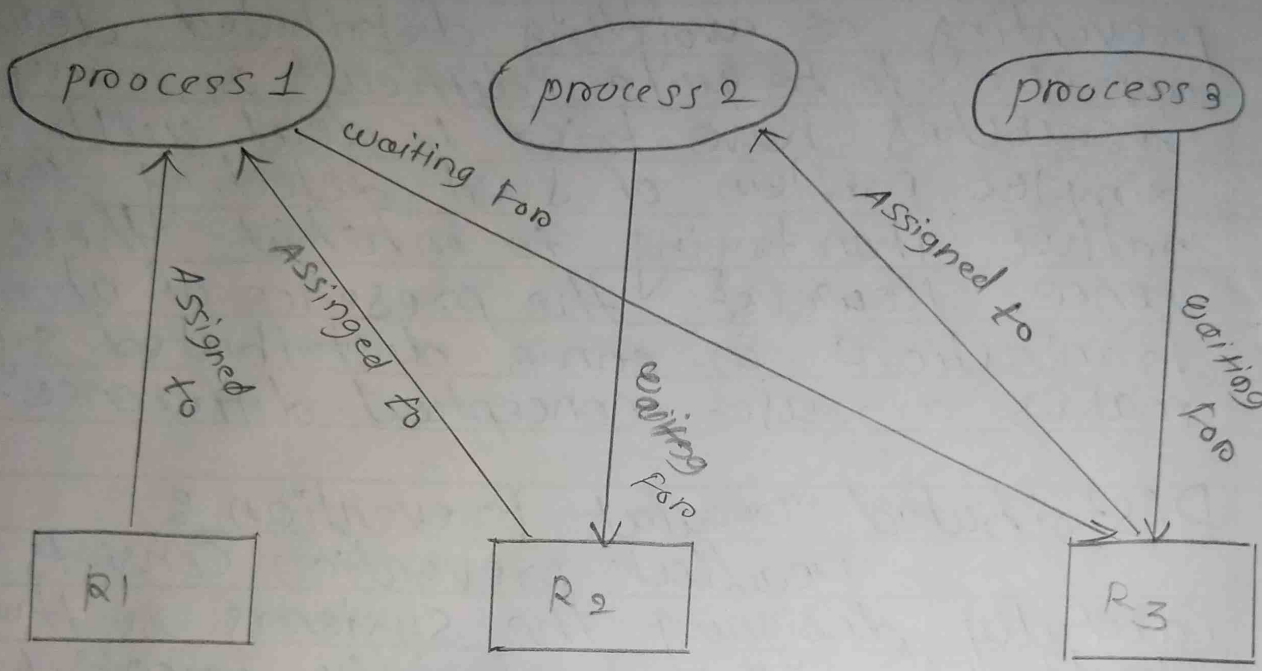
1] The ostrich algorithm
2] Detection
3] prevention 4] Avoidance

Distributed Deadlock Detection:

Finding general methods for preventing or avoiding distributed deadlocks appears to be quite difficult so many researches have tried to deal with the simpler problem of just detecting deadlocks, rather than trying to inhibit their occurrence. However, the presence of atomic transactions in some distributed systems makes a major conceptual difference.

Distributed Deadlock Prevention:

Deadlock prevention consists of carefully designing the systems so that deadlocks are structurally impossible. Various techniques include allowing processes to hold only one resource at a time, requiring processes to request all their resources initially, and making processes release all resources when asking for a new one. All of these are cumbersome in practice. A method that sometimes works is to order all the resources and works is to order all the resources and require processes to acquire them in high resources and ask for a low one thus making cycles impossible.


process 1
process 2
process 3
waiting for
Assigned to
Assigned to
Assigned to
waiting for
waiting for
R1
R2
R3

Distributed Deadlock Avoidance:

The deadlock avoidance approach handles deadlocks before they occur. It analyzes the transactions and the locks to determine whether or not waiting leads to a deadlock.

The method can be briefly stated as follows. Transactions start executing and request data items that they need to lock. The lock manager checks whether the lock is available.

Types of distributed Deadlock:

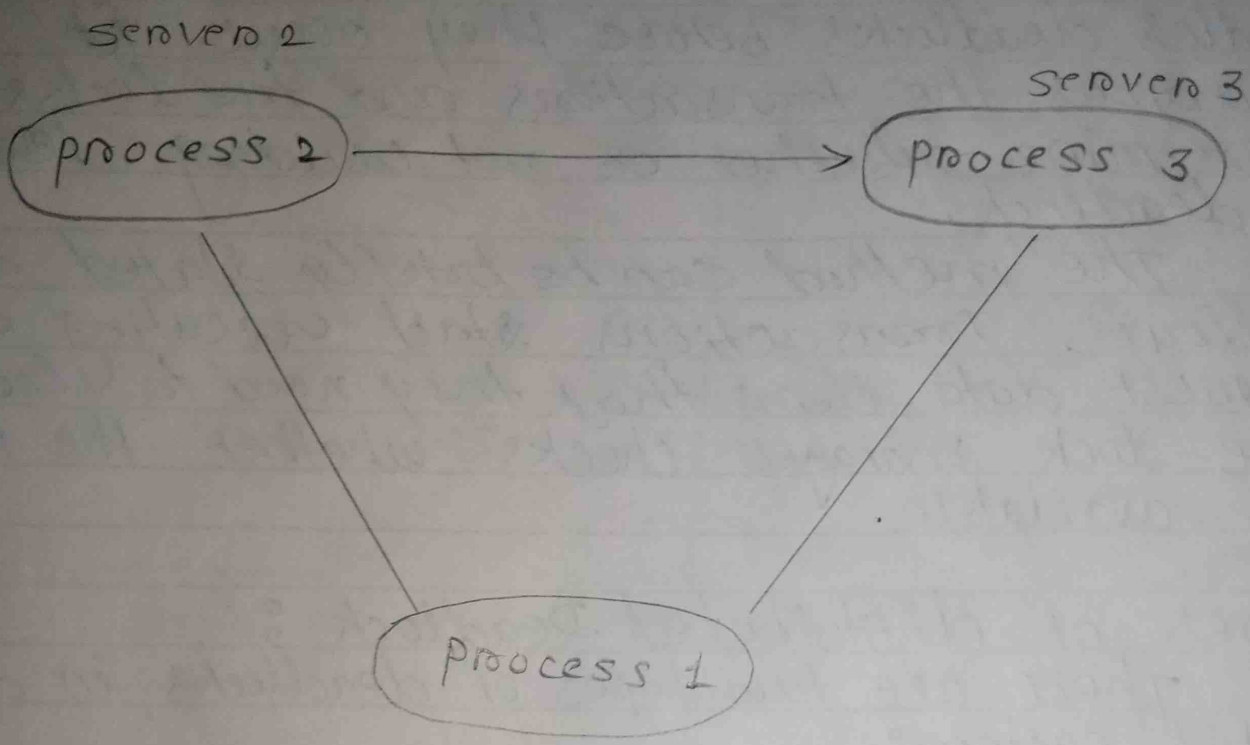
There are two types of deadlocks in distributed system:

Resource Deadlock: A resource deadlock occurs when two or more processes wait permanently for resources held by each other.

- A process that requires certain resources for its execution, and cannot proceed until it has acquired all those resources.
- It will only proceed to its execution when it has acquired all required resources.

Communication Deadlock:

On the other hand a communication deadlock occurs among a set of processes when they are blocked waiting for messages from other processes in the set in order to start execution but there are no messages in transit



Communication Deadlock.

between them when there are no messages in transit between any pair of processes in the set.
example: In a distributed system network, process 1 is trying to communicate with process 2. process 2 is trying to communicate with process 3 and process 3 is trying to communicate with process 1. In this situation, none of the processes will get unblocked and a communication deadlock occurs.

Advantages of Deadlock:
1] This situation works well for processes which perform a single burst of activity.
2] No preemption needed for Deadlock.
3] Feasible to enforce via compile-time checks.
4] Needs no run-time computation since the problem is solved in system design.

Disadvantages of Deadlock:
1] Delays process initiation
2] Processes must know future resource need.
3] Preempts more often than necessary.
4] Inherent preemption losses.

Conclusion:
The proposed deadlock detection mechanism worked perfectly fine for different sections of a distributed system. The performance of the proposed technique was measured in terms of time space the number of messages received sent.

Date / /20
Page No. 27

Aim: Study of process and processor in distributed system

Threads: In most traditional operating systems, each process has an address space and a single thread of control. In fact, that is almost the definition of a process. Nevertheless, there are frequently situations in which it is desirable to have multiple threads of control sharing one address space but running in quasiparallel. as through they were separate processes. In this section we will discuss these situations and their implications.

Introduction to Threads:

Consider, for example, a file server that occassionally has to block waiting for the disk. If the server had multiple threads of control, a second threads could run while the first one was sleeping. The net result would be a higher throughput and better performance. It is not possible because they must share a common buffer cache, which requires them to be in the same address space.

In many respects, threads are like little mini process. Each thread runs strictly sequentially and has its own program counter and stack to keep track of where it is.

Thread usage :

Threads were invented to allow parallelism to be combined with sequential execution and blocking system calls. Consider our file server example again. One possible organizations. Here one thread, the dispatcher, reads upcoming requests for work from the system mailbox.

When the worker up, it checks to see if the request can be satisfied from the shared block cache, to which all threads have access.

The main loop of the file server gets a request, examines it, and carries it out to completion before getting the next one. While waiting for the disk, the server is idle and does not process any other requests.

Design Issues for Threads Packages :

A set of primitives available to the user relating to threads is called a threads package. In this section we will consider some of the issues concerned with the architecture and functionality of threads packages.

A more general approach is to allow threads to be created and destroyed on-the-fly during execution. The thread creation call usually specifies the threads main program and a stack size, and may specify other parameters as well, for ex. a scheduling priority. The call usually returns a thread identifier to be used in subsequent calls involving

the thread.

Implementing a thread Package:

There are two main ways to implement a threads packages in user space and in the kernel. The choice is moderately controversial and a hybrid implementation is also possible.

Advantage:

1] user-level threads packages can be implemented on an operating system that does not support threads. for example the UNIX system.
2] The threads run on top of a runtime system, which is a collection of procedures that manage threads
3] The runtime system does the thread switch, store the old environment and load the new one

Disadvantage:

1] Blocking system calls are difficult to implement. letting one thread make a system call that will block the thred will stop all the threads.
2] If a thread starts running. no other thread in that process will ever run unless the first thread voluntarily gives up the CPU.
3] for the applications that are essentially CPU bound and rarely block, there is no point of using thread.

Date / /20
Page No. 30

Aim :- Study of fault tolerance in distributed system.

Fault tolerance : Fault-tolerance is the process of working of a system in a proper way in spite of the occurrence of the failures in the system. Even after performing the so many testing processes there is possibility of failure in system.

A System is said to fail when it does not meet its specification. In some cases, such as a supermarket's distributed ordering system, a failure may result in some store running out of canned beans. In other cases, such in a distributed air traffic control system a failure may be catastrophic As computers and distributed systems become widely used in safety critical missions, the need to prevent failures becomes correspondingly greater.

Any System has two major components Hardware and software. fault may occur in either of it. So there are separate techniques for fault-tolerance in both hardware and software.

Hardware Fault-tolerance Techniques :

making a hardware fault-tolerance is simple as compared to software. fault tolerance techniques make the hardware work proper and give correct result

even some fault occurs in the hardware part of the system. There are basically two techniques used for hardware fault tolerance.

1] **BIST**: BIST stands for Build in self Test. System carries out the test of itself after a creatain period of time again and again that is BIST technique for hardware fault tolerance.

2] **TMR**:- TMR is Triple modular redundancy Three redundant copies of critical components are generated and all these three copies are run concurrently.

Voting of result of all redundant copies are done and masority result is selected.

Software fault-tolerance techniques:

Software fault-tolerance techniques are used to make the software reliable in the condition of fault occurrence and failure. There are three techniques used in software fault tolerance.

1] N-version Programming:

In N-version programming N version of software are developed by N individuals or groups of developers. N-version programming is just like TMR in-hardware fault-tolerance technique.

] Recovery Blocks: Recovery blocks technique is also like the n-version programming but

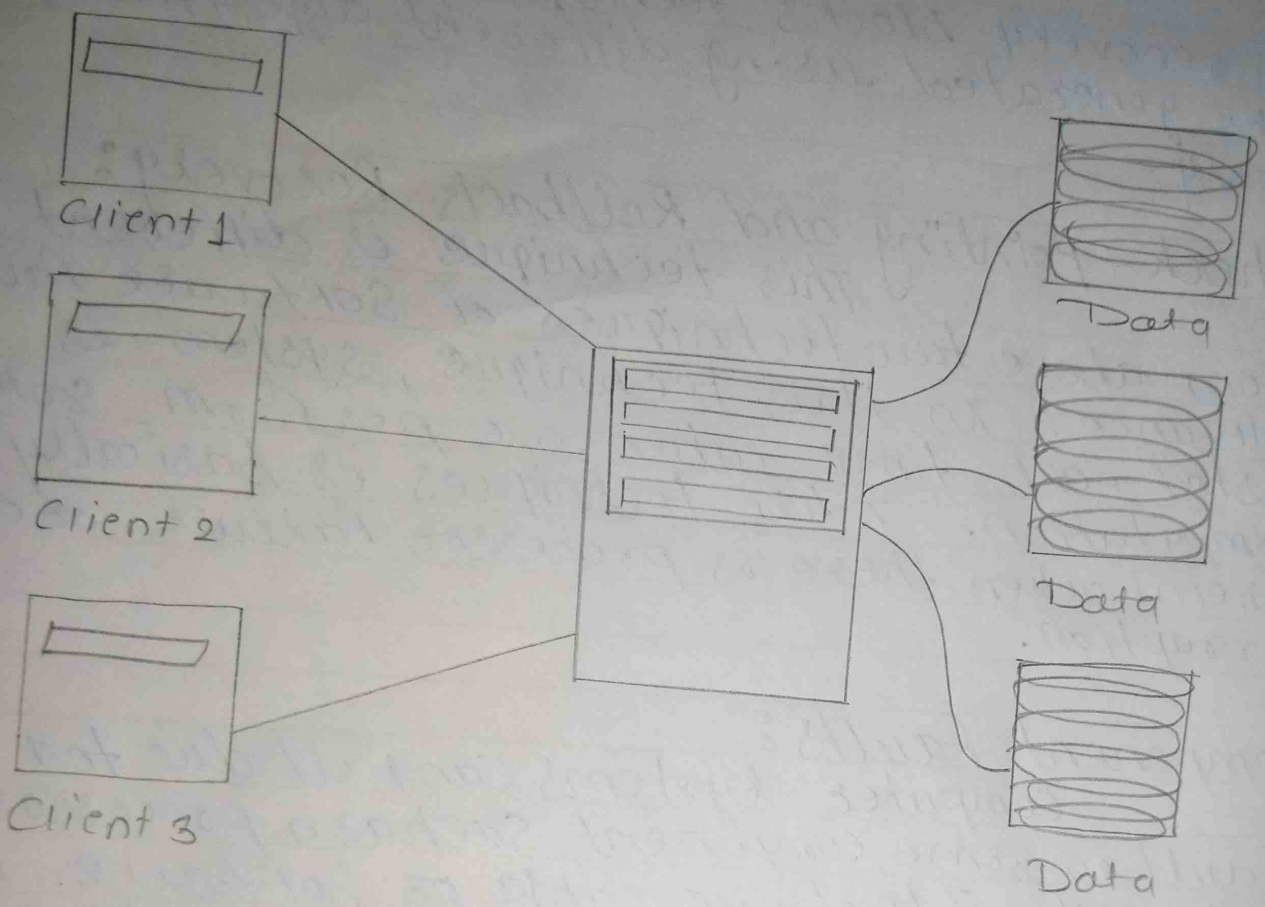
in recovery blocks technique, reludant copies are generated using different algorithms only.

3] Check-Pointing and Rollback Recovery:
This technique is different from above two techniques of software fault tolerance. In this technique, system is tested each time when we perform some computation. This techniques is basically useful when there is processor failure or data corruption.

Component Faults:
Computer systems can fail due to a fault in some component, such as a processor memory. I/O device, cable or software. A fault is a malfunction possibly caused by a design error a manufacturing error a programming error physical damage deterioration in the course of time.

Conclusion: To improve the reliability of a system, we can add fault-tolerance mechanisms. so as to tolerate faults that cannot be removed at design-time. However the resulting rise in complexity increases the probability of software faults being introduced.

Bhar


Client 1
Client 2
Client 3
Data
Data
Data

Aim :- Study of File system with design and implementation

File System design :
A distributed file system typically has two reasonably distinct components : the true file services and the directory service. The former is concerned with the operations on individual files, such as reading, writing and appending, whereas the latter is concerned with creating and managing directories adding and deleting files from directories and so on. In this section we will discuss the true file service interface; in the next one we will discuss the directory service interface.

The File Service Interface :
For any file service, whether for a single processor or for a distributed system, the most fundamental issue is : what is file ? In many systems, such as UNIX and MS-DOS, a file is an uninterpreted sequence of bytes. The meaning and structure of the information in the files is entirely up to the application programs, the operating system is not interested.

The Directory Server Interface :
The other part of the file service is the directory service, which provides operations for creating and deleting

ractical

directories, naming and renaming files and moving them from one directory to another

Distributed file System Implementation :

In the preceding section, we have described various aspects of distributed file systems from the user's perspective, that is how they appear to the users In this section we will see how these systems are implemented we will start out by presenting some experimental information about the file usage. Then we will go on to look at system structure the implemantation of catching, replication in distributed systems the concurrency control,

] on-disk structures :

Generally they contain information about total number of disk blocks, free disk blocks, location of them and etc. Below given are different on-disk structures :

1] Boot control Block :-

It is usually the first block of volume and it contains information needed to boot an operating system. In unix it is called boot block and in NTFS it is called

Teacher's Signature

Name of Practical

partition boot sector.

2] Volume Control Block:
It has information about a particular partition ex- free block count, block size and block pointers etc.

3] Directory Structure:
They store file names and associated inode numbers. In UNIX, includes file names and associated file names and in NTFS, it is stored in master file table.

4] Per-file FCB:
It contains details about files and it has a unique identifier number to allow association with directory entry. In NTFS it is stored in master file table.

II] In-memory Structure:
They are maintained in main memory and these are helpful for file system management for caching. several in memory structure given below:

Teacher's Signature

ne of Practical

1] Mount table - It contains information about each mounted volume.

2] Directory-structure cache - This catch holds the directory information of recently accessed directories.

3] system-wide open-file table - It contains the copy of FCB of each open file.

4] Per-process open-file-table - It contains information opened by that particular process and it maps with appropriate system wide open-file.

Advantages:

1] DFS allows multiple user to access or store data.
2] It allows the data to be share remotely.
3] It improved the availability of file.
4] Distributed file system provides transparency of data even if server or disk fails.

Teacher's Signature

Disadvantages :

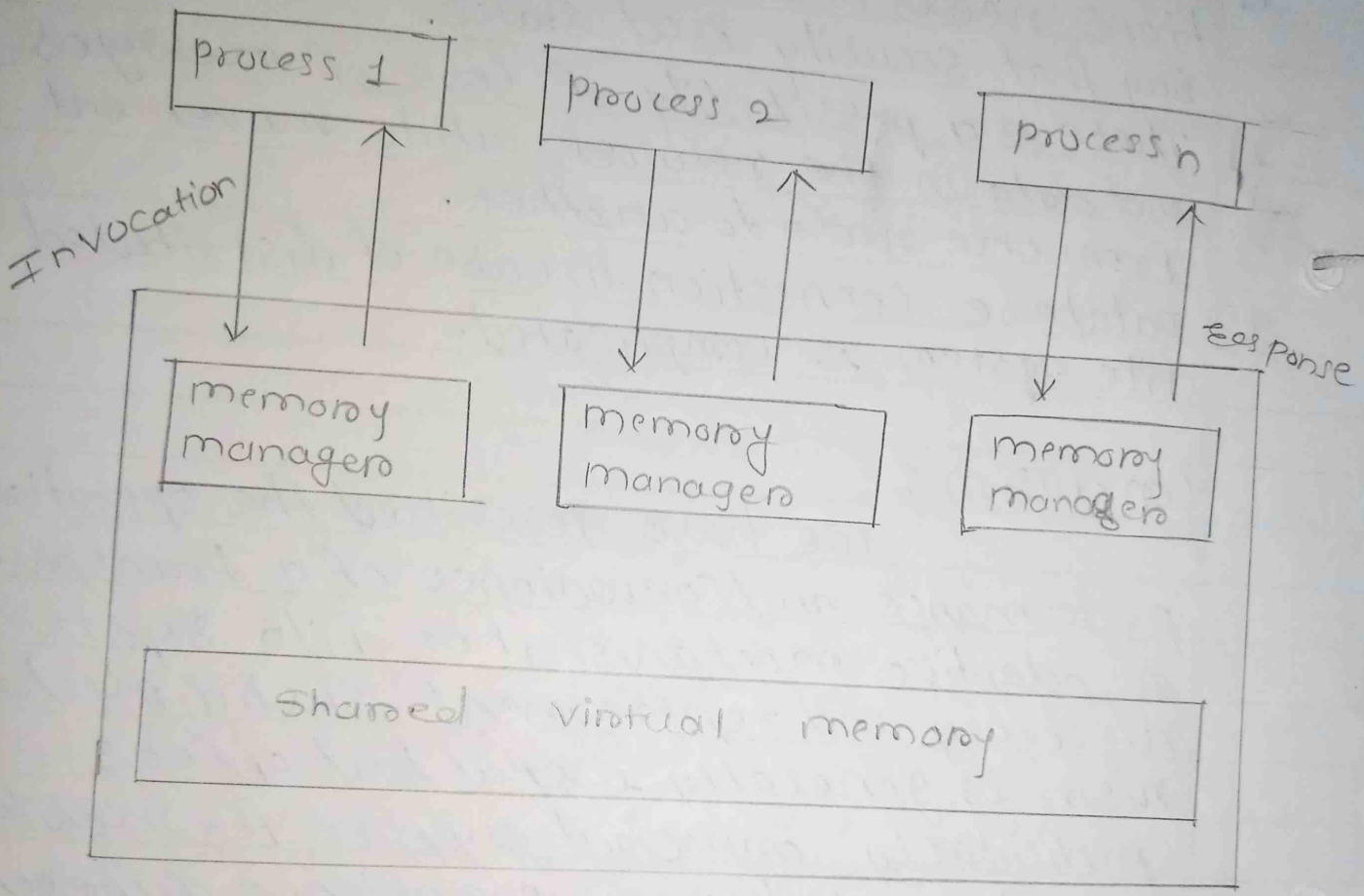
1] In distributed file system nodes and connections needs to be secured therefore we can say that security is at stake.

2] There is a possibility of lose of messages and data in the network while movement from one node to another.

3] Database connection in case of distributed file system is complicated.

Conclusion :

We have described the operation, performance, and convenience of a transparent, adaptive mechanism for file system discovery and replacement. Such a mechanism is generally useful but offers particularly important support for mobile computers which may experience difference in response time as a result of their movement.


Process 1
Process 2
Process n
Invocation
Response
memory manager
memory manager
memory manager
Shared virtual memory

of Practical

Aim :- Study of distributed shared memory

Introduction :

In Computer Science, distributed shared memory (DSM) is a form of memory architecture where physically separated memories can be addressed as a single shared address space. the term "shared" does not mean that there is a single centralized memory, but that the address space is shared i.e. the same physical address on two processors refers to the same location in memory.

Distributed Shared Memory (DSM):

DSM implements the distributed systems shared memory model in a distributed system, that hasn't any physically shared memory, shared model provides a virtual address area shared between any or all nodes.

Architecture of Distributed Shared memory-

Every node consists of 1 or additional CPU's and a memory unit, High-Speed

Teacher's Signature

Practical

Communication network is employed for connecting the nodes. A straightforward message passing system permits processes on completely different nodes to exchange one another.

Memory mapping manager unit:

Memory mapping manager routine in every node maps the native memory onto the shared computer storage. For mapping operation, the shared memory house is divided into blocks.

Communication Network unit:

Once method access information within the shared address house mapping manager maps the shared memory address to the physical memory. The mapped layer of code enforced either within the operating kernel as a runtime routine.

On-Chip memory :-

Although most computers have an external memory, self-contained chips containing a CPU and all the memory also exist.

Teacher's Signature

ame of Practical

Such chips are produced by the millions, and are widely used in cars, appliances and even toys, In this design the CPU of the chip has address and data lines that directly connect to the memory portion.

Bus-Based Multiprocessors:

- A set of parallel wires called a bus acts as a connection between CPU and memory.
- accessing of same memory simultaneously by multiple CPUs is prevented by using some algorithms.
- Cache memory is used to reduce network traffic.

Ring Based Multiprocessors:-

- There is no global centralized memory present in Ring-based DSM.
- All nodes are connected via a token passing ring.
- In ring-based DSM a single address line is divided into the shared data.

Advantages:

1] Scales well with a large number of nodes

Teacher's Signature

ractical

2] Message passing is hidden.
3] Generally cheaper than using a multiprocessor system
4] provides large virtual memory space
5] shield programmers from sending or receiving primitives.

Disadvantages:

1] Generally slower to access than non-distributed shared memory.
2] must provide additional protection against simultaneous accesses to shared data.
3] may incur a performance penalty.

Conclusion:

The implementation approach for these tasks is different for distributed memory than for shared memory architecture. In the case of the distributed memory machines the critical factor for performance is the availability of low latency, high bandwidth communication primitives.

Practical

Aim :- Study of object base and page base distributed shared memory.

object-based DSM :

object-based distributed shared memory systems allow processes on different machines to communicate through passive shared objects. This paper describes the implementation of such a system on a transputer grid. The system automatically takes care of placement and replication of objects.

Objects :-

An object is a programmer-defined encapsulated data structure, as depicted. It consists of internal data, the object state, and procedures, called methods or operations, that operate on the object state. To access or operate on the internal state, or something else. Direct access to the internal state is not allowed. This property called information hiding.

Teacher's Signature

of Practical

Page-Based DSM :

In a page-based DSM, the data to be shared among processes are organized as logical fixed-size pages that are distributed over multicomputers. whenever a page required by a process is locally available, the DSM grants access to the required page via the memory management unit (MMU) This requires simple memory access to the secondary storage that does not involve network access.

Advantages :

1] Any DSM can be used, since the runtime will decide the fate of objects for their transaction from one processor node to another. This will result in the flexibility offered to users.

2] Any consistency model as that in the ODSM can be used.

3] object access is secured since only the member method of the object can operate on them.

Teacher's Signature

Disadvantages:

1] Integration of existing system has huge overhead although changes need to be done only in the ODSM.

2] Integrated software DSM like this would have huge memory requirements

3] Complexity of the system can never be ignored.

Conclusion:

Many existing PDSMs and OBSMs are present in the real world with different features. All these systems have their pros and cons. As mentioned in the earlier section, ODSMs have scope for a lot of improvements.

f Practical

Aim :- Study of Amoeba OS with process management and memory management.

Introduction to Amoeba :

- originated at a university in Holland, 1981
- Currently used in various EU Countries.
- Built from the group up. UNIX emulation added later.
- Goal was to build a transparent distributed operating system.
- Resources, regardless of their location are managed by the system, and the user is unware of where processes are actually run.

History of Amoeba :

Amoeba originated at the Vrije universiteit, Amsterdam, the Netherlands in 1981 as a research project in distributed and parallel computing. It was designed primarily by Andrew S. Tanenbaum and three of his Ph.D. students, Frans Kaashoek, Sape J. Mullender, and Robbert van Renesse, although many other people also contributed

Teacher's Signature _____

ame of Practical

to the design and implementation. By, 1983 an initial prototype Amoeba 1.0, was operational.

The Amoeba System Architecture :

1] Assumes that a large number of CPU sare available and that each cpu has1os of Mb of memory.
2] CPUs are organised into processor pools
3] CPUs do not need to be of the same architecture (can mix SPARC, Motorola powerpc, 680x0. Intel, pentium etc).
4] when a user types a command system determines which CPUcs to execute it on CPUs can be timeshared.
5] Terminals are x-terminals or PCs running x emultors.

The Amoeba Microkernal :

- The Amoeba microkernel is used on all terminals (with an on-board processor), processors and servers.
- The microkernel manages processes and threads provides low-level memory management support supports interprocess

Teacher's Signature

actical

Communication
(Point-to-point and group)

Process Management:

- All processes are objects protected by capabilities
- Processes are managed at 3 levels by process servers, part of the microkernel
- By libraray procedures which act as interfaces. By the run server, which decides where to run the processes.
- Process management uses process descriptors.

Memory Management:

- Designed with performance, simplicity and economics in mind
- process occupies contiguous segments in memory
- All ~~the~~ of a process is constantly in memory
- process is never swapped out or paged.

Bhar

f Practical

Aim : Study of various Amoeba servet

The Bullet Server

- Designed to run on machines with large amounts of RAM and huge local disks.
- used for file storage
- Client process creates a file using the create call
- Bullet server returns a capability that can be used to read the file with.
- Files are immutable, and file size is known as file creation time. Contiguous allocation policies used.
- Handles file storage
- choice of file system is not dictated by the operating system.

The Run Server :

- used for fault tolerance and performance
- when user types a command, two decisions have to be made on which architecture should be the process be run?
- Run server manages the processor pools
- uses processes process descriptor to identify appropriate target architecture

Teacher's Signature

of Practical

- Checks which of the available processors have sufficient memory to run the processes
- Estimates which of the remaining processors has the most available computer power.

The Directory Server

- used for file server.
- Maps from ASCII names to capabilities
- Directories also protected by capabilities.
- Directory server can be used to name any object, not just files and directories.
- Handles the naming of files and other objects.
- Directories are not immutable.
- Public which contains the start of the shared public tree.

The TCP/IP server:

- A TCP/IP server has been provided to communicate with X terminals, to send and receive mail to non-Amoeba machines and to interact with other Amoeba systems via the Internet.

Teacher's Signature

Practical

- To establish a connection, an Amoeba process does an RPC with the TCP/IP server giving it a TCP/IP address.

The Boot server

- provides a degree of fault tolerance
- Ensures that servers are up and running
- If it discovers that a server has crashed, it attempts to restart it, otherwise selects another processor to provide the service.

The Replication server

- used for fault tolerance and performance.
- Replication server creates copies of files, when it has time.
- objects managed by the directory server can be replicated automatically by using the replication server.
- lazy replication:
  - when a file or other object is created, initially only one copy is made
  - Replication server can be invoked to produce identical replics when it has time.

Teacher's Signature

Practical

Other Servers:

Amoeba supports various other servers. These include a disk server (used by the directory server for storing its arrays of capability pairs), various other I/O servers, a time-of-day server and a random number server (useful for generating ports, capabilities and FLIP addresses). The so-called Swiss Army knife server deals with many activities that have to be done later by starting up processes at a specified time in the future mail servers deal with incoming and outgoing electronic mail.

Practical

Aim :- Study of process management and memory management in Mach OS.

The Introduction of Mach

Mach's development followed an evolutionary path from BSD UNIX systems.

Mach code was initially developed inside the 4.2BSD, kernel, with BSD kernel components replaced by mach components as the mach components were completed. mach was propelled to the forefont of industry attention when the open software foundation.

History of Mach

mach's earliest roots go back to a system called RIG (Rochester Intelligent Gateway). which began at the university of Rochester in 1975. Its main research goal was to demonstate that operating systems could be structured in a modular way.

Teacher's Signature

Practical

Goals of Mach

1] Providing a base for building other operating systems (eg. UNIX)
2] Supporting large sparse address spaces.
3] Allowing transparent access to network resources.
4] Making mach portable to a larger collection of machines.

Process Management in mach

Process management in mach deals with processes, threads and scheduling.

Processes:

- The process port is used to communicate with the kernel.
- The bootstrap port is used for initialization when a process starts up.
- The exception port is used to report exceptions caused by the process. Typical exceptions are division by zero and illegal instruction executed.

Teacher's Signature

ractical

Threads :

The active entities in mach are the threads. They execute instructions and manipulate their registers and address spaces. Each thread belongs to exactly one process.

A process cannot do anything unless it has one or more threads. All the threads in a process share theaddress space and all the process-wide resources.

Scheduling :

mach scheduling has been heavily influenced by its goal of running on multiprocessors. Since a single-processor system is effectively a special case of a multiprocessor can be assigned to processor sets by software.

Memory management in mach :

- mach has a powerful, elaborate, and highly flexible memory management system based on paging.
- The code of mach's memory managemont is split into three parts.

Teacher's Signature _____

The third part of the memory management code runs as a user process called a memory manager.

Sharing plays an important role in mach. No special mechanism is needed for the threads in a process to share objects.

Virtual Memory :

The conceptual model of memory that mach user processes see is a large, linear virtual address space.

A key concept relating to the use of virtual address space is the memory objects.

In reality, there is a great deal more to say. mach provides a great deal of fine-grained control over how the virtual pages are used.

To start with, the address space can be used in a sparse way.

Aim :- Study of DCE Thread and various services.

Introduction to DCE Threads

The DCE threads package is based on the P10003.4a POSIX standard. It is a collection of user-level library procedures that allow processes to create delete, and manipulate threads. However, if the host system comes with a (kernel) threads package, the vendor can set up DCE to use it.

A thread can be in one of four states. A running thread is one that is actively using the CPU to do computation. A ready thread is willing and able to run, but cannot run because the CPU is currently busy running another thread.

Scheduling :

Thread scheduling is similar to process scheduling, except that is visible to the application. The scheduling algorithm determines how long a thread may run, and which thread

Teacher's Signature

Practical

next. Just as with process scheduling, many thread scheduling algorithms are possible.

Synchronization :-

DCE provides two ways for threads to synchronize : mutexes and condition variables. Mutex are used when it is essential to prevent multiple threads from accessing the same resource at the same time.

By requiring a thread to first successfully lock the mutex associated with the list before touching the list, correct operation can be ensured.

Thread calls :

The DCE threads package has a total of 54 primitives. many of these are not strictly necessary but are provided for convience only. this approach is some calculate what analogous to a four-function pocket calculate that has keys not only for +, -, x and /. but also has keys +1, -1, x2, x10, xπ, 1/2 and

Teacher's Signature

ame of Practical

I/o on the grounds that these save the user time and effort.

DCE Services

1] Remote procedure call (RPCs)

A distributed application based on the client/server model consists of two parts: The client side of the application, which runs on another machine on the network and fulfills the services requests. The two pieces of code on two different machines need to be able to communicate across the network.

2] Time service:

Time is an important concept in most distributed systems. To see why, consider a research program in radio astronomy. A number of radio telescopes spread all over the world observe the same celestial radio source simultaneously, accurately recording the data and the observation time, the data are sent over a network to a central computer for processing.

Teacher's Signature

of Practical

3) Directory Service:

A major goal of DCE is to make all resources accessible to any process in the system, without regard to the relative location of the resource uses and the resource provided these resources include users, machines, cells servers, services files, security data and many others.

4) Security Service:

In most distributed system, security is a major screen concern. The system administrator may have definite ideas about who can use which resource and many users may want their files and mailboxes protected from prying eyes.

In a distributed system consisting of potentially untrustworthy machines communicating over an insecure network, this solution network.

Ram
23-5-2022

Teacher's Signature