**NAME: MWIHAKI MUGO JAMES**

**REG: AS261/1390/2014.**

**UNIT NAME.OPERATING SYSTEMS II.**

**1). DISCOUSS THE INTERCONNECTION NETWORK DESIGNS USED N DISTRIBUTED SYSTEMS**

1. **Completely connected network.**

This is a network design in which each node is connected to all other nodes in the network. It guarantee fast delivery of message from any source node to any destination node (only one link has to be traversed). Every node is connected to other every other node in the network, thus routing of the messages between nodes becomes a straight forward task.

This network design has the following characteristics

* Guarantee fast delivery of messages from one source node to any destination.
* Are expensive in terms of the number of links needed.
* The number of links is given by N(N-1)2
* The delay complexity is measured in terms of links traversed as messages are routed from any source to any destination

1. **Limited connection network**.

This network design does not provide direct link from every node to every other node in the network, but instead, communication between some nodes have to be routed through other nodes in the network. The length of the path between nodes is measured in terms of the links that have to be traversed and this path is expected to be longer compared to that of compete connected network.

The following conditions have to be met:

* Presence of pattern of interconnection between nodes.
* Presence of mechanism for routing messages around the network until they reach their destinations

1. **Linear Array Static limited connection network**. In this network design, each node is connected to its two immediate nodes. If node 1 needs to communicate with node 3, then the message from node 1 has to traverse through node 2. The worse condition is that, if node 1 has to send messages to node N, the massages has to Travers N-1 nodes before it can reach their destinations.

They are simple I their structure and have simple routing mechanisms. However, they are slow. Particularly when the number of nodes N is large.

If two nodes at the extreme ends of the linear array are connected, then the resultant network has a ring (loop) architecture.

1. **Tree Network Static Limited Network**. Here the network is a tree-like structure, where if a node at level I needs to communicate with a node at level j, were I>j, and the destination node belongs to the same root’s child sub tree, then it will have to send its message up to the tree traversing nodes at level i-1, i-2, …, j+1 until it reaches the destination node.

If a node at level I needs to communicate with another node at the same level I , it will to send its message up the tree until the message reaches the root node at level 0. The message will then have to be sent down from the root node until it reaches its destination.

The maximum debt of a binary tree system is [log], where N I the number of nodes (processors) in the network.

The network complexity is O (2k) and the time complexity is O (log).

1. Cube-Connected network. Also called HCN-Hyper Cube Network. It is defined as an underlying graph having (2n) vertices label 0 to 2n – 1, such that there is an edge between a given pair of vertices if and only if the binary representation of their a one bit dresses differs in one and only one bit.

In cube –based multiprocessor system, processing elements are positioned at the vertices of the graph. The edges of the graph represents the point-to-point communication links between processors.

**2). STATE AND DISCRIBE NINE TYPES OF DISTRIBUTED SYSTEMS TRANSPARENCIES.**

1. **Access Transparency**.

Clients should be unaware of the distribution of the files. The files could be present on a totally different set of servers which are physically distant apart and a single set of operations should be provided to access these remote as well as the local files. Applications written for the local file should be able to be executed even for the remote files.

The examples illustrating this property are the File system in

Network File System (NFS), SQL queries, and Navigation of the web.

**II). Location Transparency.**

Clients should see a uniform file name space. Files or groups of files may be relocated without changing their pathnames.

A location transparent name contains no information about the named object’s physical location. This property is important to support the movement of the resources and the availability of services. The location and access transparencies together are sometimes referred as Network transparency.

The examples are File system in NFS and the pages of the web.

**III). Concurrency Transparency.**

Users and Applications should be able to access shared data or objects without interference between each other. This requires very complex mechanisms in a distributed system, since there exists true concurrency rather than the simulated concurrency of a central system. The shared objects are accessed simultaneously.

The concurrency control and its implementation is a hard task. The examples

Are NFS, Automatic Teller machine (ATM) network’s 6306 Advanced Operating Systems?

**IV).Replication Transparency.** This kind of transparency should be mainly incorporated for the distributed file systems, which replicate the data at two or more sites for more reliability. The client generally should not be aware that a replicated copy of the data exists. The clients should also expect operations to return only one set of values.

The examples are Distributed DBMS and

Mirroring of Web pages.

**V).Failure Transparency.**

Enables the concealment of faults, allowing

User and application programs to complete their tasks despite the failure of

Hardware or software components. Fault tolerance is provided by the mechanisms that relate to access transparency. The distributed system are more prone to failures as any of the component may fail which may lead

To degraded service or the total absence of that service. As the intricacies

Are hidden the distinction between a failed and a slow running process is difficult. Examples are Database Management Systems.

**VI). Migration Transparency.**

This transparency allows the user to be unaware of the movement of information or processes within a system without affecting the operations of the users and the applications that are running. This mechanism allows for the load balancing of any particular client, which might be overloaded. The systems that implement this transparency are NFS and Web pages.

**VII).Performance Transparency**. Allows the system to be reconfigured to

Improve the performance as the load varies.

**VII. Scaling Transparency**.

A system should be able to grow without affecting application algorithms. Graceful growth and evolution is an important requirement for most enterprises. A system should also be capable of scaling down to small environments where required, and be space and/or time efficient as required. The best-distributed system example implementing this transparency is the World Wide Web.

**3). DEFINE KERNEL AND DISCUSS ITS ROLES.**

**kernel** is the central component of most computer **operating systems**; it is a bridge between applications and the actual data processing done at the hardware level. The **kernel's** responsibilities include managing the **system's** resources (the communication between hardware and software components).

**Roles of kernel:**

* Process management the kernel allocate resources to **processes**, enable **processes** to share and exchange information, protect the resources of each **process** from other **processes** and enable synchronization among **processes.**
* Device management. Kernel allows users to view and control the hardware attached to the **computer**. When a piece of hardware is not working, the offending hardware is highlighted for the user to deal with.
* Memory management. Kernel provide ways to dynamically allocate portions of **memory** to programs at their request, and free it for reuse when no longer needed.
* Interrupt handling. Kernel causes an external event to the currently executing process that causes a change in the normal flow of instruction execution; usually generated by hardware devices external to the CPU.
* I/O communication. Kernel makes the communication between an information processing

4).**DISCUSS DIFFERENT TYPES OF KERNEL.**

There are of two types of Kernels:

* Monolithic
* Micro Kernel

**1 Monolithic Kernels**  
 All the basic system services like process and memory management, interrupt handling etc. were packaged into a single module in kernel space. This type of architecture led to some serious drawbacks like

1) Size of kernel, which was huge.

2)Poor maintainability, which means bug fixing or addition of new features resulted in recompilation of the whole kernel which could consume hours  
  
In a modern day approach to monolithic architecture, the kernel consists of different modules which can be dynamically loaded and un-loaded. This modular approach allows easy extension of OS's capabilities. With this approach, maintainability of kernel became very easy as only the concerned module needs to be loaded and unloaded every time there is a change or bug fix in a particular module. Linux follows the monolithic modular approach

**2 Microkernels**  
  
this architecture majorly caters to the problem of ever growing size of kernel code which we could not control in the monolithic approach. This architecture allows some basic services like device driver management, protocol stack, file system etc. to run in user space. This reduces the kernel code size and also increases the security and stability of OS as we have the bare minimum code running in kernel. So, if suppose a basic service like network service crashes due to buffer overflow, then only the networking service's memory would be corrupted, leaving the rest of the system still functional.   
  
In this architecture, all the basic OS services which are made part of user space are made to run as servers which are used by other programs in the system through inter process communication (IPC).

**5). DIFFERENTIATE BETWEEN A THREADS AND A PROCESS**

**A Thread** is the smallest sequence of programmed instructions that can be managed independently by a scheduler, which is typically a part of the operating system, Whilea **Process-** is an instance of a computer program that is being executed. It contains the program code and its current activity. Depending on the operating system (OS), a process may be made up of multiple threads of execution that execute instructions concurrently.

**6): WHY PROCESS AND WHY THREADS.**

**Why a threads:** Threads makes processes to be: Responsive, busy and cooperative I n execution.

**Why a processes**: Processes simplify the executions of a task.