**KARPAGAM ACADEMY OF HIGHER EDUCATION**

Coimbatore-641 021

(For the candidates admitted from 2017 onwards)

**DEPARTMENT OF COMPUTER SCIENCE, CA & IT**

**SUBJECT NAME : DISTRIBUTED OPERATING SYSTEM**
**SEMESTER III**
**SUBJECT CODE: 17CSP305A** **CLASS: II M.SC CS**

**COURSE OBJECTIVE:**

This course focuses on software issues in the design and implementation of modern computer systems particularly the operating systems and distributed algorithms.

**COURSE OUTCOME:**
The objectives is to learn the fundamentals of

- Distributed processes (synchronization communication and scheduling)
- Concurrent processes and programming
- Process interaction and Process scheduling
- Distributed file systems and Distributed shared memory
- Security issues in network and distributed environments

**UNIT-I**

Fundamentals – message passing – Remote procedure calls : Introduction – the RPC model – transparency of RPC – Implementing RPC mechanism –stub generation – RPC messages – marshaling arguments and results – server management – parameter passing semantics – call semantics.

**UNIT- II**

Distributed shared memory : Introduction – general architecture of DSM systems – design and implementation of DSM – granularity – structure of shared memory space – replacement strategy – heterogeneous DSM – advantages of DSM.

**UNIT- III**

Synchronization: Introduction – clock synchronization – event ordering – mutual exclusion. Resource management: Introduction – desirable features of a good global scheduling algorithm – task management approach – load balancing approach – load sharing approach.

**UNIT- IV**

Distributed file system: Introduction – desirable features of a good distributed file system – file models – file accessing models.

Naming: Introduction – desirable features of a good naming system – fundamental terminologies and concepts.

**UNIT- V**

Security: Introduction – potential attacks to computer system – cryptography.

**SUGGESTED READINGS**

**TEXT BOOK**

1. Pradeep, K. Sinha.(1997). Distributed Operating Systems Concepts and Design (1st ed.). New Delhi: Prentice Hall of India.

**REFERENCES**

1. Paul, J. Fortier. (1998). Design of Distributed Operating System concepts and Technology (1st ed.). New Delhi: Tata McGraw Hill.
2. Andrew, S. Tanenbaum. (1995). Distributed Operating System. New Delhi: Pearson Education.

**WEB SITES**

1. http://staff.um.edu.mt/csta1//courses/lectures/csm202/os17.html
2. http://www.inf.uni-konstanz.de/dbis/teaching/ss06/os/ch14-wrongNumber.pdf
3. https://www.cs.columbia.edu/~smb/classes/s06-4118/l26.pdf

**ESE MARKS ALLOCATION**

| S.No | Category | Marks |
|---|---|---|
| **1.** | **Section A**<br>20 X1 = 20<br>(Online Examination) | 20 |
| **2.** | **Section B**<br>5 X 6 = 30<br>(Either 'A' or 'B' Choice) | 30 |
| **3.** | **Section C**<br>1 X 10= 10<br>(Compulsory Question) | 10 |
| **4.** | **Total** | **60** |

**KARPAGAM ACADEMY OF HIGHER EDUCATION**
**(Deemed to be University)**

**(Established Under Section 3 of UGC Act 1956)**

**(For the candidates admitted from 2017onwards)**

**Coimbatore - 641021.**

**DEPARTMENT OF CS, CA & IT**

**Subject Name:** DISTRIBUTED OPERATING SYSTEM
**Subject Code:** 17CSP305A
**Class** : II M. Sc CS **Semester:** III **Batch:** 2017-2019

## LECTURE PLAN

| Sl.No | Lecture Duration (Periods) | Topics to be covered | Support Materials |
|---|---|---|---|
| **Unit- I** | | | |
| 1 | 1 | Introduction to Distributed Operating System | **T1:** 1-4,**W1** |
| 2 | 1 | Commuting System Models | **T1:** 5-18 |
| 3 | 1 | Message Passing | **T1:** 114-138 |
| 4 | 1 | Remote Procedure Call -introduction | **T1:** 167-168,**W1** |
| 5 | 1 | The RPC Model | **T1:** 168-169 |
| 6 | 1 | Transparency of RPC , Implementing RPC Mechanism | **T1:** 170-173 |
| 7 | 1 | Stub Generation, RPC Message | **T1:** 174-176 |
| 8 | 1 | Marshalling Arguments & Results, Server Management | **T1:** 177-182 |
| 9 | 1 | Parameter Passing And Call Semantics | **T1:**183-186 |
| **Total No.Of Periods Planned** | **9** | | |
| **Sl.No** | **Lecture Duration (Periods)** | **Topics to be covered** | **Support Materials** |
| **Unit- II** | | | |
| 1 | 1 | Distributed Shared Memory Introduction | **T1:** 231-232 |
| 2 | 1 | General Architecture Of DSM Systems | **T1:** 233 |
| 3 | 1 | Design and Implement of DSM | **T1:** 234 |
| 4 | 1 | Granularity | **T1:** 235 |
| 5 | 1 | Structure of Shared Memory Space | **T1:** 237 W2 |
| 6 | 1 | Replacement strategy | **T1:** 262-263 |
| 7 | 1 | Heterogeneous DSM | **T1:** 267-269 |
| 8 | 1 | Advantages of DSM. | **T1:** 270-271 |

| Sl.No | Lecture Duration (Periods) | Topics to be covered | Support Materials |
|---|---|---|---|
| 9 | 1 | Recapitulation & Important Questions Discussion | - |
| **Total No.Of Periods Planned** | **9** | | |
| **Sl.No** | **Lecture Duration (Periods)** | **Topics to be covered** | **Support Materials** |
| **Unit- III** | | | |
| 1 | 1 | Synchronization Introduction | **T1:** 282 |
| 2 | 1 | Clock synchronization | **T1:** 283-291 |
| 3 | 1 | Event ordering | **T1:** 292-298 |
| 4 | 1 | Mutual exclusion | **T1:** 299-304 |
| 5 | 1 | Resource management Introduction<br>Desirable features of a good global scheduling algorithms | **T1:** 347-350 |
| 6 | 1 | Task management approach | **T1:** 351-354 **W3** |
| 7 | 1 | Load balancing approach | **T1:** 355-366 |
| 8 | 1 | Load sharing approach | **T1:** 367-370 |
| 9 | 1 | Recapitulation & Important Questions Discussion | |
| **Total No.Of Periods Planned** | **9** | | |
| **Sl.No** | **Lecture Duration (Periods)** | **Topics to be covered** | **Support Materials** |
| **Unit- IV** | | | |
| 1 | 1 | Distributed file system Introduction | **T3:** 421-422 |
| 2 | 1 | Desirable features of a good distributed file system | **T3:** 423-425 |
| 3 | 1 | File models | **T3:** 426-427 |
| 4 | 1 | File accessing models | **T3:** 427 W4 |
| 5 | 1 | Accessing Remote files, Unit of Data Transfer | **T3:** 428-430 |
| 6 | 1 | Naming: Introduction | **T3:** 496 |
| 7 | 1 | Desirable features of a good naming system | **T3:** 497-498 |
| 8 | 1 | Fundamental terminologies and concepts | **T3:** 499-508 |
| 9 | 1 | Recapitulation & Important Questions Discussion | - |

| Total No.Of Periods Planned | 9 | | |
|---|---|---|---|
| **Sl.No** | **Lecture Duration (Periods)** | **Topics to be covered** | **Support Materials** |
| | | **Unit- V** | |
| 1 | 1 | Security | **TP:**565-566 |
| 2 | 1 | Introduction | |
| 3 | 1 | Goals of Computer Security | |
| 4 | 1 | Potential Attacks To Computer System | **TP:**567-574 |
| 5 | 1 | Passive Attacks | |
| 6 | 1 | Active Attacks | |
| 7 | 1 | Cryptography | **TP:**575-585 |
| 8 | 1 | Basic concepts And terminology | |
| 9 | 1 | Key Distribution Problem | |
| 10 | 1 | Recapitulation & Important Questions Discussion | - |
| 11 | 1 | Discussion on Previous ESE Questions | - |
| 12 | 1 | Discussion on Previous ESE Questions | - |
| **Total No.Of Periods Planned** | **12** | | |
| **Overall Total (All Units)** | **48** | | |

## Support Materials:

## Text Book:

**T1→** Pradeep K. Sinha.1997. Distributed Operating Systems Concepts and Design, 1st Edition, Prentice Hall of India, New Delhi.

## Websites:

**W1→**http://en.m.wikipedia.org/wiki/Distributedoperatingsystem

**W2→** https://www.cs.columbia.edu/~smb/classes/s06-4118/l26.pdf
**W3→**http://staff.um.edu.mt/csta1//courses/lectures/csm202/os17.html
**W4→**http://www.inf.uni-konstanz.de/dbis/teaching/ss06/os/ch14-wrongNumber.pdf

# UNIT I

# SYLLABUS

**UNIT-I**

Fundamentals – message passing – Remote procedure calls : Introduction – the RPC model – transparency of RPC – Implementing RPC mechanism –stub generation – RPC messages – marshaling arguments and results – server management – parameter passing semantics – call semantics.

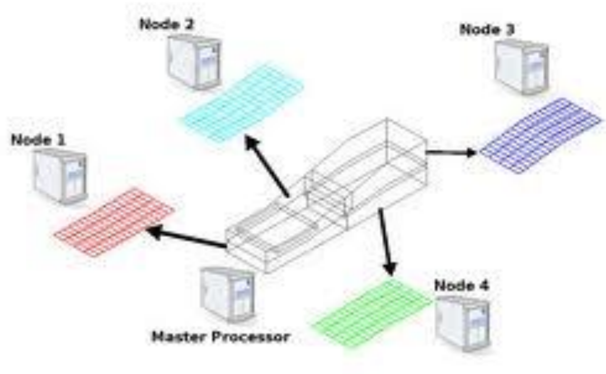
**INTRODUCTION**

**FUNDAMENTALS:**

A **distributed operating system** is a software over a collection of independent, networked, communicating, and physically separate computational nodes. Each individual node holds a specific software subset of the global aggregate **operating system**. Each subset is a composite of two distinct service provisionary.

Parallel processing is the processing of program instructions by dividing them among multiple processors with the objective of running a program in less time. In the earliest computers, only one program ran at a time. A computation-intensive program that took one hour to run and a tape copying program that took one hour to run would take a total of two hours to run. An early form of parallel processing allowed the interleaved execution of both programs together. The computer would start an I/O operation, and while it was waiting for the operation to complete, it would execute the processor-intensive program.

The total execution time for the two jobs would be a little over one hour. Parallel processing is also called parallel computing. In the quest of cheaper computing alternatives parallel processing provides a viable option. The idle time of processor cycles across network can be used effectively by sophisticated distributed computing software. The term parallel processing is used to represent a large class of techniques which are used to provide simultaneous data processing tasks for the purpose of increasing the computational speed of a computer system. Advantages:- Faster execution time., so higher throughput. Disadvantages:- More hardware required, also more power requirements. Not good for low power and mobile devices.




**MESSAGE PASSING - DISTRIBUTED PROCESSING:**

Distributed processing is a phrase used to refer to a variety of computer systems that use more than one computer (or processor) to run an application. This includes parallel processing in which a single computer uses more than one CPU to execute programs. More often, however, distributed processing refers to local-area networks (LANs) designed so that a single program can run simultaneously at various sites. Most distributed processing systems contain sophisticated software that detects idle CPUs on the network and parcels out programs to utilize them. Another form of distributed processing involves distributed databases. This is databases in which the data is stored across two or more computer systems. The database system keeps track of where the data is so that the distributed nature of the database is not apparent to users.

## ADVANTAGES

**REMOTE PROCEDURE CALL - DISTRIBUTED PROCESSING:**

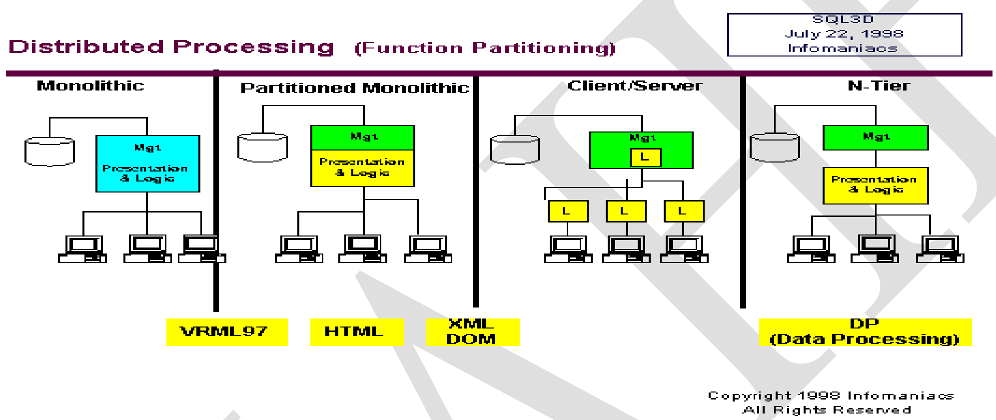
• **Quicker response time**: By locating processing power close to user, response time is typically improved. This means that the system responds rapidly to commands entered by users

• **Lower costs:** Long-distance communication costs are declining at a slower rate than the cost of computer power. Distributed processing can reduce the volume of data that must be transmitted over long-distances and thereby reduce long-distance costs.

• **Improved data integrity**: High degrees of accuracy and correctness may be achieved by giving users control over data entry and storage.

• **Reduced host processor costs**: The productive life of a costly mainframe can be extended by off-loading some its processing tasks to other, less expensive machines (whose total costs usually a fraction of the cost needed to up-grade the central processor).

• **Resource sharing:** One of the main advantages of developing microcomputer networks is because they make it possible to share expensive resources such as high-speed, color laser printers, fast data storage devices, and high-priced software packages




**RPC MODEL -**

Each segment performs partial processing dictated by the way the task is partitioned The result obtained from the computation in each segment is transferred to the next segment in the pipeline The final result is obtained after the data have passed through all segments Can imagine that each segment consists of an input register followed by an combinational circuit A clock is applied to all registers after enough time has elapsed to perform all segment activity The information flows through the pipeline one step at a time .

**TRANSPARENCY OF RPC**

Vector processors provide high-level operations that work on vectors -- linear arrays of numbers.

- The computation of each result (in vector processor) is independent of the computation of previous results.
- A single vector instruction specifies a great deal of work - it is equivalent to executing an entire loop.
- Vector instructions that access memory have a known access pattern. If the vector's elements are all adjacent, then fetching the vector from a set of heavily interleaved memory banks works very well.

**MAJOR TECHNIQUES**

- Multiple pipelined functional units that operate concurrently
- Asynchronous banks of interleaved memory
- Independent instruction and data caches
- Multiple buses to transfer data, addresses, and control signals

**IMPLEMENTING RPC MECHANISM**

In the matrix addition example, the inner loop (the J loop) can be vectorized and the outer loop can be pipelined. Basic vector architecture: most of today's vector machines are vector-register machine. All vector operations are among vector registers, except load and store.

Client-server model vs. RPC „

Client-server: ‰building everything around I/O ‰all communication built in send/receive ‰distributed computing look like centralized computing „RPC allow to call procedures located on other machines.

RPC principle

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.

With RPC:

When the message arrives at the server, the kernel passes it up to a server stub that is bound with the actual server. The server stub will have called receive and be blocked waiting for incoming messages.

The server stub unpacks the parameters from the message and then calls the server procedure in the usual way.

From the server's point of view, it is as though it is being called directly by the client-the parameters and return address are all on the stack where they belong and nothing seems unusual.

The server performs its work and then returns the result to the caller in the usual way. Our case: the server will fill the buffer, with the data. This buffer will be internal to the server stub.

Basic RPC operation – steps

1. The client procedure calls the client stub in the normal way.
2. The client stub builds a message and traps to the kernel.
3. The kernel sends the message to the remote kernel.
4. The remote kernel gives the message to the server stub.
5. The server stub unpacks the parameters and calls the server.
6. The server does the work and returns the result to the stub.
7. The server stub packs it in a message and traps to the kernel.
8. The remote kernel sends the message to the client's kernel.
9. The client's kernel gives the message to the client stub.
10. The stub unpacks the result and returns to the client.

**SUB GENERATION -**

- Start-up time: it comes from pipelining latency. Initiation rate: the time per result once a vector instruction is running
- Vector length control: although multiple function units and vector registers are available, the actual length of the vector under operation is a variable. Usually a vector length register (VLR) is used to keep track of the length of a vector. The register can also specify a maximum vector length (MVL). If the vector is longer than the MVL, compiler will be responsible to break it up and process them separately. This is called strip mining.

**STUB GENERATION - MULTI COMPUTER AND COMPUTER NETWORKS**

A computer made up of several computers. The term generally refers to an architecture in which each processor has its own memory rather than multiple processors with a shared.

**Characteristics of Multiprocessors**

Multiprocessors System = MIMD (Multiple Instruction Multiple Data)

An interconnection of two or more CPUs with memory and I/O equipment a single CPU and one or more IOPs is usually not included in a multiprocessor system. Unless the IOP has computational facilities comparable to a CPU. Computation can proceed in parallel in one of two ways,

**RPC MESSAGE - SYNCHRONIZATION**

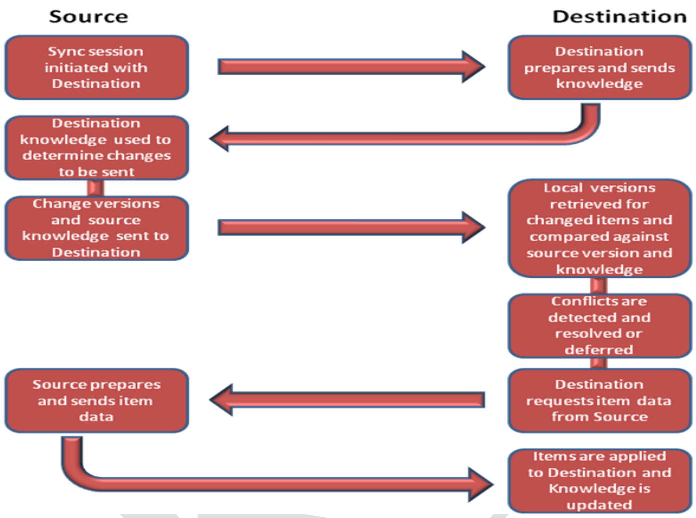
A system can be both multiprocessing and multiprogramming, only one of the two, or neither of the two of them, synchronization refers to one of two distinct but related concepts: synchronization of processes, and synchronization of data. Process synchronization refers to the idea that multiple processes are to join up or handshake at a certain point, so as to reach an agreement or commit to a certain sequence of action. Data synchronization refers to the idea of keeping multiple copies of a dataset in coherence with one another, or to maintain data integrity. Process synchronization primitives are commonly used to implement data.

**MARSHALING ARGUMENTS AND RESULT - INTERPROCESS COMMUNICATION (IPC).**

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

- Information sharing
- Computational Speedup
- Modularity
- Convenience
- Privilege separation

**SERVER MECHANISM** - IPC may also be referred to as inter-thread communication and inter-application communication. The combination of IPC with the address space concept is the foundation for address space independence/isolation.




Inter process communication (IPC).

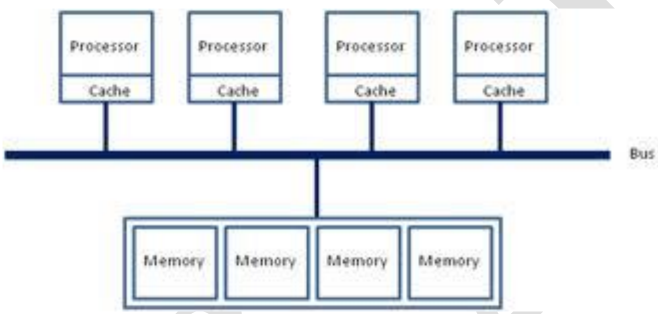
Cooperating processes need to exchange information, as well as synchronize with each other, to perform their collective task. The primitives discussed earlier can be used to synchronize the operation of cooperating processes, but they do not convey information between processes. Methods for effective sharing of information among cooperating processes are collectively known as inter process communication (IPC).

There has to be some underlying physical interconnection system to support the communication among the processors. Some basic interconnection schemes that have been used are:

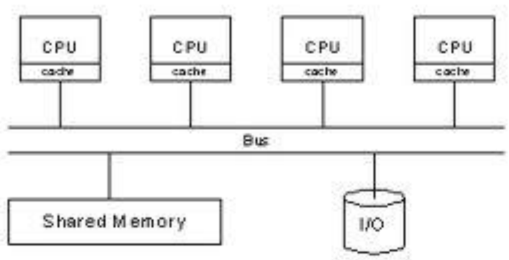
1) Timeshared or common buses
2) Crossbar switch
3) Multiport memory systems
4) Multistage networks

**1) PARAMETER PASSING SEMANTICS - Timeshared or common buses**

In this arrangement a single cable with enough lines to convey data and control bits acts as a passive channel to which all of the processors, I/O devices and memory modules are connected. The interface hardware between the bus and the functional units controls the data transmission across the bus. With this single bus system only one unit can use the medium at a time.



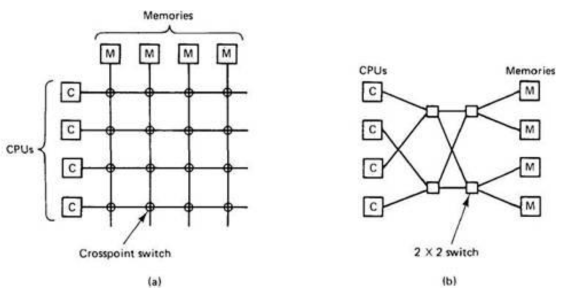

Time-shared or common bus multiprocessor arrangement




A multi bus, multiprocessor arrangement

**2) Crossbar switch**

The number of buses may be increased to permit a separate path to each memory module as in the below figure. This arrangement is called a crossbar switch.

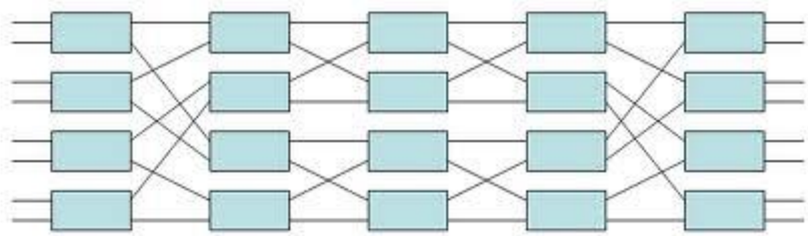



**3) Multiport memory system**

Another multiple bus arrangement can be employed to allow processor access to the memory modules at a specific entry point called a port.

**4) Multistage networks**

The multistage network links multiple switches as nodes in a tree like arrangement. The cost of those multistage networks which connects n processors to n memory modules grows as n log n. This differs somewhat considerably from that of the crossbar switch whose cost grows as n2.



A multistage network connecting eight processors to eight memory modules.

**CALL SEMANTICS - MASSIVELY PARALLEL ARCHITECTURE**

**ASSOCIATIVE PROCESSOR**

Associative memory encompasses a wide variety of phenomena related to human memory performance that it is closely related to the semantic representation of knowledge in relational structures . He describes two types of associative memory. First, direct association, the most common usage of the term, refers to the recall of one pattern by the input of a cue pattern. Direct association provide single input-to-output mapping based on similarity of content, physical, temporal, or logical relations, and typically deals with ordered sets of attributes.

Second, indirect association involves inference via multiple intermediate associative mappings. In indirect association, structural relationships are an important part of the patterns. In addition to mappings and inference, pattern completion from a partial or erroneous input is an important feature of associative memory. More importantly, most associative memory models include learning the cue-to-recall mappings. Content-addressable memory (CAM) is a physical embodiment of basic associative memory in which data is accessed by its content rather than by an address as in conventional computer memory.

- In RPC the caller and callee processes can be situated on different nodes. The normal functioning of an RPC may get disrupted due to one or more reasons mentioned below:

i. Call message is lost or response message is lost

ii. The callee node crashes and is restarted

iii. The caller node crashes and is restarted.

- In RPC system the call semantics determines how often the remote procedure may be executed under fault conditions. The different types of RPC call semantics are as follows:

**a. May-Be Call Semantics**

- This is the weakest semantics in which a timeout mechanism is used that prevents the caller from waiting indefinitely for a response from the callee.
- This means that the caller waits until a pre-determined timeout period and then continues to execute.
- Hence this semantics does not guarantee the receipt of call message nor the execution. This semantics is applicable where the response message is less important and applications that operate within a local network with successful transmission of messages.

**b. Last-Once Call Semantics**

- This call semantics uses the idea of retransmitting the call message based on timeouts until the caller receives a response.
- The call, execution and result of will keep repeating until the result of procedure execution is received by the caller.
- The results of the last executed call are used by the caller, hence it known as last-one semantics.
- Last one semantics can be easily achieved only when two nodes are involved in the RPC, but it is tricky to implement it for nested RPCs and cases by orphan calls.

**c. Last-of-Many Call Semantics**

- This semantics neglects orphan calls unlike last-once call semantics. Orphan call is one whose caller has expired due to node crash.
- To identify each call, unique call identifiers are used which to neglect orphan calls.
- When a call is repeated, it is assigned to a new call identifier and each response message has a corresponding call identifier.
- A response is accepted only if the call identifier associated with it matches the identifier of the most recent call else it is ignored.

**d. At-Least-Once Call Semantics**

- This semantics guarantees that the call is executed one or more times but does not specify which results are returned to the caller.
- It can be implemented using timeout based retransmission without considering the orphan calls.

**e. Exactly-Once Call Semantics**

- This is the strongest and the most desirable call semantics. It eliminates the possibility of a procedure being executed more than once irrespective of the number of retransmitted call.
- The implementation of exactly-once call semantics is based on the use of timeouts, retransmission, call identifiers with the same identifier for repeated calls and a reply cache associated with the callee.

## PART-B(6 MARKS)
## POSSIBLE QUESTIONS

1. Explain in detail about the Call Semantics
2. Explain Fundamentals of DOS explain with an example.
3. Describe about message passing in detail.
4. Explain Implementation of RPC mechanism.
5. Describe about RPC Models with an neat diagram.
6. Explain about RPC Message and Stub generations.

## PART-C(10 MARKS)
## POSSIBLE QUESTIONS

1. Explain in detail about RPC with an Real Time Example.
2. Explain in detail about Server management with example.
3. Describe about marshaling arguments and results.

**KARPAGAM ACADEMY OF HIGHER EDUCATION**

**Department of Computer Science**

**II M.Sc( CS) (BATCH 2017-2019)**

**Distributed Operating Systems**

**PART-A OBJECTIVE TYPE/**

| ONLINE EXAMINATIO | ONE MARK QUESTIONS | | | | |
|---|---|---|---|---|---|
| | | | | | |
| **QUESTIONS** | **opt1** | **opt2** | **opt3** | **opt4** | **ANSWER** |
| ____________means that a semantics of a rpc are identical to those of a local procedural call. | syntax | semantics | syntactic | systemetic | semantics |
| _______in these systems,the processors do not share memory,and each processor has its own local memory. | distributed computing | parallel processing system | tightly coupled systems | loosely coupled systems | loosely coupled systems |
| tightly coupled systems are referred to as ________. | loosely coupled systems | distributed computing | parallel processing system | tightly coupled systems | parallel processing system |
| loosely coupled systems are referred to as __________. | parallel processing system | local memory | shared memory | distributed computing systems | distributed computing systems |
| .A particular processor ,its own resources are_________ | remote | local | node | site | local |
| The other processor and their resources are _______. | node | site | remote | local | remote |
| A processor and its resources are usually referred to as a_______ or_____or______ | tcp | memory | machine | syntax | machine |
| ________with the use of control cards to define the beginning and end of a job. | automatic job sequencing | time sharing | offline processing | multiprogramming | automatic job sequencing |
| _______allowing CPU utilization by allowing overlap of CPU and I/O operations. | offline processing | multiprogramming | automatic job sequencing | time sharing | offline processing |
| ______improves CPU utilization by organizing jobs. | automatic job sequencing | time sharing | multiprogramming | offline processing | multiprogramming |

| | | | | | |
|---|---|---|---|---|---|
| The_________model is a simple extension of the centralized time sharing system. | minicomputer | macrocomputer | supercomputer | ARPANET | minicomputer |
| .The ___________model may be used when resource sharing with remote users desired. | ARPANET | supercomputer | minicomputer | macrocomputer | minicomputer |
| The _______is an example of distributed computing system based on the minicomputer model.B29 | ARPAnet | TELNET | TCP | UDP | ARPAnet |
| A distributed computing is based on the ___________model. | workstation | workstation server | processor pool | hybrid | workstation |
| .The first approach is to _________the remote process share the resources of the workstation. | allow | kill | migrate | destroy | allow |
| .The second approach is to _________ the remote process | kill | allow | destroy | migrate | kill |
| The third approach is to ________the remote process back to its workstation. | migrate | kill | allow | destroy | migrate |
| A workstation with its own local disk is usually called _______. | diskless | servermodel | diskful | clientmodel | diskful |
| A workstation without a local disk is called _______. | clientmodel | diskless | servermodel | diskful | diskless |
| A distributed compuing system based on________model. | workstation server model | workstation client | minicomputer | supercomputer | workstation server model |
| The _____is an example of a distributed computing system that is based on the workstation server model. | print server | database server | V-system | file server | V-system |

| | | | | | |
|---|---|---|---|---|---|
| The __________is based on the observation that most of the time a user does not need any computing power but once in a while he or she may need a vary large amount of computing power for a short time. | processor model | processor pool model | workstation servermodel | server model | processor pool model |
| A special server called_____ server manages and allocates the processors in a pool to different users on a demand basis. | file | remote | run | client | run |
| The ______model is based on the workstation server model but with the addition of a pool of processors. | workstation | workstation server model | processor pool model | hybrid | hybrid |
| The use of distributed computing system by a group of users to work comparatively is known as __________. | CSCW | remote | sharing | run | cscw |
| expand CSCW | computer supported cooperative working | computer system connection work | connection support coordinate working | communication support cooperative working | computer supported cooperative working |
| _____information is not the  only thing that can be shared in a distributed computing system. | resource pooling | sharing distributed user | resource sharing | distributed computing systems | resource sharing |
| _______refers to the degree of tolerance against errors and component failures in a sytem. | availability | stability | reliability | scalibility | reliability |

| | | | | | |
|---|---|---|---|---|---|
| Distributed computing system that have the property of extensibility and incremental growth are called __________. | distributed computing | distributed system | open distributed system | distributed growth | open distributed system |
| ______is very attractive feature because for most existing and proposed application. | incremental growth | decremental growth | increase | increasibility | incremental growth |
| _________is also easier in a distributed computing system because of new resources in existing sytem. | reliability | availability | scalability | extensibility | extensibility |
| The collection of networked machine act as a________ | uniprocessor | virtual | virtual uniprocessor | multiproces sor | virtual uniprocessor |
| _______commands for moving a file from one machine to other machine. | file transfer | file move | file navigation | copy file | file transfer |
| distributed computing syatem can be broadly classified into ______types. | 1.00 | 2.00 | 3.00 | 4.00 | 2.00 |
| The set of of system calls that an operating system supports are implemented by a set of programs called the____. | boot | kernel | dos | reboot | kernel |
| The distributed computing system that uses a network operating system is a usually referred to as______ | network architecture | network | network system | internetwor k | network system |
| Distributed operating system is usually referred to as a _____________- | false distributed system | true distributed system | distributed system | true/false distributed system | true distributed system |

| | | | | | |
|---|---|---|---|---|---|
| The main goal of distributed operating syatem is to make the multiple computers_____and provide a single syatem image to its user. | visible | lactency | transparency | decrease | visble |
| There are _____types of transparency. | 6.00 | 7.00 | 8.00 | 9.00 | 8.00 |
| ________means that users should not need or able to recognize whether resources is remote or local. | access transparency | location transparency | replication transparency | failure transparenc y | access transparency |
| _______refers to the fact that the name of a resource should not reveal any hint to as a physical location. | user mobility | replication transparency | name transparency | access transparenc y | name transparency |
| _________refers to the fact that no matter which machine a user logged onto he/she should be able to access the resources with the same name. | failure transparency | username | name server | user mobility | user mobility |
| _____________system should be simple and easy to use. | message passing | IPC | RPC | RPC and IPC | message passing |
| ______in which the communicating processes are on the same node. | local communicatio n | remote communicatio n | global communicati on | local and remote | local communication |
| _______in which the communicating processes are on the different node. | global communicatio n | local communicatio n | local and remote | remote communicat ion | remote communication |
| An IPC protocol of message passing system can be made of a_________by reducing the number of a message. | efficiency | reliability | correctness | flexibility | efficiency |

| | | | | | |
|---|---|---|---|---|---|
| A ________IPC protocol can cope with failure problems and guarentees the delivery of a message. | reliability | efficiency | flexibility | correctness | reliability |
| lost messages usually involves __________and ___________on the basis of timeouts. | acknowledge ments and retransmission | ack and non ack | ack and redundancy | ack and timeout | acknowledgements and retransmission |
| ______message may be sent in the event of failures or because of timeouts. | duplicate | replicate | redundancy | correctness | duplicate |
| The issues related to correctness are_____ | atomicity | ordered | survival | flexible | atomocity |
| ______ensures that every message sent to a group of receivers will be delivered to either all of them. | flexible | ordered delivery | atom | orded | ordered delivery |
| _____ensures that messages arrive at all receivers in an order to acceptable to the application. | atomicity | ordered delivery | survivability | correctness | ordered delivery |
| _____guarentees that messages will be delivered correctly. | survivability | atomicity | ordered delivery | correctness | survivability |
| A good message passing system must be also capable of providing a ________end -end communication. | security | potability | flexibility | reliability | security |
| There are _________aspects of portability in a message passing system. | 1.00 | 2.00 | 3.00 | 4.00 | 2.00 |
| The message passing system should itself be______ | portable | reliable | scalable | flexible | portable |
| ________uniquely identify the sending and receiving processes in the network | sequence number | address | structural information | name space | address |

| | | | | | |
|---|---|---|---|---|---|
| ____is the message identifier which is used to identify the lost messages. | structural information | sequence number | name space | address | sequence number |
| The _______commonly known as client process. | caller | callee | sender | receiver | caller |
| The ____commonly known as server process. | caller | callee | receiver | sender | callee |
| _________means that a rpc should have exactly the same syntax as a local procedural call | syntax | syntactic | semantic | systemetic | syntactic |
| The client is a user process that initiates a________ | IPC | RPC | client stub | serverstub | RPC |
| The _______handles transmission of messages across network between client and server machines. | RPC | IPC | IPSEC | RPCRuntime | RPCRuntime |
| ________that are sent by the client to the server for requesting execution of a particular remote procedure. | replymessage | callmessage | requestmessage | callee message | callmessage |
| ________that are sent by the server to the client for returning the result of remote procedure execution. | replymessage | callmessage | requestmessage | callee message | replymessage |
| A _______server maintains cliuents state information from one remote procedure call to the next. | stateless | stateful | connection oriented | connectionless | stateful |
| ________operation is used to open a file identified by filename in a specified mode. | filename,mode | fid,n,buffer | fid,buffer | file,position | filename,mode |
| _____causes the server to delete from its file table state information | close(fid) | open | open(fid) | close | close(fid) |

| | | | | | |
|---|---|---|---|---|---|
| A______server does not maintain any client state information. | stateful | stateless | call server | stub server | stateless |
| In __________method all parameters are copied into a message that is transmitted from the client to the server through the intervening network. | call by value | call by reference | caller | callee message | call by value |
| In _______a parameter is passed by reference as in the method of call by object | call by value | call by reference | call by move | call by object | call by move |

# UNIT II

## SYLLABUS

Distributed shared memory : Introduction – general architecture of DSM systems – design and implementation of DSM – granularity – structure of shared memory space – replacement strategy – heterogeneous DSM – advantages of DSM.

## DISTRIBUTED SHARED MEMORY INTRODUCTION :

The concepts of interconnecting computers and sharing information through them were introduced; advancements have been made in almost every aspect of human life. These advancements came merely due to extensive study and exploration of fundamentals related to operations and functions of computers and network systems. This study of networking and computers has always remained at the heart of communication sciences and is still taught widely all around the world. The theory of network systems and computers only involves some fundamentals to be understood; based on these fundamentals, the entire global communication scenario is functioning and operating.

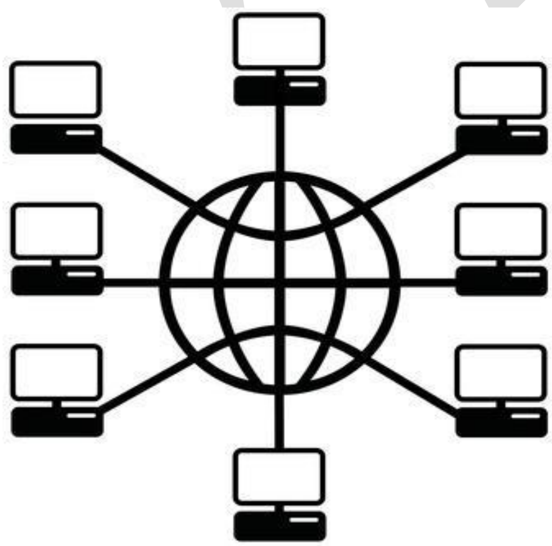


**Fig: The entire global communication scenario is functioning and operating.**

**GENERAL ARCHITECTURE OF DSM SYSTEMS ::**

Modulation techniques are employed to allow the digital signal to be carried on the analog channel. A device is needed to convert the signal from digital to analog at the sending end and to convert from analog to digital at the receiving end. This device is called modem (modulator/demodulator). Three common modulation technique used are

1) Frequency shift keying (FSK)
2) Phase Modulation
3) Amplitude Modulation

Demodulation involves the opposite operation. For the binary signal only two frequency values are needed. One constraint is that, in order to detect the frequency, at least half a cycle must be transmitted. Therefore the time interval, I s, between the changes in the value the signal must be greater than or equal to the time to complete half of the cycle, i.e. half the period, T, of the wave.

Since the lowest frequency used has the longest period, then

$$1 \geq (1/2)T$$

Where T is the period of the lowest frequency, f, That is output by the modulator. Since the number of signal changes per second gives the baud rate, b, then

$$B=1/I$$

$$\text{i.e. } 1/b \geq (1/2)(1/f) \text{ therefore}$$

$$f \geq (1/2)\, b$$

hence the lowest frequency used be greater than or equal to half of the baud rate of the data signal.

**Phase Modulation (PM)**

In PM the signal is coded in phase changes. In a phase change the wave retains its shape but there is a shift in its position. Therefore the same frequency is used but, by dedicating distinct phase changes to particular digital values, the signal can be transmitted.

Large phase changes are used to facilitate detection. At the start of the signaling interval there is a test to determine the extent of change relative to the state in the previous interval. Differential phase modulation allows four possible phase changes: $0^0$,$90^0$,$180^0$and $270^0$. With four changes, four distinct values can be coded. Therefore 2 bits of information are transmitted in each phase change.

**Amplitude modulation (AM)**

**Amplitude modulation** (**AM**) is a modulation technique used in electronic communication, most commonly for transmitting information via a radio carrier wave. AM works by varying the strength (amplitude) of the transmitted signal in relation to the information being sent. For example, changes in signal strength may be used to specify the sounds to be reproduced by a loudspeaker, or the light intensity of television pixels. This contrasts with frequency modulation, in which the frequency of the carrier signal is varied, and phase modulation, in which the phase is varied, by the modulating signal.

**DESIGN AND IMPLEMENTATION OF DSM :**

- Produces fewer errors
  - Easier to detect and correct errors, since transmitted data is binary (1s and 0s, only two distinct values))
- Permits higher maximum transmission rates
  - e.g., Optical fiber designed for digital transmission
- More efficient
  - Possible to send more digital data through a given circuit
- More secure
  - Easier to encrypt
- Simpler to integrate voice, video and data

- Easier to combine them on the same circuit, since signals made up of digital data

- Issues
  - How to keep track of the location of remote data
  - How to minimize communication overhead when accessing remote data
  - How to access concurrently remote data at several nodes
  -

1. The Central Server Algorithm
   - Central server maintains all shared data
     - Read request: returns data item
     - Write request: updates data and returns acknowledgement message
   - Implementation
     - A timeout is used to resend a request if acknowledgment fails
     - Associated sequence numbers can be used to detect duplicate write requests
     - If an application's request to access shared data fails repeatedly, a failure condition is sent to the application
   - Issues: performance and reliability
   - Possible solutions
     - Partition shared data between several servers
     - Use a mapping function to distribute/locate data
2. The Migration Algorithm
   - Operation
     - Ship (migrate) entire data object (page, block) containing data item to requesting location
     - Allow only one node to access a shared data at a time
   - Advantages
     - Takes advantage of the locality of reference
     - DSM can be integrated with VM at each node

- Make DSM page multiple of VM page size
- A locally held shared memory can be mapped into the VM page address space
- If page not local, fault-handler migrates page and removes it from address space at remote node

- To locate a remote data object:
  - Use a location server
  - Maintain hints at each node
  - Broadcast query
- Issues
  - Only one node can access a data object at a time
  - Thrashing can occur: to minimize it, set minimum time data object resides at a node.

3. The Read-Replication Algorithm

- Replicates data objects to multiple nodes
- DSM keeps track of location of data objects
- Multiple nodes can have read access or one node write access (multiple readers-one writer protocol)
- After a write, all copies are invalidated or updated
- DSM has to keep track of locations of all copies of data objects. Examples of implementations:
  - IVY: owner node of data object knows all nodes that have copies
  - PLUS: distributed linked-list tracks all nodes that have copies
- Advantage
  - The read-replication can lead to substantial performance improvements if the ratio of reads to writes is large

4. The Full–Replication Algorithm

- Extension of read-replication algorithm: multiple nodes can read and multiple nodes can write (multiple-readers, multiple-writers protocol)

- Issue: consistency of data for multiple writers
- Solution: use of gap-free sequencer
  - All writes sent to sequencer
  - Sequencer assigns sequence number and sends write request to all sites that have copies
  - Each node performs writes according to sequence numbers
  - A gap in sequence numbers indicates a missing write request: node asks for retransmission of missing write requests

**GRANULARITY**

An **analog-to-digital converter** (abbreviated **ADC**, **A/D** or **A to D**) is a device that converts a continuous physical quantity (usually voltage) to a digital number that represents the quantity's amplitude.

The conversion involves quantization of the input, so it necessarily introduces a small amount of error. Instead of doing a single conversion, an ADC often performs the conversions ("samples" the input) periodically. The result is a sequence of digital values that have converted a continuous-time and continuous-amplitude analog signal to a discrete and discrete-amplitude digital signal.

**STRUCTURE OF SHARED MENORY SPACE :**

It is a method used to digitally represent sampled analog signals. It is the standard form of audio in computers, Compact Discs, digital telephony and other digital audio applications. In a PCM stream, the amplitude of the analog signal is sampled regularly at uniform intervals, and each sample is quantized to the nearest value within a range of digital steps.

PCM streams have two basic properties that determine their fidelity to the original analog signal: the sampling rate, the number of times per second that samples are taken; and the bit depth, which determines the number of possible digital values that each sample can take.

Structure defines the abstract view of the shared memory space.

The structure and granularity of a DSM system are closely related three approaches:

- No structuring
- Structuring by data type
- Structuring as a database

**1. NO SRTUCTURING:-**

Ø The shared memory space is simply a linear array of words.

**ADVANTAGE:-**

Ø Choose any suitable page size as the unit of sharing and a fixed grain size may be used for all application.

Ø Simple and easy to design such a DSM system.

**2. STRUCTURING BY DATA TYPE:-**

Ø The shared memory space is structured either as a collection of variables in the source language.

Ø The granularity in such DSM system is an object or a variable.

Ø DSM system use variable grain size to match the size of the object/variable being accessed by the application.

**3. STRUCTURING AS A DATABASE:-**

Ø Structure the shared memory like a database.

Ø Shared memory space is ordered as an associative memory called tuple space.

Ø To perform update old data item in the DSM are replaced by new data item.

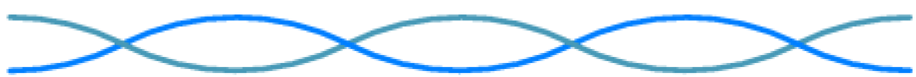
Ø Processes select tuples by specifying the number of their fields and their values or type.

Ø Access to shared data is non transparent. Most system they are transparent.

**REPLACEMENT STRATEGY :**

**COPPER WIRES**

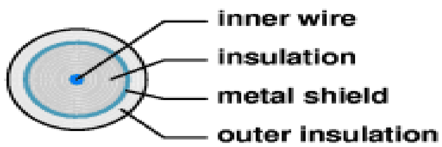
Conventional computer networks use copper wire because it is inexpensive, easy to install, and has low resistance to electrical current. Unfortunately, copper wire is prone to interference in the form electromagnetic energy emitted by neighbouring wires, especially those running in parallel. To minimise interference, twisted pair wiring, as used in telephone systems, can be used as illustrated in Figure.



**Twisted pair wiring**

A plastic coating on each wire prevents the copper in one wire from touching the copper in another. The twist helps reduce interference by preventing electrical signals on the wire radiating energy (causing interference) and by preventing signals on other wires interfering with the pair.

A second type of copper wire is coaxial cable, similar to that used for TV aerials. The coaxial cable provides better protection from interference by providing a metal shield as illustrated in Figure.




**Cross-section of a coaxial cable**

The metal shield forms a flexible cylinder around the inner wire providing a barrier to electromagnetic radiation, both incoming and outgoing. The cable can run parallel to other cables and can be bent round corners.

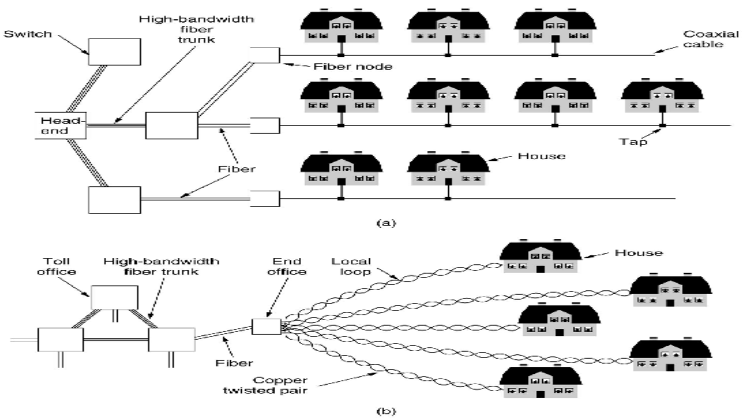
**HETEROGENEOUS DSM :**

Optical fibres use light to transmit data. A thin glass fibre is encased in a plastic jacket which allows the fibre to bend without breaking. A transmitter at one end uses a light emitting diode (LED) or laser to send pulses of light down the fibre which are detected at the other end by a light sensitive transistor. Figure illustrates a single fibre (a) and a sheath of three fibres (b). Other configurations are possible.

Optical fibres have four main advantages over copper wires.

- They use light which neither causes electrical interference nor are they susceptible to electrical interference
- They are manufactured to reflect the light inwards, so a fibre can carry a pulse of light further than a copper wire can carry a signal

- Light can encode more information that electrical signals, so they carry more information than a wire
- Light can carry a signal over a single fibre, unlike electricity which requires a pair of wires




**Cable television and POTS**

Figure illustrates the hybrid nature of neighbourhood wiring. Optical fibres carry cable TV to each street with the houses fed by coaxial cable (a). Optical fibres also carry the Plain Old Telephone Service (POTS) to the nearest exchange, with the local loop to the house consisting of twisted pairs (b).

The design, implementation, and performance of heterogeneous distributed shared memory (HDSM) are studied. A prototype HDSM system that integrates very different types of hosts has been developed, and a number of applications of this system are reported. Experience shows that despite a number of difficulties in data conversion, HDSM is implementable with minimal loss in functional and performance transparency when compared to homogeneous DSM systems

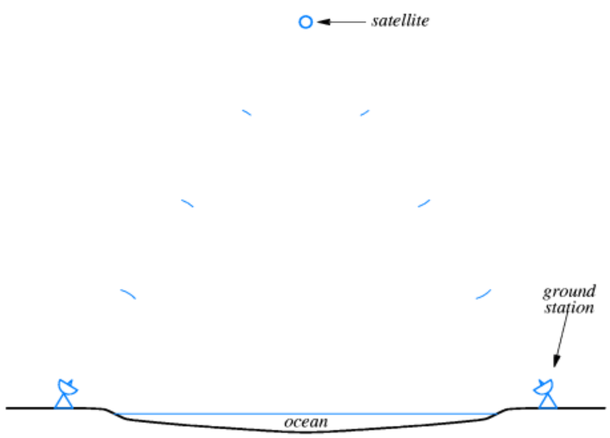
**ADVANTAGES OF DSM**

**RADIO**

A network that uses electromagnetic radio waves operates at radio frequency and its transmissions are called RF transmissions. Each host on the network attaches to an antenna, which can both send and receive RF.

**SATELLITES**

Radio transmissions do not bend round the surface of the earth, but RF technology combined with satellites can provide long-distance connections. Figure illustrates a satellite link across an ocean.




**Satellite and ground stations**

The satellite contains a transponder consisting of a radio receiver and transmitter. A ground station on one side of the ocean sends a signal to the satellite, which amplifies it and transmits the amplified signal at a different angle than it arrived at to another ground station on the other side of the ocean. A single satellite contains multiple transponders (usually six to twelve) each using a different radio frequency, making it possible for multiple communications to proceed simultaneously. These satellites are often geostationary, i.e. they appear stationary in the sky. To achieve this, their orbit must be 22,236 miles (35,785 kilometres) high.

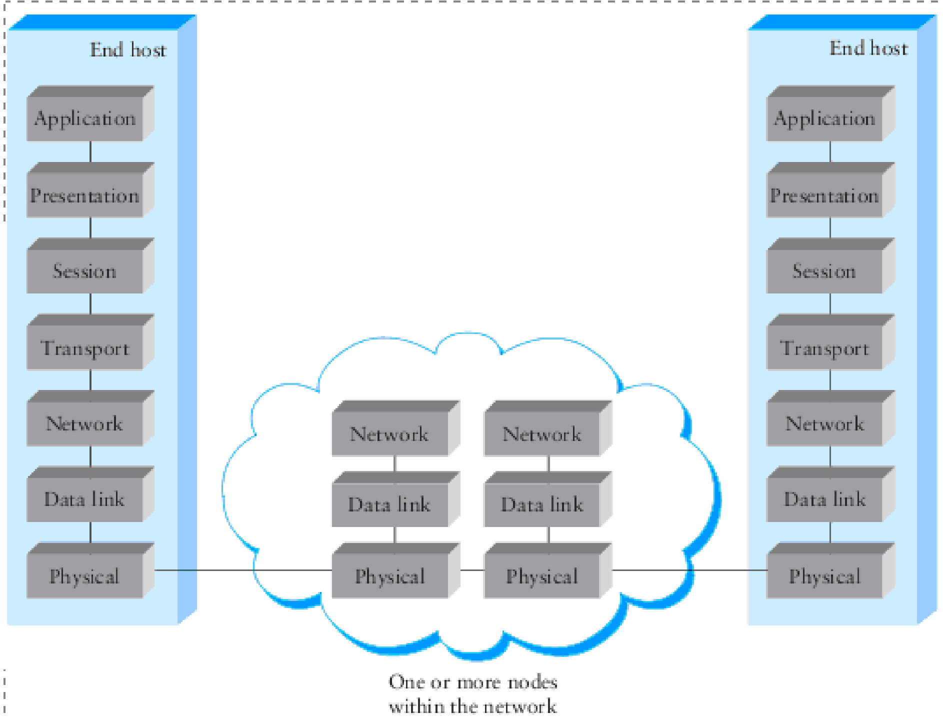
**MICROWAVE**

Electromagnetic radiation beyond the frequency range of radio and television can be used to transport information. Microwave transmission is usually point-to-point using directional antennae with a clear path between transmitter and receiver.

**INFRARED**

Infrared transmission is usually limited to a small area, e.g. one room, with the transmitter pointed towards the receiver. The hardware is inexpensive and does not require an antenna.

**Network architecture** is the design of a communications network. It is a framework for the specification of a network's physical components and their functional organization and configuration, its operational principles and procedures, as well as data formats used in its operation. In telecommunication, the specification of a network architecture may also include a detailed description of products and services delivered via a communications network, as well as detailed rate and billing structures under which services are compensated. The network architecture of the Internet is predominantly expressed by its use of the Internet Protocol Suite, rather than a specific model for interconnecting networks or nodes in the network, or the usage of specific types of hardware links.




**OSI network architecture.**

## PART-B(6 MARKS)
## POSSIBLE QUESTIONS

1. What is Thrashing? And explain detail about Thrashing concepts.
2. Explain in detail about Structure of Shared Memory Space.
3. Briefly describe about Design and Implementation Issue in DSM
4. What is DSM? And explain in detail about Advantages of DSM.
5. Explain about Heterogeneous DSM in detail.
6. What is Message Passing System? Explain factors influencing block size selection
7. Explain about the Memory Coherence and about page size as block size in granularity.
8. Explain Block Size Selection and other approaches to DSM.

## PART-C(10 MARKS)
## POSSIBLE QUESTIONS

1. Compare Memory and Distributed Shared Memory with an live example.
2. Explain about the general architecture of DSM System with neat sketch.
3. What do you mean by Replacement Strategy? And explain in detail about following issues in Replacement Strategy.

**KARPAGAM ACADEMY OF HIGHER EDUCATION**

**Department of Computer Science**

**II M.Sc( CS) (BATCH 2017-2019)**

**Distributed Operating Systems**

**PART-A OBJECTIVE TYPE/ MULTIPLE CHOICE QUESTIONS**

| ONLINE EXAMINATIONS | ONE MARK QUESTION | | | | |
|---|---|---|---|---|---|
| | | | | | |
| **QUESTIONS** | **opt1** | **opt2** | **opt3** | **opt4** | **ANSWER** |
| Message-Passing systems supporting__________ | RPCs | DSVM | Address memory | Network | RPCs |
| Process use this adress space in the same way the use normal ------------- memory | Main | Virtual | Local | Shared | Local |
| MIMD Meaning ------------------------- | Multiple-Internet,Memory-Data-Stream | Multiple-instruction,Multiple-Data-Stream | Multiple-Intrupt,Multiple-Data-Stream | MemoryIntruptMessageDataAccess | Multiple-instruction,Multiple-Data-Stream |
| A36 | Local | Memory | Size | Virtual | Virtual |
| DSVM meanS ----- | Distributed SizeVariableMemory | DistributedSharedVirtualMemory | DataSharedVirtualMemory | DataStructureVirtualMemory | DistributedSharedVirtualMemory |
| The DSM abstraction presents ------- shated-memory in space to the processors of all nodes | Large | Medium | Low | high | Large |
| Data caching is a well-known solution to -------------------- access latency | SizeOfMemory | AddressMemory | SizeMemory | VirtualMemory | AddressMemory |
| data caching is used in DSM system to reduce ------------------ latency | Structure | Network | Thrashing | Address | Network |

| | | | | | |
|---|---|---|---|---|---|
| The ------------ of individual nodes is used to cache pieces of the shared-memory space | Memory | Size | Local | Mainmemory | Mainmemory |
| The basic unit of caching is a ------- block | Virtual | Memory | Thrashing | Granularity | Memory |
| DSM system allows replication and / or migration of ------------- data blocks | Shared-Size | Shared-Data | Shared-memory | Shared-Address | Shared-memory |
| What is the block of size in DSM system ---------- | Granularity | Thrashing | Homogeneous | Replacement strategy | Granularity |
| It -------- refers to the layout of the shared data in memory. | Network | Size | Thrashing | Structure | Structure |
| A data block of the ----------- memory must be replaced by the new data block | Virtual | Local | Address | Structure | Local |
| Data blocks migrate between nodes on demand is __________ | Homogeneous | Structure | Networking | Thrashing | Thrashing |
| The DSM systems built for ------------ systems | Thrashing | Homogeneous | Granularity | Heterogeneity | Homogeneous |
| several criteria for choosing this -------- parameter are described below. | Granularity | Homogeneous | Thrashing | Heterogeneity | Granularity |
| The ---------- is the large block size was less the small block sizes | Paging oversize | Paging overwrite | Paging overhide | Paging overhead | Paging overhead |
| The same data block are being updated by multiple nodes at tge same time ------- accur or design | Homogeneous | Heterogeneity | Thrashing | Granularity | Thrashing |

| | | | | | |
|---|---|---|---|---|---|
| Two different processes access two unrelated variable in same data block it ------------ occuring | Memory sharing | Network sharing | Virtual sharing | False sharing | False sharing |
| Memory  coherence problems can be resolved in ------- handlers | Page-fault | page-size | page-address | Both a&b | Page-fault |
| Structure of shared-memory space is commonly using ------- types | Two | One | Three | Five | Three |
| DSM systens do not structure their ------ space | Shared-Size | Shared-Data | Shared-address | Shared-memory | Shared-memory |
| The ---------- space is simply a linear array of words | Shared-memory | Shared-address | Shared-Data | Shared-size | Shared-memory |
| LRU meaning ---------. | Large recently used | Least recently used | Local recently used | Local record used | Least recently used |
| The --------- algorithms are not suitable for a DSM system | Virtual-Space | Virtual-System | Variable-System | Variable-Space | Variable-Space |
| Both --------&--------- blocked have the replacement priority | Unused & nil | used & nil | Unaccessd & nil | Acessed & nil | unused & nil |
| A ------- is the no longer useful and future access to the block | Block | Unblock | Both a&b | Nil block | Nil block |
| The DSM systems must be designed to take care of ----------------- | Thrashing | Heterogeneit y | Granularity | Shared-memory | Heterogeneity |
| The shared memoty of DSM exists only --------------- | Virtually | Manually | Automatically | configuiring | Virtually |
| Most DSM system do not __________their shared memory. | no structuring | structuring | structuring by datatype | structuring by database | no structuring |

| | | | | | |
|---|---|---|---|---|---|
| The granularity in such DSM system is_____or a _________. | class or object | object or method | object or a variable | variable or a method | object or a variable |
| Another method is to structure the shared mememory like a___________ | objects | database | structure | unstructure | database |
| A set primitives that can be added to any base languages are _____ and____ | unix and linux | c and c++ | c and FORTRAN | java and c++ | c and FORTRAN |
| A shared memory space is ordered as an associative memory called_______. | tables | record | objects | tablespace | tablespace |
| A_______model is basically refers to the degree of consistency that has to be maintained for the shared memory data. | inconsistency | consistency | regular | automatic | consistency |
| The _______model is the strongest form of memory coherance havimg the most stringent consistency requirement. | strict consistency | weak consistency | strong consistency | no consistency | strict consistency |
| The _______model was proposed by Lamport. | strict consistency | sequential consistency | weak consistency | strong consistency | sequential consistency |
| The _______model was proposed by Hutto and Ahamad. | casual consistency | weak consistency | strong consistency | no consistency | casual consistency |
| A________model is simple and easy to implement and also has good performance | PRAM | PROM | RAM | PREM | PRAM |
| Expand PRAM_____ | pipelined read only memory | pipe read only memory | potential random only memory | pipelined random access memory | pipelined random access memory |

| | | | | | |
|---|---|---|---|---|---|
| _________means that for any memory location all processes agree on the same order of all write operation to that location. | memory space | memory coherance | shared memory | shared coherance | memory coherance |
| All accesses to synchronization variable must obey sequential consistency semantics. | weak consistency | strong consistency | PRAM | processor | weak consistency |
| All changes made to the memory by the process are propagated to other nodes. | release consistency | weak consistency | strong consistency | processor | release consistency |
| Nonreplicated, nonmigrating blocks________ | NRNB | NRNMB | NRNRM | RAMB | NRNMB |
| Nonreplicated,migrating blocks___________ | NRNMB | NRMB | RMB | RNMB | NRMB |
| Replicated.migrating blocks | RMB | RMO | RMS | MRS | RMB |
| Replicated,nonmigrating blocks____________ | RMB | RNMB | RNB | RENB | RNMB |
| There is a single copy of each block in the entire system_______ | NRMB | NRNMB | RMB | RMNB | NRNMB |
| A______algorithm is used to centtralized server maintains a block table that contains the location information. | server | centralized server | decentralized server | client/server | centralized server |
| The fixed_________scheme is a direct extension of the centralized server scheme. | centralized server | distributed server | client/server | client machine | distributed server |
| _________copies of a piece of data except one are invalidated before a write can be performed on it. | write validate | write update | read request | write request | write validate |

| | | | | | |
|---|---|---|---|---|---|
| A write operation is carried out by updating all copies of data on which the write is performed. | write validate | write update | read request | write request | write update |
| If there is a local block containing the data and if it is valid,the request is satisfied by accessing the local copy of the data. | read request | write request | write validate | write update | read request |
| If there is a local block containing the data and if it is valid and writable,the request is immediately satisfied by accessing the local copy of data. | write request | write update | read request | write request | write request |
| Keeping track of the nodes that currently have a valid copy of the block. | RMB | PRAM | RAM | PROM | RMB |
| Shared data variables annotated as _____only are immutable data items. | write | read | migratory | write shared | read |
| ________shared variables that are accessed in phases,where each phase corresponds to a series of accesses by a single process. | migratory | write | read request | read | migratory |
| A free memory block that is not currently being used. | Unused | nil | read only | readowned | Unused |
| A block that has been invalidated. | unused | nil | read only | write only | nil |

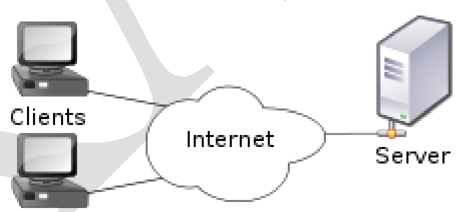
**UNIT III**

**SYLLABUS**

Synchronization: Introduction – clock synchronization – event ordering – mutual exclusion. Resource management: Introduction – desirable features of a good global scheduling algorithm – task management approach – load balancing approach – load sharing approach.

## SYNCHRONIZATION :

A network operating system (NOS) is a software program that controls other software and hardware that runs on a network. It also allows multiple computers, also known as network computers, to communicate with one main computer and each other, so as to share resources, run applications, and send messages, among other things. A computer network can consist of a wireless network, local area network (LAN), a wide area network (WAN), or even two or three computer networks. The heart of any of these networks, however, is the network operating system. There are different types of operating systems. Most individual computer users run client operating systems, like Windows XP, which run on a single computer. Personal computers that individuals use at home have a client operating system which manages the interactions and processes between the computer and its peripherals such as the keyboard, mouse, external monitor, and printer. In a sense, this is also a network, though it is different in scale than a network operating system which manages the interactions of many computers.




Schematic clients-server interaction.

The client/server characteristic describes the relationship of cooperating programs in an application. The server component provides a function or service to one or many clients, which initiate requests for such services.

## INTRODUCTION

Functions such as email exchange, web access and database access, are built on the client/server model. Users accessing banking services from their computer use a web browser client to send a request to a web server at a bank. That program may in turn forward the request to its own database client program that sends a request to a database server at another bank computer to retrieve the account information. The balance is returned to the bank database client, which in turn serves it back to the web browser client displaying the results to the user. The client–server model has become one of the central ideas of network computing. Many business applications being written today use the client–server model. So do the Internet's main application protocols, such as HTTP, SMTP, Telnet, and DNS.

The interaction between client and server is often described using sequence diagrams. The Unified Modeling Language has support for sequence diagrams. Specific types of clients include web browsers, email clients, and online chat clients. Specific types of servers include web servers, ftp servers, application servers, database servers, name servers, mail servers, file servers, print servers, and terminal servers. Most web services are also types of servers.

A network architecture in which each computer or process on the network is either a client or a server. Servers are powerful computers or processes dedicated to managing disk drives (file servers), printers (print servers), or network traffic (network servers ). Clients are PCs or workstations on which users run applications. Clients rely on servers for resources, such as files, devices, and even processing power.

## CLOCK SYCHRONIZATION

A **distributed operating system** is the logical aggregation of operating system software over a collection of independent, networked, communicating, and physically separate computational nodes. Individual nodes each hold a specific software subset of the global aggregate operating system. Each subset is a composite of two distinct services provisionary. The first is a ubiquitous minimal kernel, or microkernel, that directly controls that node's hardware. Second is a higher-level collection of system management components that coordinate the node's individual and collaborative activities. These components abstract microkernel functions and support user applications. The microkernel and the management components collection work together. They support the system's goal of integrating multiple resources and processing functionality into an efficient and stable system. This seamless integration of individual nodes into a global system is referred to as transparency, or single system image; describing the illusion provided to users of the global system's appearance as a single computational entity.

As in non-distributed systems, the knowledge of when events occur is necessary. However, clock synchronization is often more difficult in distributed systems because there is no ideal time source, and because distributed algorithms must sometimes be used.

Physical Clocks „The time difference between two computers is known as drift. Clock drift over time is known as skew. Computer clock manufacturers specify a maximum skew rate in their products. „Computer clocks are among the least accurate modern timepieces. „Inside every computer is a chip surrounding a quartz crystal oscillator to record time. These crystals cost 25 seconds to produce. „Average loss of accuracy: 0.86 seconds per day „This skew is unacceptable for distributed systems.

Physical Clocks - UTC Coordinated Universal Time (UTC) is the international time standard. UTC is the current term for what was commonly referred to as Greenwich Mean Time (GMT). Zero hours UTC is midnight in Greenwich, England, which lies on the zero longitudinal meridian. UTC is based on a 24-hour clock. „

Physical Clocks – Berkeley Algorithm „

One daemon without UTC: „Periodically, the daemon polls all machines on the distributed system for their times. „The machines answer. „The daemon computes an average time and broadcasts it to the machines so they can adjust.

Physical Clocks – Decentralized Averaging Algorithm „

Each machine on the distributed system has a daemon without UTC. „Periodically, at an agreed-upon fixed time, each machine broadcasts its local time. „Each machine calculates the correct time by averaging all results.

Physical Clocks – Network Time Protocol (NTP) „

Enables clients across the Internet to be synchronized accurately to UTC. „Overcomes large and variable message delays „

Employs statistical techniques for filtering, based on past quality of servers and several other measures „

Can survive lengthy losses of connectivity:

„Redundant servers „

Redundant paths to servers „

Provides protection against malicious interference through authentication techniques.

Uses a hierarchy of servers located across the Internet.

Primary servers are directly connected to a UTC time source. NTP has three modes:

„Multicast Mode: „Suitable for user workstations on a LAN „One or more servers periodically multicasts the time to other machines on the network. „

Procedure Call Mode: „Similar to Christian's Algorithm „Provides higher accuracy than Multicast Mode because delays are compensated.

Symmetric Mode: „Pairs of servers exchange pairs of timing messages that contain time stamps of recent message events. „The most accurate, but also the most expensive mode.

Logical Clocks

„Often, it is not necessary for a computer to know the exact time, only relative time. This is known as "logical time".

„Logical time is not based on timing but on the ordering of events.

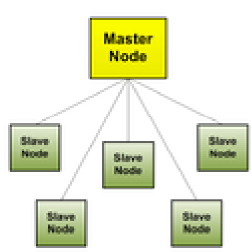
„Logical clocks can only advance forward, not in reverse.

„Non-interacting processes cannot share a logical clock.

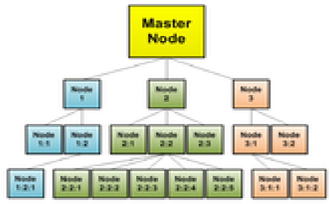
„Computers generally obtain logical time using interrupts to update a software clock. The more interrupts (the more frequently time is updated), the higher the overhead.

„Scattering of information. Local, rather than global, decision-making

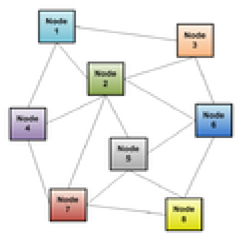
**EVENT ORDERING**




Generalized organization of nodes in a centralized model.




Generalized organization of nodes in a networked model.



**MUTUAL EXCLUSION**

Problems Unique to Distributed Systems

- Distributed Operating Systems:
    - Generation: Third Generation Operating System.
    - Characteristics: Global view of file system, name space, time, security, computational power.
    - Goal: Single computer view of multiple computer system (transparency)
- Distributed Operating System Goals:
    - Efficiency
    - Consistency
    - Robustness

Every node in the system keeps a request queue sorted by logical time stamp.

Logical clocks are used to impose total global order on all events.

Ordered message delivery between every pair of communicating sites is assumed.

1. Messages sent from Site arrive at Site in the same order.

Site Si sends a request and places the request in the local request queue.
2. When Site Sj receives the request, it sends a time-stamped reply to Site Si and places the request in its local request queue.

3. Site Si gains the critical section of the requested data when it has received a message from all other sites with a timestamp larger than the request.
4. Centralized Algorithm
5. The most simple and straightforward way to achieve mutual exclusion in a
6. distributed system is to simulate how it is done in a one-processor system:
7. One process is elected as the coordinator.
8. When any process wants to enter a critical section, it sends a request message to
9. The coordinator stating which critical section it wants to access.
10. If no other process is currently in that critical section, the coordinator sends back
11. A reply granting permission. When the reply arrives, the requesting process enters
12. The critical section. If another process requests access to the same critical section,
13. It is ignored or blocked until the first process exits the critical section and sends a
14. message to the coordinator stating that it has exited.

It is often unacceptable to have a single point of failure.

When a process wants to enter a critical section, it builds a message containing the name of the critical section, its process number, and the current time. It then sends the message to all other processes, as well as to itself.

Token-Based Algorithms

Another approach is to create a logical or physical ring.

Each process knows the identity of the process succeeding it.

When the ring is initialized, Process 0 is given a token. The token circulates around the ring in order, from Process k to Process k + 1.

When a process receives the token from its neighbor, it checks to see if it is attempting to enter a critical section. If so, the process enters the critical section and does its work, keeping the token the whole time.

After the process exits the critical section, it passes the token to the next process in the ring. It is not permitted to enter a second critical section using the same token. „If a process is handed a token an is not interested in entering a critical section, it passes the token to the next process.

**RESOURCE MANAGEMENT - INTRODUCTION**

- Objects models and identification.
- Distributed Coordination.

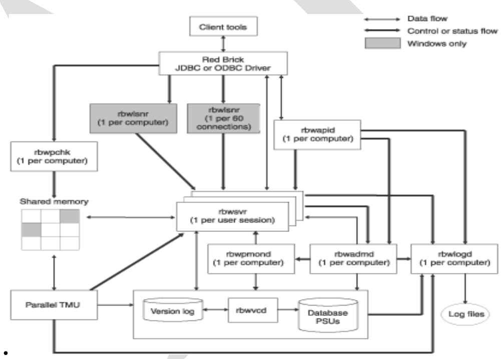
- Intercrosses Communication
- Distributed Resources.
- Fault Tolerance and Security.

**DESIRABLE FEATURES OF A GOOD GLOBAL SCHEDULING ALGORITHM :**

The term Inter-Process Communication (IPC) refers to a predefined library or set of interfaces that allow processes to communicate with each other. IPC gives the appearance of programs that run concurrently in an operating system's background and allows computer users to conduct multiple tasks at once on a computer. IPCs can share memory, run in synchrony with other processes, pass messages, and conduct remote procedure calls. The specific IPC method varies based on the Operating System (OS), latency of communication between program threads, and the type of information being exchanged between the processes.

Inter-Process Communication Methods

There are several ways to support Inter-Process Communications on an OS. These include: Message queuing – one or more message queues sends messages between running processes and the OS kernel manages them. Pipes – information can only be sent in one direction and is buffered until received




.

Named pipes – a pipe has a certain name and can be used among processes that do not share a common origin. Shared memory – permits information exchange through a predefined area of memory and has to be allocated before data can gain access to the memory location.

Semaphores – solves problems when synchronization or race conditions arise between processes.

Socket – processes use these to communicate over a network via a client/server relationship.

**TASK MANAGEMENT APPROACH**

A common Inter-Process Communications problem is that when one or more resources cannot be shared, they are mutually exclusive and may result in a waste of system resources or processor time. Basic inter-processes that help prevent this from blocking Inter-Process Communication include: 1) sleep and wake up conditions that require a caller to wake a process up when it has enough resources to work or is asleep otherwise, 2) the producer-consumer issue that may result if a process attempts to remove resources from a buffer before another produces them, 3) an events counter that counts the amount of resources that a process produces that are placed into a buffer and the number that is removed, and 4) an inter-process monitor that is a collection of data structures, variables, and procedures that work together to prevent mutual exclusion by using "WAIT" and "SIGNAL" instructions based on when a calling process has sufficient resources to work.

Each process is viewed as a collection of tasks. These tasks are scheduled to suitable processor to improve performance. This is not a widely used approach because:

· It requires characteristics of all the processes to be known in advance.

· This approach does not take into consideration the dynamically changing state of the system.

In this approach, a process is considered to be composed of multiple tasks and the goal is to find an optimal assignment policy for the tasks of an individual process. The following are typical assumptions for the task assignment approach:

· Minimize IPC cost (this problem can be modeled using network flow model)

· Efficient resource utilization

· Quick turnaround time

· A high degree of parallelism

**LOAD BALANCING APPROACH**

There are many Inter-Process Communication implementations that are both platform dependent and independent. Some of the platform independent implementations include: COBRA (Common Object Request Broker Architecture), Distributed Computing Environment (DCE), Message Bus (MBUS), ONC RPC, Lightweight Communications and Marshalling (LCM), Unix domain sockets, and XML RPC. Some platform specific implementations include: the Java Remote Method Invocation (RMI), Apple Computer's Apple Events, KDE's Desktop Communications Protocol (DCOP), Libt2n for C++ on Linux, Microsoft ActiveX, DCOM, and COM, and Solaris Doors.

There are several reasons for providing an environment that allows process cooperation:

Information sharing
Speedup
Modularity
Convenience
Privilege separation

In this, the processes are distributed among nodes to equalize the load among all nodes. The scheduling algorithms that use this approach are known as Load Balancing or Load Leveling Algorithms. These algorithms are based on the intuition that for better resource utilization, it is desirable for the load in a distributed system to be balanced evenly. This a load balancing algorithm tries to balance the total system load by transparently transferring the workload from heavily loaded nodes to lightly loaded nodes in an attempt to ensure good overall performance relative to some specific metric of system performance.

We can have the following categories of load balancing algorithms:

· Static: Ignore the current state of the system. E.g. if a node is heavily loaded, it picks up a task randomly and transfers it to a random node. These algorithms are simpler to implement but performance may not be good.

· Dynamic: Use the current state information for load balancing. There is an overhead involved in collecting state information periodically; they perform better than static algorithms.

· Deterministic: Algorithms in this class use the processor and process characteristics to allocate processes to nodes.

· Probabilistic: Algorithms in this class use information regarding static attributes of the system such as number of nodes, processing capability, etc.

· Centralized: System state information is collected by a single node. This node makes all scheduling decisions.

· Distributed: Most desired approach. Each node is equally responsible for making scheduling decisions based on the local state and the state information received from other sites.

· Cooperative: A distributed dynamic scheduling algorithm. In these algorithms, the distributed entities cooperate with each other to make scheduling decisions. Therefore they are more complex and involve larger overhead than non-cooperative ones. But the stability of a cooperative algorithm is better than of a non-cooperative one.

· Non-Cooperative: A distributed dynamic scheduling algorithm. In these algorithms, individual entities act as autonomous entities and make scheduling decisions independently of the action of other entities.

Static versus Dynamic

Static algorithms use only information about the average behavior of the system

Static algorithms ignore the current state or load of the nodes in the system

Dynamic algorithms collect state information and react to system state if it changed

Static algorithms are much more simpler

Dynamic algorithms are able to give significantly better performance.

Deterministic versus Probabilistic

Deterministic algorithms use the information about the properties of the nodes and the characteristic of processes to be scheduled

Probabilistic algorithms use information of static attributes of the system (e.g. number of nodes, processing capability, topology) to formulate simple process placement rules

Deterministic approach is difficult to optimize

Probabilistic approach has poor performance

Centralized versus Distributed

Centralized approach collects information to server node and makes assignment decision

Distributed approach contains entities to make decisions on a predefined set of nodes

Centralized algorithms can make efficient decisions, have lower fault-tolerance

Distributed algorithms avoid the bottleneck of collecting state information and react faster

Issues in designing Load-balancing algorithms

Load estimation policy

determines how to estimate the workload of a node

Process transfer policy

determines whether to execute a process locally or remote

State information exchange policy

determines how to exchange load information among nodes

Location policy

determines to which node the transferable process should be sent

Priority assignment policy

determines the priority of execution of local and remote processes

Migration limiting policy

determines the total number of times a process can migrate

In some cases the true load could vary widely depending on the remaining service time, which can be measured in several way:

Memoryless method assumes that all processes have the same expected remaining service time, independent of the time used so far

Pastrepeats assumes that the remaining service time is equal to the time used so far

Distribution method states that if the distribution service times is known, the associated process's remaining service time is the expected remaining time conditioned by the time already used

None of the previous methods can be used in modern systems because of periodically running processes and daemons

An acceptable method for use as the load estimation policy in these systems would be to measure the CPU utilization of the nodes

Central Processing Unit utilization is defined as the number of CPU cycles actually executed per unit of real time

It can be measured by setting up a timer to periodically check the CPU state (idle/busy)

**LOAD SHARING APPROACH**

Resource sharing is one of the major advantages obtained from distributed systems. Fair and reliable resource sharing is an active area of research in this field. In this paper we propose a framework for reliable and fair resource sharing in distributed systems. The goal of fairness is achieved by using concept of bank accounts, salaries and resource rates.

Several researchers believe that load balancing, with its implication of attempting to equalize workload on all the nodes of the system, is not an appropriate objective. This is because the overhead involved in gathering the state information to achieve this objective is normally very large, especially in distributed systems having a large number of nodes. In fact, for the proper utilization of resources of a distributed system, it is not required to balance the load on all the nodes. It is necessary and sufficient to prevent the nodes from being idle while some other nodes have more than two processes. This rectification is called the Dynamic Load Sharing instead of Dynamic Load Balancing.

The design of a load sharing algorithms require that proper decisions be made regarding load estimation policy, process transfer policy, state information exchange policy, priority assignment policy, and migration limiting policy. It is simpler to decide about most of these policies in case of load sharing, because load sharing algorithms do not attempt to balance the average workload of all the nodes of the system. Rather, they only attempt to ensure that no node is idle when a node is heavily loaded. The priority assignments policies and the migration limiting policies for load-sharing algorithms are the same as that of load-balancing algorithms.

**FRAMEWORK FOR RESOURCE SHARING IN DISTRIBUTED ENVIRONMENT**

We propose a framework for fair and reliable resource sharing among the systems. By "fair" we mean that we should not allow a system to just use resources from other systems but rather it should also provide its own resources to other systems in a proportional scale. Thus a system should not only be a service user but also a service provider in the distributed environment. We use the concept of bank account and salary as in to model the goal of fairness. Thus there is a monetary agent unit in the environment which will be act as a bank for the systems in the network. This monetary agent unit will be responsible for maintaining the bank balances for each system in the network, to deposit regular salaries to each system at regular times, and to adjust the balance between systems whenever one of them takes service (consumes resource) from the other system.

By "reliable" we mean that the consumer of the resource (CPU time, Storage etc...) should get good service from other systems and that he was not "cheated". Moreover any particular system should not be overloaded by offering a lot of services to other system. This will again be achieved by the concept of variable rates Thus our framework not only promotes fair sharing of resources among the systems but also reliable resource sharing. Resource implies any service offered by a system in the network like storage, computation. There can be one or more than one resources available in the network for sharing.

**OPERATING SYSTEM FORPARALLEL PROCESSING**

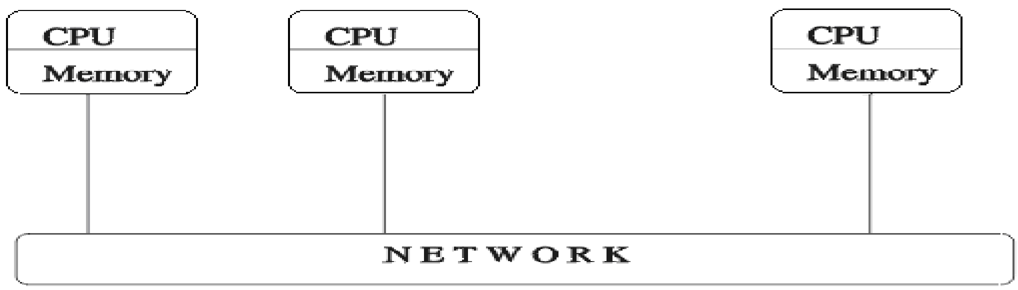
Parallel computing involves the design of a computing system that uses more than one processor to solve a single problem. For example, if two arrays with ten elements each must be added, two processors can be used to compute the results. One processor computes the sum of the first five elements and the second processor computes the sum of the second five elements. After the computation, the results from one processor must be communicated to the other processor. Before starting the computation, both processors agree to work on independent sub-problems. Each processor works on a sub-problem and communicates when the solution is available. Theoretically, a two-processor computer should add the array of numbers twice as fast as a single-processor computer. In practice, there is overhead and the benefits of using more processors decrease for larger processor configurations.

Obtaining a Unix workstation for the cost of a PC has been one of the benefits of using Linux. This idea has been carried a step further by linking together a number of Linux PCs. Several research projects are underway to link PCs using high performance networks. High speed networking is a hot topic and there are a number of projects using Linux to develop a low latency and high bandwidth parallel machine. (One URL is http://yara.ecn.purdue.edu/~pplinux.)Currently, there is not much high level support for shared memory programming under SMP Linux. The basic Linux mechanisms for sharing memory across processors are available. They include the System V Inter-Processor Communication system calls and a thread library. But, it will be some time before a parallel C or C++ compiler will be available for Linux. Parallel programming can still be done on an SMP Linux machine or on a cluster of Linux PCs using message passing.

Parallel computing is advantageous in that it makes it possible to obtain the solution to a problem faster. Scientific applications are already using parallel computation as a method for solving problems. Parallel computers are routinely used in computationally intensive applications such as climate modeling, finite element analysis and digital signal processing. New commercial applications which process large amounts of data in sophisticated ways are driving the development of faster computers. These applications include video conferencing, data mining and advanced graphics. The integration of parallel computation, multimedia technology and high performance networking has led to the development of video servers. A video server must be capable of rapidly encoding and decoding megabytes of data while simultaneously handling hundreds of requests. While commercial parallel applications are gaining popularity, scientific applications will remain important users of parallel computers. Both application types are merging as scientific and engineering applications use large amounts of data and commercial applications perform more sophisticated operations. Parallel computing is a broad topic and this article will focus on how Linux can be used to implement a parallel application. We will look at two models of parallel programming: message passing and shared memory constructs.

**Message Passing**

Conceptually, the idea behind message passing is simple—multiple processors of a parallel computer run the same or different programs, each with private data. Data is exchanged between processors when needed. A message is transmitted by a sender processor to a receiver processor. One processor can be either a sender or a receiver processor at any time. The sender processor can either wait for an acknowledgement after sending or it can continue execution. The receiver processor checks a message buffer to retrieve a message. If no message is present, the processor can continue execution and try again later or wait until a message is received. Multiple sends and receives can occur simultaneously in a parallel computer.




A Parallel Computer with Distributed Memory

All processors can exchange data with all other processors. The routing of messages is handled by the operating system. The message-passing model can be used on a network of workstations or within a tightly coupled group of processors with a distributed memory.

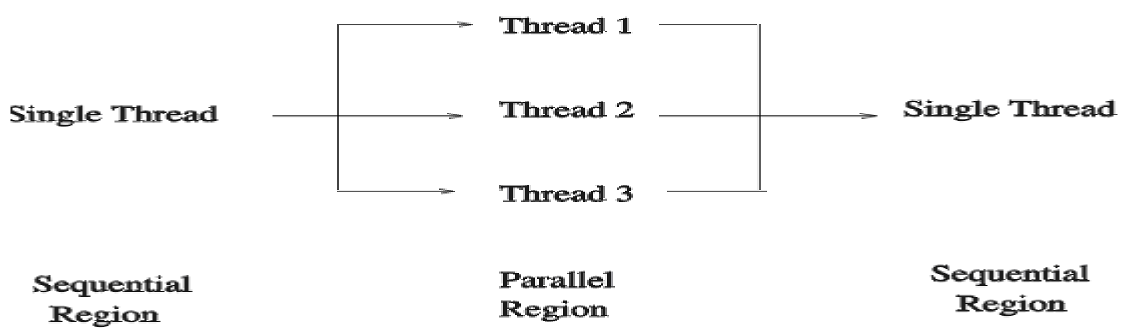
The number of hops between processors can vary depending on the type of inter-connection network.

Message passing between processors is achieved by using a communication protocol. Depending on the communication protocol used, the send routine usually accepts a destination processor ID, a message type, the start address for the message buffer and the number of bytes to be transmitted. The receive routine can receive a message from any processor or from a particular processor. The message can be of any particular type. Most communication protocols maintain the order in which messages are sent between a pair of processors. For example, if processor 0, sends a message of type a followed by a message of type b to processor 1, then when processor 1 issues a receive from processor 0 for a generic message type, the message of type a will be received first. However, in a multi-processor system, if a processor issues a receive from any processor, there is no guarantee of the order of messages received from the sending processors. The order in which messages are transported depends on the router and the traffic on the network. To maintain the order of messages sent, the safest way is to use the source processor number and message type.

Message passing has been used successfully to implement many parallel applications. But a disadvantage of message-passing is the added programming required. Adding message-passing code to a large program requires considerable time. A domain decomposition technique must be chosen. Data for the program must be divided such that there is minimal overlap between processors, the load across all processors is balanced and each processor can independently solve a sub-problem. For regular data structures, the domain decomposition is fairly straightforward, but for irregular grids, dividing the problem so that the load is balanced across all processors is not trivial. Another disadvantage of message passing is the possibility of deadlock. It is very easy to hang a parallel computer by misplacing a call to the send or receive routines. So, while message passing is conceptually simple, it has not been adopted fully by the scientific or commercial communities.

**SHARED MEMORY CONSTRUCTS**

Another approach to parallel programming is the use of shared memory parallel language constructs. The idea behind this scheme is to identify the parallel and sequential regions of a program (Figure). The sequential region of a program is run on a single processor while the parallel region is executed on multiple processors. A parallel region consists of multiple threads which can run concurrently.




Parallel and Sequential Regions of a Program

For some programs, identifying parallel and sequential regions may be a trivial task, but for other programs it may involve modifying the sequential program to create parallel regions. The easiest approach is to rely on the compiler to determine parallel regions. This is the automatic parallelization approach which usually gives poor results. Most compilers are conservative when introducing parallelism and will not parallelize code if there is any ambiguity. For example, if elements of an array **x** are accessed through an index array, e.g., **x(index(i))**, in a loop.

## PART-B(6 MARKS)
## POSSIBLE QUESTIONS

1. What is Synchronization? Explain in detail about Clock Synchronization.
2. Discuss in detail about Event Ordering with an example
3. Discuss about the Mutual Exclusion.
4. Explain desirable features of a Good Global Scheduling Algorithms.
5. Explain Load Balancing Algorithms and their approach.
6. What is Dynamic Load Balancing? And explain in detail about Load Sharing Approach.

## PART-C(10 MARKS)
## POSSIBLE QUESTIONS

1. What is Deadlock? And explain in detail about Deadlock Modeling.
2. What is Resource Manager? Explain in detail about Task Assignment Approach.
3. Illustrate with an example explain in detail Synchronization.

**KARPAGAM ACADEMY OF HIGHER EDUCATION**

**Department of Computer Science**

**II M.Sc( CS) (BATCH 2017-2019)**

**Distributed Operating Systems**

**PART-A OBJECTIVE TYPE/ MULTIPLE CHOICE QUESTIONS**

| **ONLINE EXAMINATIONS ONE MARK QUESTIONS** | | | | | |
|---|---|---|---|---|---|
| **Questions** | **opt1** | **opt2** | **opt3** | **opt4** | **Answer** |
| A distributive system consists of collection of ______________________ process | Direct | Dynamic | Distinct | Decode | Distinct |
| clock synchronization are of _____________________types | 1 | 2 | 5 | 7 | 2 |

| | | | | | |
|---|---|---|---|---|---|
| A computer clock usually consists of _________________________components | 5 | 7 | 3 | 4 | 3 |
| The value of register each intrrupt is called ____________ | clock time | deadlock | clockcycle | clocktick | clocktick |
| clock synchronization algorithm may be broadly classified as _______________________ | centralized,distributed | distributed,centralized | distributed clock | centralized clock | centralized,distributed |
| centralized clock synchronization algorithms suffer from ___________________major drawbacks | 5 | 7 | 6 | 2 | 2 |
| The happened before relation on a set of events satisfy __________________ conditions | 1 | 3 | 2 | 4 | 3 |
| Lamport provided a solution _______________________ y | 1977 | 1999 | 1978 | 1973 | 1978 |
| The Logical clocks must satisfy ______________________ | 3 | 2 | 1 | 4 | 3 |
| Each process Pi increments Ci between any two successive events _______________implementatio rules | 1R2 | 1R1 | 1R3 | 1R0 | 1R1 |
| The time stamps assigned to the events by the system of logical clocks must be satisfy following _______ | clock condition | clock | clock condition | clock direction | clock condition |

| | | | | | |
|---|---|---|---|---|---|
| If a and b are two events with in same process Pi and a occurs before b then _______________ | ci(a)<ci(b) | ci(b)<ci(a) | ci<cj | ci(a)=ci9b) | ci(a)<ci(b) |
| A distributive system have been proposed ____________________ approach | 5 | 3 | 4 | 2 | 3 |
| The clock condition mentioned above is satisfy it ________ | 3 | 2 | 1 | 3 | 3 |
| The difference in time values of two clock is called ______ | clock | skew | clockskew | nodes | clockskew |
| Active server algorithm that overcomes the drawback of the above algorithm is ___________________ | RSA | Fuzzy | Active | Berkley | Berkley |
| NTP means ______________________ | Network time procedure | Network total protocol | Network Time Protocol | Network time pooling | Network Time Protocol |
| DTS means ________________________ | Distributed Time Service | Detail Time service | Distributed Time Server | Distributed Type Service | Distributed Time Service |
| One DTS Server of each LAN is designed a ___________ | Local | global | Internal | External | global |
| The Distributive techniques broadly classified into _______ | 8 | 7 | 5 | 3 | 3 |

| | | | | | |
|---|---|---|---|---|---|
| The relation on set of events denoted _______________ | -----> | <-----> | <----- | Equal to | Equal to |
| UTC means _________________________ | Universal Time | Coordinated Universal time | Code Unit time | Unique Coding type | Coordinated Universal time |
| The value in the constant register is chosen so that _________________ clock ticks occur the second | 100 | 50 | 150 | 60 | 60 |
| Synchronization of the clocks of different nodes of the system________________________ | Mutual | Mode | Node | External | Mutual |
| Synchronization of the computer clocks with ___________________ clocks. | runtime | clock | real time | internal | real time |
| In centralized clock synchronization algorithms one node has a real time ____________________ | receiver | clock | client | server | receiver |
| API means ____________________________ | Access Protocol Internet | clock | Application Programming Interface | Application Provide Interface | Application Programming Interface |
| DCE means ________________________ | Distributed Computing Event | Distributed Computing Environment | Denial Computer Event | Denial Covering Event | Distributed Computing Environment |
| DTS client node runs a demon process called a_________ | DTS clock | clock | DTS client | DTS clerk | DTS clerk |

| | | | | | |
|---|---|---|---|---|---|
| NTP can founded in ________________ year | 1991 | clock | 1999 | 1996 | 1991 |
| IPC stands for _______________________ | Inter Permannent Communication | clock | Inter persistent Communication | Inter Process communication | Inter Process communication |
| A Scheduling algorithm is said to be ________________ if | stable | clock | both A and B | unstable | Unstable |
| In which each process submitted by a user for processing is viewed as a collection of related tasks_______ | Task assignment approach | clock | Load-sharing approach | task management | Task assignment approach |
| In which all the process submitted by the users are distributed among the nodes ______________ | Task assignment approach | clock | Load-sharing approach | Load-Balancing approach | Load-Balancing approach |
| In which simply attempts to conserve the ability of the system to perform work by assuring that no node is idle_________________ | Task assignment approach | clock | Load-sharing approach | load assignment | Load-sharing approach |
| A process split into pieces called _______________ | Processor | clock | Tasks | microprocessor | Tasks |
| Distributed dynamic scheduling cateroized into_________ | 3 | clock | 4 | 2 | 2 |
| Policy determines how to estimate the workload of a particular node of the system_______________ | process transfer | clock | Location | Load estimation | Load estimation |

| | | | | | |
|---|---|---|---|---|---|
| policy determines to which node a process selecetd for transfer should be sent | process transfer | clock | Location | Load estimation | Location |
| The ______________ priority assignment rule yields the worst time performance of three policies _________ | Altruistic | clock | Selfish | asymmetric | Selfish |
| The __________________ priority assignment rule achieves the best time performance of three policies. | Altruistic | clock | Selfish | all the above | Altruistic |
| _____________ deals with process of deciding which process should be assigned to which processor | Process migration | clock | Process allocation | process memory | Process allocation |
| _________________ deals with fine-grained parallelism for better utilization of the processing of the system | Process migration | clock | Process allocation | Thread | Thread |
| __________________deals with the movement of a processor from its current location to the processor | Process migration | clock | Process allocation | resource allocation | Process migration |
| FIFO stands for____________________ | First In First Out | clock | Find In Find Out | first in found out | First In First Out |
| A _______mutex variable is one that allows a thread to lock an already locked mutex variable | Fast | clock | Non-Recursive | recursive | recursive |
| A _______mutex variable is one that neither allows a thread to block | Fast | clock | Non-Recursive | recursive | Non-Recursive |

| | | | | | |
|---|---|---|---|---|---|
| In ____________mehod the first thread of the first non empty highest priority queue is selected to run | First In First Out | clock | Round Robin | default | First In First Out |
| In _______________method first nonempty highest priorit | First In First Out | clock | Round Robin | synchronization | Round Robin |
| In ____________ method the threads on all proority queues are run after another using a Round Robin algorithm | First In First Out | clock | Round Robin | Default | Default |
| A __________ is a sub system of an operating system tha performs file mangement activities | Object | clock | Storage | objecting | Object |
| A _____________________ distributed file system typically provides ____________ types of services | 4 | clock | 3 | 5 | 3 |
| ___________provides a mapping between text names for files and references to files | Storage Service | clock | Name Service | naming server | Name Service |
| Transparencies are of ______________ types | 4 | clock | 2 | 3 | 4 |
| RFS stands for _______________________ | Resource File System | clock | Remote file system | remote file sharing | Remote file system |
| __________property ensures that to the outside world all the operations of transaction appear | Serializability | clock | Performance | Atomicity | Atomicity |

| | | | | | |
|---|---|---|---|---|---|
| Serializability property is also known as _____________ property | Isolation | clock | Durability | atomicity | Isolation |
| Permanence property is also known as____________ property | Isolation | clock | Durability | atomicity | Durability |
| DFS stands for _____________________ | Direct File System | clock | Distributed File System | polarized | Distributed File System |
| ___________________ is the DFS local file system | Token Manager | clock | Episode | RFS | Episode |

## UNIT-IV

## SYLLABUS

Distributed file system: Introduction – desirable features of a good distributed file system – file models – file accessing models.
Naming: Introduction – desirable features of a good naming system – fundamental terminologies and concepts.

## DISTRIBUTED FILE SYSTEM

In the client/server network model a computer plays a centralized role and is known as a server all other computers in the network are known as clients. All client computers access the server simultaneously for files, database, docs, spreadsheets, web pages and resources like hard diver, printer, fax modem, CD/DVD ROM and others. In other words, all the client computes depends on the server and if server fails to respond or crash then networking/communication between the server and the client computes stops.

The Client – Server Model
Blocking Vs Non Blocking Primitives
Buffered Versus Un buffered Primitives
Implementation of Client – Server Model.

## INTRODUCTION

Client/server model is a concept for describing communications between computing processes that are classified as service consumers (clients) and service providers (servers). Figure (a) presents a simple C/S model. The basic features of a C/S model are:

1. Clients and servers are functional modules with well defined interfaces (i.e., they hide internal information). The functions performed by a client and a server can be implemented by a set of software modules, hardware components, or a combination thereof.

2. Each client/server relationship is established between two functional modules when one module (client) initiates a service request and the other (server) chooses to respond to the service request. For a given service request, clients and servers do not reverse roles (i.e., a client stays a client and a server stays a server). However, a server for SR R1 may become a client for SR R2 when it issues requests to another server (see Figure). For example, a client may issue an SR that may generate other SRs.

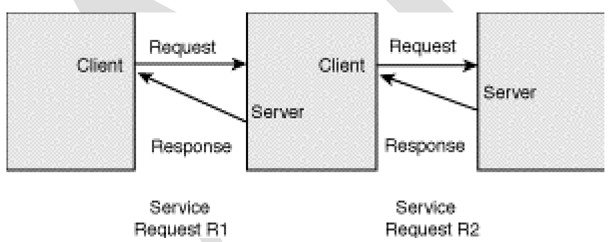
3. Information exchange between clients and servers is strictly through messages (i.e., no information is exchanged through global variables). The service request and additional information is placed into a message that is sent to the server. The server's response is similarly another message that is sent back to the client. This is an extremely crucial feature of C/S model

A computer that has access to services over a computer network. The computer providing the services is a server.

Client-Server Architecture: An information-passing scheme that works as follows: a client program, such as Mosaic, sends a request to a server. The server takes the request, disconnects from the client and processes the request. When the request is processed, the server reconnects to the client program and the information is transferred to the client. This architecture differs from traditional Internet databases where the client connects to the server and runs the program from the remote site.

4. Messages exchanged are typically interactive. In other words C/S model does notsupport an off-line process. There are a few exceptions. For example, message queuing systems allow clients to store messages on a queue to be picked up asynchronously by the servers at a later stage.

5. Clients and servers typically reside on separate machines connected through a network. Conceptually, clients and servers may run on the same machine or on separate machines. The implication of the last two features is that C/S service requests are real-time messages that are exchanged through network services. This feature increases the appeal of the C/S model (i.e., flexibility, scalability) but introduces several technical issues such as portability, interoperability, security, and performance.




**Conceptual Client/Server Model**

**CHARACTERISTICS**

**Characteristics of a Client**

- ✓ **Request sender is known as client**
- ✓ Initiates requests
- ✓ Waits for and receives replies.
- ✓ Usually connects to a small number of servers at one time
- ✓ Typically interacts directly with end-users using a graphical user interface

**CHARACTERISTICS OF A SERVER**

- ✓ **Receiver of request which is send by client is known as server**
- ✓ Passive (slave)

- ✓ Waits for requests from clients
- ✓ Upon receipt of requests, processes them and then serves replies
- ✓ Usually accepts connections from a large number of clients
- ✓ Typically does not interact directly with end-users

## INTRODUCTION

Server Architecture comprises of various types of servers, such as File Server, Print Server, and Email Servers. In this lesson, we will be discussing about these servers and various protocols used in Mail Servers, we also see a comparative study of various mail servers.

## ORGANIZATION

In a client-server environment, an organization's files, and sometimes itsapplications,arestorednot on individual desktop computers but on centralized servers instead. That "client-server" structure has benefits that range from tighter system security to easier file backups.

## DESIRABLE FEATURES OF A GOOD DISTRIBUTED FILE SYSTEM

In a client-server environment, companies use a centralized file and print server to store individual user documents. Users store the files they create on a shared network drive, with each user allocated a dedicated storage space on the server. The printer drivers reside on the server as well, and users connect to the network printers through that server. Each individual desktop computer is a node, or client, on that centralized file and print server.

## APPLICATION SERVER

Some companies that use the client-server organization also use a centralized repository for their programs and applications. Instead of having the programs loaded on each client machine, those programs are loaded to a central application server. Clients connect to the network and access the programs they need, using those programs to create documents, spreadsheets and databases, just as they would if the programs were loaded on their individual client computers.

## EASIER BACKUPS

Client-server organization provides multiple users working on a big project, such as a team making a newspaper every day, with an easier and more effective way to back up their critical files. When each user stores files locally on a PC, there is always a chance that the PC could suffer hard drive failure that would destroy those files for good. But when users store their files on a centralized file server, the network administrator can back up those files every night, and they can be recovered in the case of an equipment failure or accidental deletion. Many modern file servers also take file snapshots throughout the day, providing almost instantaneous recovery of damaged or deleted files.

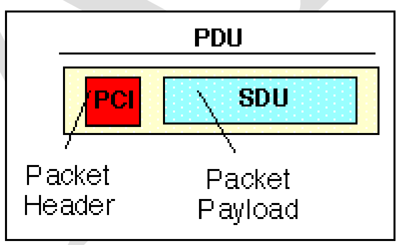
**CENTRALIZED CONTROL**

The client-server structure provides greater security and easier management than a network of individual computers. When applications are stored on a central server, it is easier for network administrators to keep track of licenses and available seats. Storing individual user files on the server makes backup and recovery easier, while allowing users in multiple locations to access those files any time they need to.

**FILE MODELS**

A communications protocol defines the rules for sending blocks of data (each known as a Protocol Data Unit (PDU)) from one node in a network to another node. Protocols are normally defined in a layered manner and provide all or part of the services specified by a layer of the OSI reference model. A protocol specification defines the operation of the protocol and may also suggest how the protocol should be implemented. It consists of three parts:

1. Definition of Protocol Control Information (PCI) format which forms the PDU header
2. Definition of procedures for transmitting and receiving PDUs
3. Definition of services provided by the protocol layers




A Protocol Data Unit

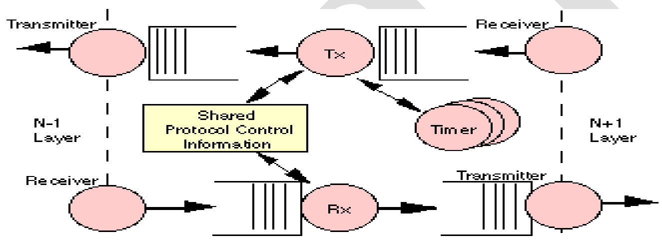
The PDUs exchanged have two parts: a header (also known as the Protocol Control Information (PCI)) and a payload (also known as a Service Data Unit (SDU)). The protocol does not define or constrain the data carried in the payload part. It does specify the format of the PCI, defining the fields which are present and the way in which the patterns of bits are to be interpreted. A protocol also defines the procedures which determine how the PDU will be processed at the transmit and receive nodes. The procedures specify the valid values for the PCI fields, and the action be taken upon reception of each PCI value (usually based on stored control information). Examples of procedures which are implemented in protocols include:

- error recovery (e.g. the checkpoint procedure, the go-back-n procedure)

- flow control
- segmentation
- service access point selection
- connection management

The documents which define a protocol procedures are usually large and are seldom concrete (i.e. they may not be directly translated to executable programs). They typically define the actions to be taken when a particular condition is detected, but not how the condition is to be detected.

It has been said that "Part of what makes a protocol mature is good implementation guidelines and folklore". The success of the TCP/IP protocol suite is largely due to the "industrial strength" code available in freely distributed reference implementations.




A protocol implemented by several processes (represented by circles) communicating using queues of PDUs, a shared information area (shown as a rectangle) and function calls between the processes (represented by arrows)

Protocols are usually implemented by writing a number of programs (processes) which communicate with one another through queues and by function calls. One or more timers are also usually required to ensure correct operation of the protocol. To start and stop timers, a protocol normally uses an interface to the computer's operating system. This interface is also used to request new (empty) buffers for received PDUs (or PDUs created by the layer) and to release buffers which are no longer needed by the protocol.

Protocols are generally described using a layered architecture known as the OSI reference model. Which abstracts the details of the protocol and allows a simple description of the service provided by the protocol to the protocol layer above and the service required by protocol layer from the layer below? Examples of protocols include:

- Link Layer - HDLC, MAC, ARP
- Network Layer - IP, ICMP
- Transport Layer - UDP, TCP

**FILE ACCESSING MODEL**

File servers generally offer some form of system security to limit access to files to specific users or groups. In large organizations, this is a task usually delegated to what is known as directory services such as Novell's directory or Microsoft's Active Directory. These servers work within the hierarchical computing environment which treat users, directories, computers, applications and files as distinct but related entities on the network and grant access based on user or group credentials. In many cases, the directory service spans many file servers, potentially hundreds for large organizations. In the past, and in smaller organizations, authentication can take place directly to the server itself.

Integrity provides endpoint security and policy enforcement that protect enterprise networks proactively from worms, spyware, and hacker attacks that evade other security technologies. Quickly and easily deployed and administered, the integration of Integrity with InterSpect security appliances enables comprehensive internal security with minimal cost of ownership.

**NAMING : INTRODUCTION**

In telecommunication, the term **file server** has the following meanings:

In the client/server model, a file server is a computer responsible for the central storage and management of data files so that other computers on the same network can access the files.
A file server allows users to share information over a network without having to physically transfer files by floppy diskette or some other external storage device. Any computer can be configured to be a host and act as a file server. In its simplest form, a file server may be an ordinary PC that handles requests for files and sends them over the network. In a more sophisticated network, a file server might be a dedicated network-attached storage (NAS) device that also serves as a remote hard disk drive for other computers, allowing anyone on the network to store files on it as if to their own hard drive.

The naming facility of a distributed operating system enables users and programs to assign character-string names to objects and subsequently use these names to refer to those objects.

The locating facility, which is an integral part of the naming facility, maps an object's name to the object's location in a distributed system.

The naming and locating facilities jointly form a naming system that provides the users with an abstraction of an object that hides the details of how and where an object is actually located in the network.

It provides a further level of abstraction when dealing with object replicas. Given an object name, it returns a set of the locations of the object's replicas.

The naming system plays a very important role in achieving the goal of v location transparency, facilitating transparent migration and replication of objects, object sharing.

**DESIRABLE FEATURES OF A GOOD NAMING SYSTEM**

A form of disk storage that hosts files within a network; file servers do not need to be high-end but must have enough disk space to incorporate a large amount of data. Many people mistake file servers for a high-end storage system, but in reality, file servers do not need to possess great power or super fast computer specifications.

A computer program that allows different programs, running on other computers, to access the files of that computer

In common parlance, the term **file server** refers specifically to a computer on which a user can map or mount a disk drive or directory so that the directory appears to be on the machine at which the user is sitting. Additionally, on this type of file server, the user can read or write a file as though it were part of the file system of the user's computer. Files and directories on the remote computer are usually accessed using a particular protocol, such as WebDAV, SMB, CIFS, NFS, Appletalk or their mutations.

Although files can be sent to and received from most other computers unless their primary function is access by the above means, they are generally not considered file servers as such.

1. Location transparency. Location transparency means that the name of an object should not reveal any hint as to the physical location of the object. That is, an object's name should be independent of the physical connectivity or topology of the system, or the current location of the object.

2. Location independency. For performance, reliability, availability, and security reasons, distributed systems provide the facility of object migration that allows the movement and relocation of objects dynamically among the various nodes of a system. Location independency means that the name of an object need not be changed when the object's location changes. Furthermore, a user should be able to access an object by its same name irrespective of the node from where he or she accesses it (user migration). Therefore, the requirement of location independency calls for a global naming facility with the following two features:

An object at any node can be accessed without the knowledge of its physical location (location independency of request-receiving objects).

An object at any node can issue an access request without the knowledge of its own physical location (location independency of request-issuing objects).

This property is also known as user mobility.

3. Scalability. Distributed systems vary in size ranging from one with a few nodes to one with many nodes. Moreover, distributed systems are normally open systems, and their size changes dynamically. Therefore, it is impossible to have an a priori idea about how large the set of names to be dealt with is liable to get. Hence a naming system must be capable of adapting to the dynamically changing scale of a distributed system that normally leads to a change in the size of the name space. That is, a change in the system scale should not require any change in the naming or locating mechanisms.

4. Uniform naming convention. In many existing systems, different ways of naming objects, called naming conventions, are used for naming different types of objects. For example, file names typically differ from user names and process names. Instead of using such non uniform naming conventions, a good naming system should use the same naming convention for all types of objects in the system.

5. Multiple user-defined names for the same object. For a shared object, it is desirable that different users of the object can use their own convenient names for accessing it. Therefore, a naming system must provide the flexibility to assign multiple user-defined names to the same object. In this case, it should be possible for a user to change or delete his or her name for the object without affecting those of other users.

6. Group naming. A naming system should allow many different objects to be identified by the same name. Such a facility is useful to support broadcast facility or to group objects for conferencing or other applications.

7. Meaningful names. A name can be simply any character string identifying some object. However, for users, meaningful names are preferred to lower level identifiers such as memory pointers, disk block numbers, or network addresses. This is because meaningful names typically indicate something about the contents or function of their referents, are easily transmitted between users, and are easy to remember and use. Therefore, a good naming system should support at least two level of object identifiers, one convenient for human users and one convenient for machines.

8. Performance. The most important performance measurement of a naming system is the amount of time needed to map an object's name to its attributes, such as its location. In a distributed environment, this performance is dominated by the number of messages exchanged during the name-mapping operation. Therefore, a naming system should be efficient in the sense that the number of messages exchanged in a name-mapping operation should be as small as possible.

9. Fault tolerance. A naming system should be capable of tolerating, to some extent, faults that occur due to the failure of a node or a communication link in a distributed system network. That is, the naming system should continue functioning, perhaps in a degraded form, in the event of these failures. The degradation can be in performance. functionality, or both but should be proportional, in some sense, to the failures causing it.

10. Replication transparency. In a distributed system, replicas of an object are generally created to improve performance and reliability. A naming system should support the use of multiple copies of the same object in a user-transparent manner. That is, if not necessary, a user should not be aware that multiple copies of an object are in use.

11. Locating the nearest replica. When a naming system supports the use of multiple copies of the same object, it is important that the object-locating mechanism of the naming system should always supply the location of the nearest replica of the desired object. This is because the efficiency of the object accessing operation will be affected if the object-locating mechanism does not take this point into consideration.

## FUNDAMENTAL TERMINOLOGIES AND CONCEPT

### File and print

Traditionally, file and print services have been combined on the same computers due to similar computing requirements for both functions. Usually, such computers are distinct from application and database servers, which have different, usually more processorintensive, and requirements. However, as computing power increases and file serving requirements remain relatively constant, it is more common to see these functions combined on the same machine.

## PRINT SERVER

A **print server**, or printer server, is a computer or device to which one or more printers are connected, which can accept print jobs from external client computers connected to the print server over a network. The print server then sends the data to the appropriate printer that it manages. The term **print server** can refer to:

1. A host computer running Windows OS with one or more shared printers. Client computers connect using Microsoft Network Printing protocol.
2. A computer running some operating system other than Windows, but still implementing the Microsoft Network Printing protocol (typically Samba running on a UNIX or Linux computer).
3. A computer that implements the LPD service and thus can process print requests from LPD clients.
4. A dedicated device that connects one or more printers to a LAN. It typically has a single LAN connector, such as an RJ-45 socket, and one or more physical ports (e.g. serial, parallel or USB (Universal Serial Bus)) to provide connections to printers. In essence this dedicated device provides printing protocol conversion from what was sent by client computers to what will be accepted by the printer. Dedicated print server devices may support a variety of printing protocols including LPD/LPR over TCP/IP, NetWare, NetBIOS/NetBEUI over NBF, TCP Port 9100 or RAW printer protocol over TCP/IP, DLC or IPX/SPX. Dedicated server appliances do not provide spooling or print queue services, since they typically have very little memory.
5. A dedicated device similar to definition 4 above, that also implements Microsoft Networking protocols to appear to Windows client computers as if it were a print server defined in 1 above. The term **print server** normally has the meaning defined in 1 or 2 above, while the term **print server device** usually refers to definition 4.

## ELECTRONIC MAIL SERVER

A mail transfer agent or MTA (also called a mail transport agent, message transfer agent, mail server, SMTPD (short for SMTP daemon), or a mail exchanger(MX) in the context of the Domain Name System) is a computer program or software agent that transfers electronic mail messages from one computer to another. It receives messages from another MTA (relaying), a mail submission agent (MSA) that itself got the mail from a mail user agent (MUA), or directly from an MUA, thus acting as an MSA itself. The MTA works behind the scenes, while the user usually interacts with the MUA.

The delivery of e-mail to a user's mailbox typically takes place via a mail delivery agent (MDA); many MTAs have basic MDA functionality built in, but a dedicated MDA like procmail can provide more sophistication. According to various surveys the most popular mail server software are sendmail, Postfix, Microsoft Exchange Server, Exim, IMail (by Ipswitch, Inc.), MDaemon by Alt-N Technologies, MailEnable, Merak Mail Server and mail. The Mail Channels survey also found that many organizations use the services of e-mail security services such as Postini, MXLogic or Concentric Hosting to receive e-mail. This is a **list of mail servers**: mail transfer agents, mail delivery agents, and other computer software which provide e-mail services. SMTP, POP/IMAP, Mail filtering

**SMTP**

Apache James, Atmail, AXIGEN, Citadel, CommuniGate Pro, Courier, Eudora Internet Mail Server, Exim, Hexamail server, IBM Lotus Domino, IpBrick, Ipswitch IMail Server ,Kerio MailServer, MailEnable Mail Server, Mailtraq, Merak Mail Server, MercuryMail Transport System, MeTA1 (successor of the sendmail X project), Microsoft Exchange Server, MMDF, Novell GroupWise, Novell NetMail, Open-Xchange, Post Cast Server, Postfix, PostPath Email and Collaboration Server, qmail, Scalix, Sendmail, Smarter Mail, SparkEngine, Sun Java System, WinGate, WorkgroupMail, Xmail, XMS Email Application Server, Zimbra, ZMailer

**POP/IMAP**

Apache James, Axigen, Binc IMAP - uses Maildir, Bluebottle, Citadel - uses a database-driven mail store, CommuniGate Pro, Courier Mail Server - uses Maildir format, Cyrus IMAP server, Dovecot, Eudora Internet Mail Server, Hexamail server, IpBrick, Ipswitch IMail Server, Malware Communication Server (Free open source [multi-platform] mail server), Kerio MailServer, Lotus Domino IMAP4 Server, MailEnable Mail Server, Mailtraq, Merak Mail Server, MercuryMail Transport System, Microsoft Exchange Server, Microsoft Windows POP3 Service, Novell GroupWise, Novell NetMail, Open-Xchange, Oryx Archiveopteryx, PostPath, Qpopper, SmarterMail, UW IMAP - uses mbox format, WinGate, WorkgroupMail, Zimbra

**MAIL FILTERING**

ASSP, Bayesian filters, Bogofilter, DSPAM, Hexamail Guard, maildrop, Mailtraq, Procmail, PureMessage, SurfControl, SpamAssassin, WinGate, WorkgroupMail, Gattaca Serve, Vipul's Razor

**COMPARISON OF MAIL SERVERS**

This is a **comparison of mail servers**: mail transfer agents, mail delivery agents, and other computer software which provide e-mail services.

**DISTRIBUTED DATA BASES SYSTEMS**

**INTRODUCTION**

A distributed database system consists of multiple independent databases that operate on two or more computers that are connected and share data over a network. The databases are usually in different physical locations. Each database is controlled by an independent DBMS, which is responsible for maintaining the integrity of its own databases.

In extreme situations that databases might be installed on different hardware, different operating systems, and could even use DBMS software from different vendors. That last contingency is the hardest to handle. Most current distributed databases function better if all of the environments are running DBMS software from the same vendor.

**NEED FOR DISTRIBUTED DATABASE**

When an organization is geographically dispersed, it may choose to store its database on a central computer or to distribute them to local computers (or a combinatory of both). The following conditions encourage the need of distributed database in a business organization:

**DISTRIBUTION AND AUTONOMY OF BUSINESS UNITS:**

Divisions, departments, and facilitates in modern organizations are often geographically (and possibly internationally) distributed. Often each unit has the authority to create its own information systems, and often these units want local data over which they can have controls.
**Data sharing:** Even moderately complex business decisions require sharing data across business units, so it must be convenient to consolidate data across local databases on demand.

**DATA COMMUNICATIONS COSTS AND RELIABILITY:**

The cost to ship large quantities of data cross a communications network or to handle a large volume of transactions from remote sources can be high. It is often more economical to locate data and applications close to where they are needed. Also, dependence on data communications can be risky, so keeping local copies or fragments of data can be reliable way to support the need for rapid access to data across the organization.

**PART-B(6 MARKS)**
**POSSIBLE QUESTIONS**

1. What is file? And explain main purpose of using and services of files.
2. Brief about the File Accessing Model.
3. Explain about the concept of accessing the remote files.
4. Discuss in detail about Unit of data transfer in files.
5. Discuss in detail about Name Space and Name Server.
6. What is Context? And explain in detail about Context.
7. Brief about the fundamental Name Resolution absolute and Relative Names.

**PART-C(10 MARKS)**
**POSSIBLE QUESTIONS**

1. Describe about desirable features of Distributed File System
2. What is Naming? And explain in detail about desirable features of Good Naming System.
3. Explain about the fundamental terminologies and concept of Naming System.
4. Discuss in detail about File System With an Example.

KARPAGAM ACADEMY OF HIGHER EDUCATION
Department of Computer Science
II M.Sc( CS) (BATCH 2017-2019)
Distributed Operating Systems

| ONLINE EXAMINATIONS | ONE MARK QUESTIONS | | | | |
|---|---|---|---|---|---|
| QUESTIONS | OPT1 | OPT2 | OPT3 | OPT4 | ANSWER |
| In a _______System ,a file is a object that comes into existence by explicit creation. | computer | memory | sharing | files | computer |
| There are _____main purpose of using files. | 1 | 2 | 3 | 4 | 2 |
| The two main purposes of using files is _______ | Permanent storage of information and System | Sharing of information and memory | Permanent Storage of information and sharing information | Sharing and resource allocation | Permanent Storage of information and sharing information |
| ______ is achieved by storing a file on a Secondary storage media. | Sharing of information | Permanent Storage of information | Both A and B | Memory Storage | Permanent Storage of information |
| ______Files provide a natural and easy means of information sharing | Permanent Storage of information | Information Sharing | Sharing of information | memory storage | Sharing of information |
| A ______System provider similar abstraction to the users of a distributed System. | distributed file | shared file | memory file | files | distributed file |
| In addition to the advantages of permanent storage and sharing of information provided by the file System of a _______ System. | Double –Processor | Single processor | Distributed file | Processor | Single processor |

| | | | | | |
|---|---|---|---|---|---|
| ________ distributed file system allows a file to be transparently accessed by processes of any node of the System. | Remote information sharing | Permanent Storage of information | Sharing of information | Distributed Sharing | Remote information sharing |
| In a distributed System, ________ implies that a user should not be forced to work on a Specific node. | Availability | Remote information sharing | User mobility | workstation | User mobility |
| A Distributed file system normally allows a user to work on ______ nodes at different times. | same | different | block | both | different |
| Each copy is called a _______ of the file. | Replica | replicant | availability | node | Replica |
| In a ideal design, both the existence of multiple copies and their locations are hidden from the _____ | server | sender | receiver | client | client |
| ______ are relatively expensive compared to the cost of most other parts in a Workstation. | diskless | diskless workstation | diskdriver | mobility | diskdriver |
| ______ is more economical, is less noisy, and generates less heat. | diskless workstation | diskless | both a and b | workstation | diskless workstation |
| A distributed file System typically provides types of services. | 1 | 3 | 4 | 6 | 3 |
| Which one of the following includes the types of services in distributed file System. | storage device | true file services | name | False file service | true file services |

| | | | | | |
|---|---|---|---|---|---|
| __________ deals with the allocation and management of space on a secondary storage device that is used for Storage of files in the file System. | storage services | true file services | name service | allocate memory | storage services |
| ________ provides a logical view of the storage system by providing operations for storing and retrieving data. | True file Service | shared memory | Storage Service | False file service | Storage Service |
| Most systems use magnetic disks as the device for files_______ | primary storage | secondary storage | both a and b | name service | secondary storage |
| The storage service is also known as ________ | True file Service | black service | name service | disk service | disk service |
| Several systems allocate disk space in units of fixed size blocks ,and hence the storage service is also________ in these systems. | name service | block service | service | name service | block service |
| ______is concerned with the operations on individual files such as operations for accessing and modifying the data in files and for creating and deleting files. | storage services | true file services | name service | information service | true file services |
| _____Provides a mapping between text names for files and references to file, that is file ID's. | storage services | true file services | name service | information service | name service |
| Transparency is of _____types. | 1 | 2 | 3 | 4 | 4 |
| A distributed file system normally uses file servers. | single | double | multiple | triple | multiple |

| | | | | | |
|---|---|---|---|---|---|
| Both ______and_______ should be accessible in the same way. | client and server | receiver and sender | remote and local files | encryption and decryption | remote and local files |
| _______ is not necessary, for Performance, Scalability and reliability reasons. | structure transparency | access transparency | naming transparency | transparency | structure transparency |
| The file system should automatically locate on _____and arrange for the transport of data to the client's side. | accessed file | remote file | server file | local file | accessed file |
| ____name of a file should give no hint as to where the file is located. | transparency | naming transparency | replication transparency | access transparency | naming transparency |
| If a file is replicated on multiple nodes, both the existence of multiple copies and their locations should be hidden from the clients. | Access transparency | naming transparency | replication transparency | structure transparency | replication transparency |
| The _________of a file system is usually measured as the average amount of time needed to satisfy client requests. | performance | simplicity | scalability | ease of use | performance |
| ________is inevitable that a distributed system will grow with time. | scalability | availability | simplicity | mobility | scalability |
| The commonly used criteria for file modeling is __________ | structure | Modifiability | both a and b | ease of use | both a and b |
| A file appears to the file server as an ordered sequence of________ | pixels | records | bytes | blocks | records |
| Modifiability criteria ,files are of ______types. | 3 | 2 | 4 | 8 | 2 |
| _________the processing of the client's request is performed at the server's node. | Data –caching model | Remote service model | File sharing model | data sharing | Remote service model |

| | | | | | |
|---|---|---|---|---|---|
| In the _______model, every remote file access request result in network traffic | data sharing | Remote service model | Data caching model | Information model | Remote service model |
| LRU is ______. | Least Recently Used | Least Research Used | Light Research Uniform | Least Recently Users | Least Recently Used |
| . _______When operation requires file data to be transferred across the network. | Remote Service model | Data –caching model | File-level transfer model | Block – level transfer model | File-level transfer model |
| File data transfers across the network between a client and server take place in units of file blocks. | Remote Service model | Data –caching model | File-level transfer model | Block – level transfer model | Block – level transfer model |
| _______file data transfers across the network between a client and server take place in units of bytes. | Remote service model | Byte level transfer model | Record level transfer model | Block – level transfer model | Byte level transfer model |
| ______model is suitable for use with those file models in which file contents are structured in the form of records. | Data caching model | Block-level transfer model | Byte level transfer model | Record level transfer model | Record level transfer model |
| RSS is ____ | Research Storage System | Researcher Storage System | Recent Storage System | Restoration storage system | Research Storage System |
| _______means that the name of the object should not reveal any hint as to the physical location of the object. | Location independency | Location Transparency | Naming | System | Location Transparency |
| ______means that the name of the object need not be changed when the objects location changes. | . Naming | Location Transparency | Location Independency | Model | Location Independency |

| | | | | | |
|---|---|---|---|---|---|
| A _________ system should allow many different objects to be identified by the same name. | Multiple user name | Performance | Meaningful names | Group naming | Group naming |
| Name space are managed by _______ | name servers | meaningful name | records | naming | name servers |
| The name servers that store the information about an object are called _______ | Authoritative name servers | Primitive Name servers | Several name servers | server name | Authoritative name servers |
| Name agents may be of ___________types | 2 | 3 | 4 | 5 | 2 |
| A ________can be thought of as the environment in which a name is valid. | name pair | context | structure | resource | context |

# UNIT V

## SYLLABUS

Security: Introduction – potential attacks to computer system – cryptography.

## SECURITY

The main emphasis in the design of NESL was to make parallel programming easy and portable. Algorithms are typically significantly more concise in NESL than in most other parallel programming languages. Furthermore the code closely resembles high-level pseudocode. Here is a comparison of a parallel quick sort in NESL and MPI (10 lines of code vs. 1700). Of course this comes at the cost of placing more responsibility on the compiler and runtime system for achieving good efficiency. We have found NESL very useful for teaching parallel algorithms. It has allowed us to do give out homework assignments with significantly more interesting problems than would be possible with other languages. For example here is a homework assignment on the finite-volume method for fluid flow. This involves setting up the problem using the Delaunay triangulation of an unstructured mesh, and then solving it using the conjugate gradient technique on an irregular sparse matrix.

INTRODUCTION

Assignments include finding all-closest-pairs in the plane and shortest paths in a graph. Here is a course on parallel algorithms for which we use NESL. Algorithm Experimentation: We have used NESL extensively for running experiments on algorithms. In particular it has allowed us to quickly compare the work required by various algorithms and improve the algorithms. Here are some of the algorithms we have experimented with using NESL:

- Delaunay triangulation: We have run experiments on a variety of parallel algorithms for planar Delaunay triangulation and have developed a practical variant of an algorithm of Edels brunner and Shi. This work is described in the paper Developing a practical projection-based parallel Delaunay algorithm which appears in the the Proceedings of the ACM Symposium on Computational Geometry, May 1996.

- The N-body problem: We have compared three algorithms for the N-body problem: the Barnes-Hut, Greengard's algorithm and a hybrid. All three were code in NESL and the relative costs under various assumptions were studied. This work is described in the paper A Practical Comparison of N-Body Algorithms which appears in the proceedings of the Dimacs implementation challenge workshop, October 1994.

- Graph Connectivity We have compared several algorithms for graph connectivity and derived a hybrid technique which appears very promising. This work is described in the paper A Comparison of Data-Parallel Algorithms for Connected Components which appears in the proceedings of the ACM Symposium on Parallel Algorithms and Architectures, June 1994.

- Others: Other algorithms experiments that have used NESL include a comparison of graph separators and the development of a support tree conjugate gradient technique.

Algorithm Animation: NESL is very well suited for developing animations of parallel algorithms. All the animations on the algorithm animations page are fully written in NESL as is the Pittsburgh Map server. NESL has a well developed library of window routines. Many were specifically designed with animations in mind. Also, the execution image for the animations can be quite small requiring little effort on the part of the host machine. Even though the full NESL image is large, only the intermediate code (VCODE) along with the VCODE interpreter is required to run NESL applications.

## POTENTIAL ATTACK TO COMPUTER SYSTEM

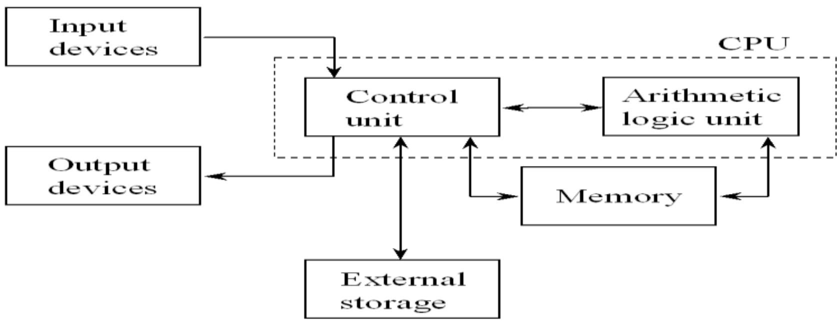
A von Neumann language is any of those programming languages that are high-level abstract isomorphic copies of von Neumann architectures. As of 2009, most current programming languages fit into this description, likely as a consequence of the extensive domination of the von Neumann computer architecture during the past 50 years[].

The differences between Fortran, C, and even Java, although considerable, are ultimately constrained by all three being based on the programming style of the von Neumann computer[citation needed]. If, for example, Java objects were all executed in parallel with asynchronous message passing and attribute-based declarative addressing, then Java would not be in the group.

The isomorphism between von Neumann programming languages and architectures is in the following manner:

- program variables ↔ computer storage cells
- control statements ↔ computer test-and-jump instructions
- assignment statements ↔ fetching, storing instructions
- expressions ↔ memory reference and arithmetic instructions

A single lecture is devoted to describing how a computer works, or "what's under the hood". The Von Neumann computer architecture model (see Figure) is introduced and examples are given for all of the components of the model. The operation of memory and disk storage is described. Having understood spreadsheets, students can make an analogy between memory addressing and spreadsheet addresses. The concept of machine language, and assembly language as a human form of machine language, is introduced. Students see how machine language is capable of only very simple operations such as moving data words to and from memory and simple arithmetic operations. The operation of an assembler follows from the discussion on machine and assembly language.




The Von Neumann computer architecture model

- The next step from assembly language is a high-level language. The development of high-level languages as application specific languages to problem solving is introduced. Here again the idea of choosing the right tool for the problem is made. it is shown that FORTRAN is intended for scientific applications, COBOL for business applications, BASIC and Pascal for education, C for systems programming, and Java originally for consumer appliance control applications. The operation of a compiler is described and hands-on laboratory exercises are conducted to familiarize students with the use of a compiler. However, before moving to programming in a high-level language, proper design methods must be covered.

## CRYPTOGRAPHY

Concurrent Pascal (also known as PASCAL-FC) was designed by Per Brinch Hansen for writing concurrent computing programs such as operating systems and real-time monitoring systems on shared memory computers.

A separate language, Sequential Pascal, is used as the language for applications programs run by the operating systems written in Concurrent Pascal. Both languages are extensions of Niklaus Wirth's Pascal, and share a common threaded code interpreter. The following describes how Concurrent Pascal differs from Wirth's Pascal.

Several constructs in Pascal were removed from Concurrent Pascal for simplicity and security: variant records

- the go to statement (and labels)
- procedures as parameters

- packed arrays
- pointer types
- file types (and associated standard input/output procedures).

These omissions make it possible to guarantee, by a combination of compile-time checks and minimal run-time checking in the threaded-code interpreter, that a program can not damage itself or another program by addressing outside its allotted space.

Concurrent Pascal includes class, monitor, and process data types. Instances of these types are declared as variables, and initialized in an init statement.

Classes and monitors are similar: both package private variables and procedures with public procedures (called procedure entries). A class instance can be used by only one process, whereas a monitor instance may be shared by processes. Monitors provide the only mechanism for interprocess communication in a Concurrent Pascal program.

Only one process can execute within a given monitor instance at a time. A built in data type, the queue, together with operations delay and continue, are used for scheduling within monitors. Each variable of type queue can hold a single process; if many processes are to be delayed in a monitor, multiple queue variables, usually organized as an array, must be provided. The single process queue variable gives a monitor complete control over medium-term scheduling, but the programmer is responsible for unblocking the correct process. A process, like a class or monitor, has local variables, procedures, and an initial statement, but has no procedure entries. The initial statement ordinarily executes forever, calling local procedures, class procedures, and monitor procedures. Processes communicate through monitor procedures. Language rules prevent deadlock by imposing a hierarchy on monitors. But nothing can prevent a monitor from erroneously forgetting to unblock a delayed process (by not calling continue) so the system can still effectively hang up through programming errors.

The configuration of processes, monitors, and classes in a Concurrent Pascal program is normally established at the start of execution, and is not changed thereafter. The communication paths between these components are established by variables passed in the init statements, since class and monitor instance variables cannot be used as procedure parameters.

## COMMUNICATING SEQUENTIAL PROCESSES

In computer science, Communicating Sequential Processes (CSP) is a formal language for describing patterns of interaction in concurrent systems.[1] It is a member of the family of mathematical theories of concurrency known as process algebras, or process calculi. CSP was highly influential in the design of the Occam programming language,[1][2] and also influenced the design of programming languages such as Limbo] and Go.

CSP was first described in a 1978 paper by C. A. R. Hoare but has since evolved substantially. CSP has been practically applied in industry as a tool for specifying and verifying the concurrent aspects of a variety of different systems, such as the T9000 Transputer, as well as a secure ecommerce system. The theory of CSP itself is also still the subject of active research, including work to increase its range of practical applicability (e.g., increasing the scale of the systems that can be tractably analyzed)

## APPLICATIONS

An early and important application of CSP was its use for specification and verification of elements of the INMOS T9000 Transputer, a complex superscalar pipelined processor designed to support large-scale multiprocessing. CSP was employed in verifying the correctness of both the processor pipeline, and the Virtual Channel Processor which managed off-chip communications for the processor.

Industrial application of CSP to software design has usually focused on dependable and safety-critical systems. For example, the Bremen Institute for Safe Systems and Daimler-Benz Aerospace modeled a fault management system and avionics interface (consisting of some 23,000 lines of code) intended for use on the International Space Station in CSP, and analyzed the model to confirm that their design was free of deadlock and livelock.The modeling and analysis process was able to uncover a number of errors that would have been difficult to detect using testing alone. Similarly, Praxis High Integrity Systems applied CSP modeling and analysis during the development of software (approximately 100,000 lines of code) for a secure smart-card Certification Authority to verify that their design was secure and free of deadlock. Praxis claims that the system has a much lower defect rate than comparable systems.

Since CSP is well-suited to modeling and analyzing systems that incorporate complex message exchanges, it has also been applied to the verification of communications and security protocols. A prominent example of this sort of application is Lowe's use of CSP and the FDR refinement-checker to discover a previously unknown attack on the Needham-Schroeder public-key authentication protocol, and then to develop a corrected protocol able to defeat the attack.

**TOOLS**

Over the years, a number of tools for analyzing and understanding systems described using CSP have been produced. Early tool implementations used a variety of machine-readable syntaxes for CSP, making input files written for different tools incompatible. However, most CSP tools have now standardized on the machine-readable dialect of CSP devised by Bryan Scattergood, sometimes referred to as $CSP_M$.[16] The $CSP_M$ dialect of CSP possesses a formally defined operational semantics, which includes an embedded functional programming language.

The most well-known CSP tool is probably Failures/Divergence Refinement 2 (FDR2), which is a commercial product developed by Formal Systems (Europe) Ltd. FDR2 is often described as a model checker, but is technically a refinement checker, in that it converts two CSP process expressions into Labelled Transition Systems (LTSs), and then determines whether one of the processes is a refinement of the other within some specified semantic model (traces, failures, or failures/divergence).[17] FDR2 applies various state-space compression algorithms to the process LTSs in order to reduce the size of the state-space that must be explored during a refinement check.

The Adelaide Refinement Checker (ARC) [18] is a CSP refinement checker developed by the Formal Modelling and Verification Group at The University of Adelaide. ARC differs from FDR2 in that it internally represents CSP processes as Ordered Binary Decision Diagrams (OBDDs), which alleviates the state explosion problem of explicit LTS representations without requiring the use of state-space compression algorithms such as those used in FDR2.The ProB project,[19] which is hosted by the Institut Informatik, Heinrich-Heine-Universität Düsseldorf, was originally created to support analysis of specifications constructed in the B method. However, it also includes support for analysis of CSP processes both through refinement checking, and LTL model-checking. ProB can also be used to verify properties of combined CSP and B specifications.

The Process Analysis Toolkit (PAT) is a CSP analysis tool developed in the School of Computing at the National University of Singapore. PAT is able to perform refinement checking, LTL model-checking, and simulation of CSP and Timed CSP processes. The PAT process language extends CSP with support for mutable shared variables, asynchronous message passing, and a variety of fairness and quantitative time related process constructs such as deadline and waituntil. The underlying design principle of the PAT process language is to combine a high-level specification language with procedural programs (e.g. an event in PAT may be a sequential program or even an external C# library call) for greater expressiveness.

Mutable shared variables and asynchronous channels provide a convenient syntactic sugar for well-known process modelling patterns used in standard CSP. The PAT syntax is similar, but not identical, to $CSP_M$. The principal differences between the PAT syntax and standard $CSP_M$ are the use of semicolons to terminate process expressions, the inclusion of syntactic sugar for variables and assignments, and the use of slightly different syntax for internal choice and parallel composition.

Occam is a concurrent programming language that builds on the Communicating Sequential Processes (CSP) process algebra, and shares many of its features. It is named after William of Ockham of Occam's Razor fame.

Occam is an imperative procedural language (such as Pascal). It was developed by David May and others at INMOS, advised by Tony Hoare, as the native programming language for their transputer microprocessors, but implementations for other platforms are available. The most widely known version is Occam 2; its programming manual was written by Steven Ericsson-Zenith and others at INMOS.

In the following examples indentation and formatting are critical for parsing the code: expressions are terminated by the end of the line, lists of expressions need to be on the same level of indentation. This feature, named the off-side rule, is also found in other languages such as Haskell and Python.

**PART-B(6 MARKS)**
**POSSIBLE QUESTIONS**

1. Explain in detail about Security and its Types.
2. Brief note on goals of computer security
3. Discuss in detail about passive attacks.
4. Discuss in detail about active attacks.
5. Explain the general architecture of cryptosystem with an example
6. What is authentication? And explain authentication Process.
7. Brief about the Key Distribution problem.

**PART-C(10 MARKS)**
**POSSIBLE QUESTIONS**

1. What is called as attacker? And explain in detail about potential attacks to computer system.
2. What is cryptography? And explain basic requirements of cryptography.
3. Discuss in detail about symmetric and asymmetric crypto systems with an key diagram.
4. Explain Potential Attack in a Computer System with an running Example.

**KARPAGAM ACADEMY OF HIGHER EDUCATION**
**Department of Computer Science**
**II M.Sc( CS) (BATCH 2017-2019)**
**Distributed Operating Systems**

| PART-A ONLINE EXAMINATIONS ONE MARK QUESTIONS |
|---|
| ONLINE EXAMINATIONS ONE MARK QUESTIONS |

| QUESTIONS | OPT 1 | OPT 2 | OPT3 | OPT4 | ANSWER |
|---|---|---|---|---|---|
| ________is a means of protecting private information against unauthorized access in those activities . | spoofing | cryptography | morphing | stegnography | cryptography |
| _______is the process of transforming an intelligible information into an unintelligible form | encryption | decryption | encoding | decoding | encryption |
| Transforming the information back from ciphertext to the plaintext is called ________. | decoding | encryption | decryption | encoding | decryption |
| When cryptography is employed for protecting information transmitted through communication channels,plaintext is also called as a ___________. | call | response | message | request | message |

| | | | | | |
|---|---|---|---|---|---|
| ________is an intruder is able to intercept ciphertext and tries to derive kd from the ciphertext. | ciphertext attack | known plain text attack | chosen plain text | plain text | ciphertext attack |
| _______attack,an intruder has cosiderable amount of both ciphertext and corresponding plaintext and tries to derive kd from them. | ciphertext attack | known plain text attack | chosen plain text | encryption | known plain text attack |
| ______attack an intruder has access to ciphertext for any plaintext of his or her choice. | chosen plain text | ciphertext attack | encryption | known plain text attack | chosen plain text |
| Two broad classes of cryptosystem is _______ and __________. | symmetric and asymmetric | decoding and encoding | enciphering and deciphering | encrypt and decrypt | symmetric and asymmetric |
| symmetric cryptosytem is either both__________and__________. | private key and public key | secret key and private key | public key and secret key | encryption key and decryption key | encryption key and decryption key |
| symmetric cryptosystem are also known as _________. | private key | shared key | shared key or private key | public key | shared key or private key |

| | | | | | |
|---|---|---|---|---|---|
| _______are useful in those situations when both encryption and decryption of information are performed by trusted subsytem. | symmetric cryptosytem | asymmetric cryptosytem | public key cryptosystem | private key cryptosytem | symmetric cryptosytem |
| ______are comptationally expensive and hence are not suitable for bulk data encryption. | private key cryptosytem | public key cryptosystem | symmetric cryptosytem | asymmetric cryptosytem | public key cryptosystem |
| A typical use of public key cryptosystem in distributed systems is for the exchange of message using a ______. | private key cryptosytem | public key cryptosystem | symmetric cryptosytem | asymmetric cryptosytem | symmetric cryptosytem |
| ________deals with how to securely supply the keys necessary to create logical channels. | symmetric key distribution | asymmetric key distribution | key distribution problem | private key cryptosytem | key distribution problem |
| KDC_____ | key distribution center | key disturb core | key distributed centre | key distributing center | key distribution center |
| __________is a single centralized KDC is used that maintain a table of secret key for each user. | centralized approach | fully distributed approach | partially distributed approach | key distributing center | centralized approach |

| | | | | | |
|---|---|---|---|---|---|
| ________is used by key distribution approach in asymmetric cryptosystem. | public key manager | private key manager | secret key manager | stegnography | public key manager |
| PKM____ | public key manager | private key manager | secret key manag | stegnography | public key manager |
| ______maintains a directory of public key of all users in the system. | private key manager | stegnography | public key manager | private key manager | public key manager |
| ____is generally trusted entity shared by all communicating users of the system. | PKM | KDC | KDD | KDM | KDC |
| Information within the system must be accessible only to authorized users _______. | privacy | authenticity | secrecy | integrity | secrecy |
| _________ deals with security the com,puter system against external factors. | internal security | external security | access control | user authentication | external security |
| _________ that are used to connect the computers are normally exposed to attacker. | communication channel | communication entities | integrity | access control | communication channel |

| | | | | | |
|---|---|---|---|---|---|
| The term_______ is commonly used to refer to a person or program trying to obtain unauthorized access | intruder | attacker | integrity of message | security | intruder |
| A ________ does not cause any harm to the system being threatened | active attack | passive attack | delay attack | attacker | passive attack |
| __________mechanism are used to prevent unauthorized reading of stored files and other processes | active attack | passive attack | access control | external security | access control |
| _________ is an intruder uses an accomplice who leaks the information to him/her | leaking | information | masquarade | intruder | leaking |
| _________ is an intruder masquerades as an authorized user or program in order to gain access to unauthorized data | leaking | information | masquarade | intruder | masquarade |
| ________is an intruder tries to draw some inference by closely and analysing the systems data. | leaking | inferencing | masquarade | intruder | inferencing |

| | | | | | |
|---|---|---|---|---|---|
| 30.program is a program that consists of clandestine code to do nasty things in addition to its usual function but apperas to be begin. | leaking | trojan horse | intruder | masquarade | trojan horse |
| ___________are more malicious than passive intruders | active intruders | virus | intruder | trojan horse | active intruders |
| A________ is a piece of code attached to a legitimate program that,infects other programs in the sysyem by replicating | computer virus | active intruder | passive intruder | attacker | computer virus |
| _________ are programs that spread from one computer to another in a network of computers. | virus | worms | intruder | hacker | worms |
| A worms program may perform destructive activities after arrival at a________ | network node | intruder | hacker | attacker | network node |
| A _______ is a program that lies dormant until some trigger condition causes it to explode | logic bomb | virus | worms | attacker | logic bomb |
| Several message have time value _______ | replay attack | delay attack | deniel attack | passive attack | delay attack |

| | | | | | |
|---|---|---|---|---|---|
| An intruder retransmits old messages that are accpted as new message by their recepients _______ | replay attack | delay attack | passive intruder | access control | replay attack |
| A _______is an information that is guaranteed to be fresh | nonce | active | passive | attacker | nonce |
| _________ are those that utilize system storage such as shared variable or files to leak information to other process | storage channel | cover channel | legitimate channel | intruder attack | storage channel |
| _________ deals with the encryption of sensitive data to prevent its comprehension and is the only practical means for protecting information | cryptography | networksecurity | attacker | intruder | cryptography |