Quiz 1
Operating Systems and Networks
IIIT Hyderabad
Time: 30 minutes; Max. Marks: 20

Note: Give a brief and correct answer. **Mention your name and roll number on the answer sheet. Answer the question after writing the question in the answer sheet.**

1. Is it possible to build an operating system without employing the notion of "caching". Discuss.

2. Elaborate the issue of "cache coherency".

3. Like any system, the operating system has Input and Output. Based on the topics you have studied, what constitutes as Input and Output of an operating system?

4. How spooling has improved the CPU utilization?

5. Operating system is called as a "Interrupt Controller". Justify.

Answers

Q.1 - Caching is storing data in a separate disk (very fast-speed disk). The data which is to be used many times results in a waste of time if it is on hard disk, but storing the data in cache reduces this time wastage.

∴ Caching is only used to speed things up; it is not necessary for an operating system to function.

Q2. - In a multiprocessor system, data inconsistency may occur among adjacent levels or within the same level of the memory hierarchy. For example, the cache and the main memory may have inconsistent copies of the same object.

As multiple processors operate in parallel, and independently multiple caches may possess different copies of the same memory block, this creates cache coherence problem.

Q3. - Input refers to the signals or instructions sent to the operating system by the user. Output refers to the signals sent out from the computer/operating system. This term is also known as I/O operations, which references the input and output actions.

Q4. - SPOOL stands for Simultaneous Peripheral Operation OnLine. In this, more than one I/O operation can be performed simultaneously while the CPU is executing some process.

Since there is no interaction of I/O devices with CPU, so the CPU need not wait for the I/O operation to take place. The I/O operations take a large amount of time. The CPU is kept busy most of the time which increases CPU utilization.

Q.5 - An interrupt is a signal emitted by hardware or software when a process or an event needs immediate attention. It alerts the processor to a high-priority process requiring interruption of the current working process. Since the OS contains an interrupt handler which prioritises and manages interrupts, it is called often deemed as an interrupt controller.

Quiz 2
Operating Systems and Networks
IIIT Hyderabad
Time: 30 minutes; Max. Marks: 24

Note: Give a brief and correct answer. **Mention your name and roll number on the answer sheet. Answer the question after writing the question in the answer sheet.**

1. What do you understand by the term "long term scheduler"?
2. What are the strengths and weaknesses of the two models of inter-process communication?
3. Consider a multiprocessor system and a multi threaded program using many-to-many threading model. Let the number of user-level threads in the program be more than the number of processors in the system. Discuss the performance implications of the following: "The number of kernel threads allocated to the program is less than the number of processors".
4. Do you agree that "multithreading framework improves both Concurrency and Parallelism on multicore system?" Justify your answer.
5. Suppose a CPU scheduling algorithm favors those programs that have used little processor time in the recent past. Explain why this algorithm favors I/O-bound processes and yet does not permanently deny processor time to CPU-bound processes ?
6. What is the purpose of thread pools?

Answers

Q.1 - A long-term scheduler is a scheduler that is responsible for bringing processes from the JOB queue (or secondary memory) into the READY queue (or main memory). In other words, a long-term scheduler determines which programs will enter into the RAM for processing by the CPU.

Q.2 -
• Message passing
- strengths: program structures better separated, dangerous operations firewalled
- weaknesses: message passes more slowly, for symmetrical copy operations are to be made

• Shared memory
- strengths: fast and direct
- weaknesses: unexpected behavior when unauthentic programs wrongly accesses the shared memory segments

Q.3 - All the processors/kernel threads are not fully utilised. Some of the kernel threads will not have any work. So, the system is underutilised.

Q.4 - Concurrency is about activities that have no clear temporal ordering. If the hardware supports it, they can be done in parallel. Traditionally multi-threading is almost synonymous with concurrency. And both of them only become parallel if the hardware supports it.

So having a multicore system does, in fact, 'improve' or allow both Concurrency and Parallelism.

Q.5 - Suppose the ready queue contains both CPU-bound and I/O-bound processes. If the CPU gives priority to 1/O bound process, it finishes 1/O bound processes quickly and the remaining time is used to process CPU-bound processes. In this way, both 1/O utilisation and CPU-utilisation can be improved. Otherwise imagine a situation, if a processor gives priority to processes which consume more CPU time in the past. In such a case 1/0 processes do not get CPU time if there are CPU-bound processes in the ready queue.

Q.3 - The concept of thread pools is used to increase the efficiency of the operating system. The idea is to create a number of threads at system startup and place them in a pool, where they sit and wait for work. If the pool has no available thread, the server waits until one is free. It is faster to service with an existing thread than waiting to create a new thread. A thread pool limits the number of threads at any point which is important on systems that can not support a large number of concurrent threads.

Quiz 3

Operating Systems and Networks

IIIT Hyderabad

Time: 30 minutes; Max. Marks: 24

Note: Give a brief and correct answer. **Mention your name and roll number on the answer sheet. Answer the question after writing the FIRST 10 WORDS OF THE QUESTION ALONG WITH THE QUESTON NUMBER in the answer sheet.**

1. In the UNIX system, directory is also treated as a file and there is a minimal difference between file and directory. What is the advantage of keeping minimal difference.
2. Describe the scheduling algorithm followed in UNIX.
3. What are the advantages and the problems of deferred block I/O system in UNIX.
4. Consider the following levels in a hypothetical OS. Take any two operations and discuss the corresponding differences at each level.

LEVEL	NAME	OBJECTS	OPERATIONS
Level 11	Devices (access to external devices)	Printers, displays, and key boards	Create, destroy, open, close, read, write
Level 10	File system	Files	Create, destroy, open, close, read, write

5.There are five levels in THE operating system.. Discuss the advantages of hierarchical organization considering the sample functions or operations of level 0 and level 1. What would have happened, if the services of level 0 and level 1 would have been written in a single level ?

6. Discuss basic problem faced by Dijkstra when he started designing the operating system. Explain how he has employed “probe instructions” and "interrupts" in OS design.

Solutions

1. Directory is treated as a file:
→ Disk I/O follows the same routing as files, so it is fast. Operations can be performed with read/write systems calls which are fast.
→ Directories are just files with a few extra characteristics. Permission, paths, modification is the same.
→ Mounting a directory becomes easy because we just have to load the directory’s file descriptor.

2. Scheduling in UNIX
UNIX uses Multilevel Feedback Queues. All runnable processes are assigned a scheduling priority that determines which queue they are places in. Priorities are dynamically adjusted and all queues use round robin.
When a non running process with a higher priority becomes available, it immediately preempts the current running process if the current process is in user mode. Otherwise, it switches when the current process exits the kernel.
UNIX uses 'niceness', which ranges from -20 (highest priority) to 20 (least priority). Default = 0

3. Deferred Block I/O System
Advantages -
On a read request, the cache is searched for the desired block. If the block is found, the data are made available to the requester without any physical I/O.
The correct buffer is found and relabeled if necessary. The write is performed simply by marking the buffer as ''dirty.'' The physical I/O is then deferred until the buffer is renamed. All these factors make I/O fast.

Disadvantages -
The algorithm is asynchronous, which makes error reporting difficult. It also delays writes. If the system stops unexpectedly, it is almost certain that there is a lot of logically complete, but physically incomplete, I/O in the buffers. Also, the associativity in the buffers can alter the physical I/O sequence from that of the logical I/O sequence. This means that there are times when data structures on disk are inconsistent, even though the software is careful to perform I/O in the correct order.

4.
Read vs Write in Devices (disk drivers) -
Disk drivers are implemented with a queue of transaction records. Each record holds a read/write flag, a primary memory address, a secondary memory address, and a transfer byte count.
The devices are treated as files and read and write operations are performed with their respective file descriptors.

Read vs. Write in File System -
Each file definition is a 64-byte structure, called an i-node. The system maintains a table of all active i-nodes. As a new file is accessed, the system locates the corresponding i-node, allocates an i-node table entry, and reads the i-node into primary memory.

A pointer to the file table entry is placed in the system data segment for the process. **create** first creates a new i-node entry, writes the i-number into a directory, and then builds the same structure as for an **open**. **read** and **write** just access the i-node entry as described above. seek simply manipulates the I/O pointer.

5. 6. From paper :((

Quiz 3

Operating Systems and Networks

IIIT Hyderabad

Time: 30 minutes; Max. Marks: 24

Note: Give a brief and correct answer. **Mention your name and roll number on the answer sheet. Answer the question after writing the the FIRST 10 WORDS OF THE QUESTION ALONG WITH THE QUESTON NUMBER in the answer sheet.**

1. Discuss how the notions of hierarchy and information hiding are helpful in easing the development of operating system?

2. There are five levels in THE operating system.. Discuss the advantages of hierarchical organization considering the sample functions or operations of level 0 and level 1. What would have happened, if the services of level 0 and level 1 would have been written in a single level ?

3. What motivated Dijkstra to introduce semaphores?

4. Explain the context of "probe instructions" in Dijkstra's paper entitled "My recollections of OS design". List the problems caused by probe instructions.

5. Discuss the difference between block I/O system and character I/O system in UNIX.

6. Consider the following levels in a hypothetical OS. Take any two operations and discuss the corresponding differences at each level.

LEVEL	NAME	OBJECTS	OPERATIONS
Level 11	Devices (access to external devices)	Printers, displays, and key boards	Create, destroy, open, close, read, write
Level 10	File system	Files	Create, destroy, open, close, read, write

1. The notion of hierarchy makes OS development easier. The OS is layered into multiple layers, and layer T can only access the protocols/ system calls/ functions provided by layer T – 1. This makes it more secure as applications cannot access the device's hardware directly. The design is modular, and debugging can be done layer by layer.

Information hiding -
Developers working on layer T need not know the exact implementation details of layer T-1. They only need to know the services provided by layer T-1 and how they work. This is known as Data Abstraction.
The operating system abstracts the machine part of computer system in terms of simple services by hiding the details of the machine (hardware).

Also, with information hiding, designers can protect themselves from extensive reprogramming if the hardware or some part of the software changes: the change affects only the small portion of the software interfacing directly with that system component. This principle has been extended from isolated subsystems to an entire operating system.

Q 2,3,4 from Papers

Q 5
Block IO System -
The model block I/O device consists of randomly addressed, secondary memory blocks of 512 bytes each. The blocks are uniformly addressed 0, 1, . . . up to the size of the device. The block device driver has the job of emulating this model on a physical device.
Read requests are handled through a buffer pool, which constitutes a data cache for the block devices. If the block is found, the data are made available to the requester without any physical I/O. If not found, then cache is updated.
The write is performed simply by marking the buffer as ‘‘dirty.’’

Character IO System -
The character I/O system consists of all devices that do not fall into the block I/O model. This includes the ‘‘classical’’ character devices such as communications lines, paper tape, and line printers. It also includes magnetic tape and disks when they are not used in a stereotyped way, for example, 80- byte physical records on tape and track-at-a-time disk copies.
I/O requests from the user are sent to the device driver essentially unaltered.

Q6. done in 3-1 quiz

Quiz 4
Operating Systems and Networks
IIIT Hyderabad
Time: 30 minutes; Max. Marks: 24

1.Explain how FCFS CPU scheduler penalizes I/O bound processes
Ans: I/O bound processes have a short burst cycle. If there is a long CPU bound process, then the I/O bound process has to wait to complete a long CPU burst. In this way, FCFS CPU scheduler penalizes I/O bound processes.

2.Consider a variation of round robin we will call progressive round robin. In progressive round-robin, each process has its own time quantum. This starts out at 50ms, and increases by 50 ms each time it goes through the round-robin queue. Give the advantages and disadvantages of this variant over ordinary-round robin with respect to the users of batch jobs and interactive jobs.
Ans: Ordinary round robin: Independent of burst time, the ordinary round robin gives equal priority to all kinds of jobs . As a result, the interactive processes which have burst time less than time quantum are processes with good response time. A process is guaranteed to get the CPU slot after (time quantum* (n-1)). The interactive processes receive higher priority than the batch processes. The performance of the batch process suffers, if more interactive processes start arriving. This is unfair for the batch jobs.
Progressive round robin: In this scheme, the processes are processes like FIFO. As time quantum increases, the batch job consumes CPU time. As a result, the interactive processes have to wait longer as compared to ordinary round robin. Batch processes receive fair treatment as compared to ordinary round robin. Negative point is that one batch job with a long burst time may force everyone to wait resulting into convoy effect. As one can noe, the convoy effect is not possible in ordinary round robin

3.Explain the application of priority inheritance protocol
Ans: Priority Inheritance Protocol (PIP) is a technique which is used for sharing critical resources among different tasks. This allows the sharing of critical resources among different groups without the occurrence of unbounded priority inversions.
Advantages of PIP :
Priority Inheritance protocol has the following advantages:

- It allows the different priority tasks to share the critical resources.
- The most prominent advantage with Priority Inheritance Protocol is that it avoids the unbounded priority inversion.

Disadvantages of PIP :

Priority Inheritance Protocol has two major problems which may occur:
Deadlock – There is a possibility of deadlock in the priority inheritance protocol.
For example, there are two tasks T1 and T2. Suppose T1 has a higher priority than T2. T2 starts running first and holds the critical resource CR2. After that, T1 arrives and preempts T2. T1 holds critical resource CR1 and also tries to hold CR2 which is held by T2. Now T1 blocks and T2 inherits the priority of T1 according to PIP. T2 starts execution and now T2 tries to hold CR1 which is held by T1. Thus, both T1 and T2 are deadlocked. Chain

Blocking – When a task goes through priority inversion each time it needs a resource then this process is called chain blocking.
For example, there are two tasks T1 and T2. Suppose T1 has the higher priority than T2. T2 holds the critical resource CR1 and CR2. T1 arrives and requests for CR1. T2 undergoes the priority inversion according to PIP. Now, T1 request CR2, again T2 goes for priority inversion according to PIP. Hence, multiple priority inversion to hold the critical resource leads to chain blocking.

4.Identify whether the following statement is TRUE or FALSE. If the statement is FALSE, correct it and justify the corrected sentence. If the statement is TRUE, justify it. Restrict the justification to a few sentences. "As compared to CPU-bound process, OS gives higher priority for I/O bound process."
Ans: TRUE
I/O-bound programs have the property of performing only a small amount of computation before performing I/O. Such programs typically do not use up their entire CPU quantum. CPU-bound programs, on the other hand, use their entire quantum without performing any blocking I/O operations. Consequently, one could make better use of the computer's resources by giving higher priority to I/O-bound programs and allow them to execute ahead of the CPU-bound programs.

Quiz 5
Operating Systems and Networks
IIIT Hyderabad
Time: 30 minutes; Max. Marks: 24

1.An old bridge on a busy highway is too narrow to permit two-way traffic, so one-way traffic is to be implemented on the bridge by alternatively permitting vehicles travelling in opposite direction to use the bridge. The following rules are formulated for use of bridge:
(a) Any time the bridge is used by vehicle(s) travelling in one direction only.
(b) If the vehicles are waiting to cross the bridge at both ends, only one vehicle from one end is allowed to cross the bridge before a vehicle from the other end starts crossing the bridge.
(c) If no vehicles are waiting at one end, then any number of vehicles from the other end are permitted to cross the bridge.
Develop a program (Pseudocode) to synchronize vehicles with semaphores.
Ans:
Concurrent System with Semapohores:

- Semaphores are used to ensure mutually exclusive execution of a critical section (as locks do) and to enter the critical section (bridge in this case) so it will first execute sem_wait(with some variable free) which means that if the bridge is free then the vehicles at the end will be allowed , since at one end only vehicles are present then they can directly go and whenever they enter, we make the variable free set to 0 - as the other end vehicles are not allowed and this is set by hardware implementation itself and once the vehicles are not allowed and will sem_post(&free) which implies the free is set to 1. So,one sided waiting is directly allowed as sem_wait(&free) doesn't restrict in this case.
- Consider the other case, when two vehicles arrived at the same time,then the vehicles that has first executed sem_wait(&free) will go because till now no vehicle is there and hence the vehicles at other end should wait because free variable is now set to zero, and sem_wait(&free) doesn't allow the other vehicles and whenever it completed crossing the bridge sem_post(&free) will be implemented, allowing the vehicles from other end. Now, vehicles at another end would come, and they follow the above implementation.

Semaphores in code:
num_waiting_north : Used for indicating the number of vehicles waiting in the north direction
num_waiting_south : Used for indicating the number of vehicles waiting in the south direction
free: for holding the bridge for entering of vehicles and stopping vehicles

Quiz - 6

3. Given memory partitions (holes) of 100K, 500K, 200K, 300K and 600 K (in order), how would each of the First-fit, Best-fit, and Worst-fit algorithms place processes of 212K, 417K,112K and 426K (in order) ? Which algorithm makes efficient use of memory?

Solution:
First-Fit:
212K is put in 500K partition.
417K is put in 600K partition.
112K is put in 288K partition (new partition 288K = 500K - 212K).
426K must wait.

Best-Fit:
212K is put in 300K partition.
417K is put in 500K partition.
112K is put in 200K partition.
426K is put in 600K partition.

Worst-Fit:
212K is put in 600K partition.
417K is put in 500K partition.
112K is put in 388K partition.
426K must wait.

In this example, Best-Fit turns out to be the best.

4. Assume a page size of 4Kbytes and that a page table entry takes 4 bytes, how many levels of page tables would be required to map a 64-bit address space, if the top level page fits into a single page?

Solution:

(a) Physical memory size = 2^{64} bytes

Page size = 4 Kbytes

So, number of pages = $2^{64}/2^{12} = 2^{52}$

So, a page table entry should have atleast 52 bits.

Rounding to the nearest power of two, the answer is 64.

(b) According to our above result, a page table entry needs 64 bits = 8 bytes.

So, number of entries in one level of page table is 4 KB$/$ 8B $= 2^9$

So the first level can address 2^9 x $2^{12} = 2^{21}$ bytes

Remaining addresses $= 2^{64}/2^{21} = 2^{43}$

For this, we need $43/9 = 5$ more levels.

Therefore, the required levels of page tables is 5 + 1 $= 6$.

2. Explain the problems if you swap a process with pending I/Os.

Solution: Swapping is constrained by other factors as well. If we want to swap a process, we must be sure that it is completely idle. Of particular concern is any pending I/O. A process may be waiting for an I/O operation when we want to swap that process to free up memory. However, if the I/O is asynchronously accessing the user memory for I/O buffers, then the process cannot be swapped. Assume that the I/O operation is queued because the device is busy. If we were to swap out process P1 and swap in process P2, the I/O operation might then attempt to use memory that now belongs to process P2. There are two main solutions to this problem: never swap a process with pending I/O, or execute I/O operations only into operating-system buffers. Transfers between operating-system buffers and process memory then occur only when the process is swapped in. Note that this **double buffering** itself adds overhead. We now need to copy the data again, from kernel memory to user memory, before the user process can access it.

1. Developments in operating systems have generally occurred in an evolutionary rather than revolutionary fashion. For the following transition, describe the primary motivations of operating systems designers that led them to produce the new type of system from the old**: Contiguous storage allocation systems to non-contiguous storage allocation systems**.

Solution Non-Contiguous memory allocation is basically a method on the contrary to contiguous allocation method, allocates the memory space present in different locations to the process as per its requirements. As all the available memory space is in a distributed pattern so the freely available memory space is also scattered here and there. This technique of memory allocation helps to reduce the wastage of memory.

- Only External fragmentation occurs in Non-Contiguous memory allocation method.
- No memory wastage is there.
- In non-contiguous memory allocation, swapped-in processes can be arranged in any place in the memory.
- It is of five types: Paging, Multilevel Paging, Inverted Paging, Segmentation, Segmented Paging

Code Implementation:

```
Monitor_bridge{
        Semaphore num_waiting_north=0;
        Semaphore num_waiting_south=0;
        Int on_bridge=0;
        Condition free=0;
        Int prev=0;

        Void enterbridge_north(){
                Num_waiting_north++;
                while(on_bridge || (prev==0 && num_waiting_south>0){
                        free.wait();
                        Num_waiting_north–;
                prev=0;
                }
                }

                Void exit_bridge(){
                        on_bridge=0;
                        free.broadcast();
                }

                Void enter_bridge_south(){
                Num_waiting_south++;
                while(on_bridge || (prev==0 && num_waiting_north>0){
                        free.wait();
                        Num_waiting_south–;
                prev=1;
                }
                }

                Void exit_bridge(){
                        on_bridge=0;
                        free.broadcast();
                }
}
```

2.Compare the deadlock prevention algorithms by preventing circular-wait with the deadlock avoidance algorithm (i.e., the banker's algorithm) with respect to the following: runtime overheads and system throughput

Ans:

In the banker's algorithm we make sure that every process is in a safe sequence. For this we have to compare every process and its allocated resources, maximum resources and available resources. This contributes to runtime overhead as we

compare the resources after every allocation. But in preventing circular-wait, the system throughput takes a hit as we only allocate processes and resources in increasing order.

3.Identify whether the following statement is TRUE or FALSE. If the statement is FALSE, correct it and justify the corrected sentence. If the statement is TRUE, justify it. Restrict the justification to a few sentences: “Deadlock prevention protocols improve CPU utilization”

Ans:

Deadlock Prevention

Deadlock prevention algorithms ensure that at least one of the necessary conditions (Mutual exclusion, hold and wait, no preemption and circular wait) does not hold true. However most prevention algorithms have poor resource utilization, and hence result in reduced throughputs.

Mutual Exclusion

It is not always possible to prevent deadlock by preventing mutual exclusion (making all resources shareable) as certain resources cannot be shared safely.

Hold and Wait

We will see two approaches, but both have their disadvantages. A resource can get all required resources before it starts execution. This will avoid deadlock, but will result in reduced throughputs as resources are held by processes even when they are not needed. They could have been used by other processes during this time. Second approach is to request for a resource only when it is not holding any other resource. This may result in a starvation as all required resources might not be available freely always.

No preemption

We will see two approaches here. If a process requests for a resource which is held by another waiting resource, then the resource may be preempted from the other waiting resource. In the second approach, if a process requests for a resource which is not readily available, all other resources that it holds are preempted. The challenge here is that the resources can be preempted only if we can save the current state and processes could be restarted later from the saved state.

Circular wait

To avoid circular wait, resources may be ordered and we can ensure that each process can request resources only in an increasing order of these numbers. The algorithm may itself increase complexity and may also lead to poor resource utilization.

Quiz 7
Operating Systems and Networks
IIIT Hyderabad
Time: 30 minutes; Max. Marks: 24

Note: Give a brief and correct answer. **Mention your name and roll number on the answer sheet. Answer the question after writing the FIRST 10 WORDS OF THE QUESTION ALONG WITH THE QUESTION NUMBER in the answer sheet.**

1. Briefly explain the importance of "quality of service" and "admission control". [3]

2. Identify whether the following statements are TRUE or FALSE. If the statement is FALSE, correct it and justify the corrected sentence. If the statement is TRUE, justify it. Restrict the justification to a few (less than five) sentences. "In UNIX, the system administrator (root) can know the passwords of users". [3]

3. Briefly explain the following consistency semantics with positive and negative points. (i) UNIX semantics (ii) session semantics. [6]

4. How RAID 4 is better than RAID 3? Also, compare RAID 4 and RAID 5 [6]

5. Define the word "quality of service". Discuss what techniques could be used to meet quality-of service requirements for multimedia applications for the following components of a system: (a) Process scheduler (b) Disk scheduler (c) Memory manager [6]

Answers

Q1. Quality of service: The main benefit of QoS is that it ensures the availability of an organization's network and the applications that run on that network. It provides the safe, efficient transfer of data over that network.
Admission control: Admission control (AC) is used to manage incoming traffic and to prevent network congestion.

Q2. The given statement is False. In UNIX, the system administrator (root) cannot know the passwords of users. They are hashed or otherwise encrypted through a variety of algorithms.

Q3. (i) Unix Semantics
The file systems in UNIX uses following consistency semantics –

- **The file which the user is going to write will be visible to all users who are sharing that file at that time.**
- **There is one mode in UNIX semantics to share files via the sharing pointer of current location. But it will affect all other sharing users.**

(ii) Session Semantics
The file system in Andrew uses the following consistency semantics.

- **The file which the user is going to be writing will not be visible to all users who are sharing that file at that time.**

- **After closing a file, changes done to that file by the user are only visible only in sessions starting later. If a changed file is already open by another user, then changes will not be visible to that user.**

Q4. RAID 4 is very similar to RAID 3. The main difference is the way of sharing data. They are divided into blocks (16, 32, 64, or 128 kB) and written on disks – similar to RAID 0. For each row of written data, any recorded block is written on a parity disk. In short, this means that RAID 4 does not stripe data at the block level, but it uses byte levels for striping (block-level striping with a dedicated parity disk).
There are also similarities in relation to RAID 5, but it confines all parity data to a single drive. RAID 4 does not use distributed parity.

Q5. Quality of service (QoS) is the description or measurement of the overall performance of a service, such as a telephony or computer network, or a cloud computing service, particularly the performance seen by the users of the network.
The process scheduler can use the rate monotonic scheduling algorithm to guarantee that the processor is utilized for meeting the quality of service requirements in a timely manner. In this scheme, processes are modeled to be periodic and require a fixed processing time every time they are scheduled. The disk scheduler needs to schedule requests in a timely manner and also minimize the movement of the disk head in order to minimize disk seek overheads. One option is to use a technique that combines the disk SCAN technique with the earliest-deadline-first strategy. Tasks are processed in the order of deadlines. When a batch of requests have the same or related deadlines, then the SCAN technique is used to satisfy the batch of requests. The memory management needs to ensure that unexpected page faults do not occur during playback of media files. This can be guaranteed if the required pages are swapped into physical memory before the multimedia application is scheduled.

Quiz 8
Operating Systems and Networks
IIIT Hyderabad
Time: 30 minutes; Max. Marks: 24

Note: Give a brief and correct answer. **Mention your name and roll number on the answer sheet. Answer the question after writing the FIRST 10 WORDS OF THE QUESTION ALONG WITH THE QUESTON NUMBER in the answer sheet.**

1. What is the difference between distributed system and computer network?[2]

2. What is the purpose of communication satellites? [2]

3. How Fourier analysis will help to reduce the bandwidth? [5]

4. What is the purpose of multiplexing? What will happen if you do not use multiplexing?[5]

5. Explain how the notion of piggybacking is employed by datalike layer? What is the issue with piggybacking?[5]

6. Why Nonpersistent CSMA Protocol gives better performance over 1-Persistent CSMA Protocol? Discuss.[5]

Que 1:

N Network Operating System.

Nd(difference) Distributed Operating System

1. Network Operating System's main objective is to provide the local services to remote client.
1. 1d- Distributed Operating System's main objective is to manage the hardware resources.
2. 2. In Network Operating System, Communication takes place on the basis of files. 2d In Distributed Operating System, Communication takes place on the basis of messages and shared memory.

Que 2: Communication satellites are designed to relay several, or more usually many, signals simultaneously. In some cases there may be a separate transponder for each carrier; this is typical of broadcasting satellites and of satellites used for distributing television signals to terrestrial broadcasting stations. More usually, each transponder will relay, not one carrier, but several or many. This is called 'multiple access'.

Que 3:

Any electromagnetic signal, whether it is analog or digital, is generally composed of a range of different frequencies, with each frequency component having a specific weightage in the overall value of the signal, at any instant of time. Given the time domain representation of a signal, Fourier Analysis helps us in finding the different frequency components of the signal, along with their respective weightages. Fouries analysis also helps in the reverse process, namely, if the frequency components of an electromagnetic signal are known, along with their weights, then it enables us to get the time domain representation of the signal.

Thus enable us to send shorter data thus reducing the bandwidth.

Que 4:

Multiplexing basically involves taking multiple signals and combining them into one signal for transmission over a single medium, such as a telephone line. The input signals can be either analog or digital. The purpose of multiplexing is to enable signals to be transmitted more efficiently over a given communication channel, thereby decreasing transmission costs.

A device called a multiplexer (often shortened to "mux") combines the input signals into one signal. When the multiplexed signal needs to be separated into its component signals (for example, when your email is to be delivered to its destination), a device called a demultiplexer (or "demux") is used.

Que 5:

Assume that A and B are users. Then the data frames from A to B are interconnected with the acknowledgement from A to B. and can be identified as a data frame or acknowledgement by checking the sort field in the header of the received frame.

One more improvement can be made. When a data frame arrives, the receiver waits does not send the control frame (acknowledgement) back immediately. The receiver waits until its network layer moves to the next data packet.

Acknowledgement is associated with this outgoing data frame. Thus the acknowledgement travels along with the next data frame.

Definition of Piggybacking :
This technique in which the outgoing acknowledgement is delayed temporarily is called piggybacking.

Disadvantages of piggybacking :

The disadvantage of piggybacking is the additional complexity.
If the data link layer waits long before transmitting the acknowledgement (block the ACK for some time), the frame will rebroadcast.
NOTE – To avoid the delay and rebroadcast of frame transmission, piggybacking uses a very short duration timer.

Que 6:

In Non-persistent CSMA, station that has frames to send only senses for channel. In the case of an idle channel, it will send frames immediately to that channel. In case when channel is found busy, it will wait for a fixed amount of time and again sense for state of station to be idle or busy. In this method, station does not immediately sense for channel for only purpose of capturing it when it detects end of previous transmission. This method reduces the chances of a collision but reduces efficiency of network.

QUIJ - 9

1. Explain why FIFO packet scheduling algorithm fails to provide good service for diverse applications?

Ans -
FIFO routers usually drop newly arriving packets when the queue is full. Since the newly arrived packet would have been placed at the end of the queue, this behavior is called tail drop.

2. Explain the purpose of leaky bucket algorithm? What will happen, if you do not implement leaky bucket algorithm?

Ans -
The leaky bucket algorithm is a "traffic shaping" algorithm to reduce the load the transport layer places on the network layer and reduce congestion in the network. Commonly used in asynchronous transfer mode (ATM) networks, the algorithm provides a way to temporarily store a variable number of requests and then organize them into a set-rate output of packets.
Network congestion and traffic shaping
Traffic congestion is a common problem in all networks. When too many packets flow through a network, packet delays or packet losses can occur, both of which degrade the network's performance. This situation is known as congestion.
Congestion in a network often happens when the traffic is bursty. In the seven-layer OSI model for networking system communications, the network and transport layers are layer 3 and layer 4, respectively. The two layers are jointly responsible for handling network congestion, which they do by "shaping" the traffic.
Chart showing the Open Systems Interconnection (OSI) network model
The 7 layers of the OSI model

3. What are the issues of implementing "Connection Establishment" in the transport protocol? Explain the justification of employing three-way hand shake.

Ans –
Problem: Delayed and Duplicate Packets
Imagine a network that is so congested that acknowledgements hardly ever get back in time and each packet times out and is retransmitted two or three or more times. Suppose that the network uses datagrams inside and that every packet follows a different route. Some of the packets might get stuck in a traffic jam inside the network and take a long time to arrive. That is, they may be delayed in the network and pop out much later, when the sender thought that they had been lost.
The worst possible nightmare is as follows. A user establishes a connection with a bank, sends messages telling the bank to transfer a large amount of money to the account of a not-entirely-trustworthy person. Unfortunately, the packets

decide to take the scenic route to the destination and go off exploring a remote corner of the network. The sender then times out and sends them all again. This time the packets take the shortest route and are delivered quickly so the sender releases the connection.

Unfortunately, eventually the initial batch of packets finally come out of hiding and arrive at the destination in order, asking the bank to establish a new connection and transfer money (again). The bank has no way of telling that these are duplicates. It must assume that this is a second, independent transaction, and transfers the money again.

This scenario may sound unlikely, or even implausible but the point is this: protocols must be designed to be correct in all cases. Only the common cases need be implemented efficiently to obtain good network performance, but the protocol must be able to cope with the uncommon cases without breaking. If it cannot, we have built a fair-weather network that can fail without warning when the conditions get tough.

Part - 2 of this que

Now let us see how the three-way handshake works in the presence of delayed duplicate control segments. In Fig. 6-11(b), the first segment is a delayed duplicate CONNECTION REQUEST from an old connection. This segment arrives at host 2 without host 1's knowledge. Host 2 reacts to this segment by sending host 1 an ACK segment, in effect asking for verification that host 1 was indeed trying to set up a new connection. When host 1 rejects host 2's attempt to establish a connection, host 2 realizes that it was tricked by a delayed duplicate and abandons the connection. In this way, a delayed duplicate does no damage.

The worst case is when both a delayed CONNECTION REQUEST and an ACK are floating around in the subnet. This case is shown in Fig. 6-11(c). As in the previous example, host 2 gets a delayed CONNECTION REQUEST and replies to it. At this point, it is crucial to realize that host 2 has proposed using y as the initial sequence number for host 2 to host 1 traffic, knowing full well that no segments containing sequence number y or acknowledgements to y are still in existence. When the second delayed segment finally arrives at host 2, the fact that z has been acknowledged rather than y tells host 2 that this, too, is an old duplicate. The important thing to realize here is that there is no combination of old segments that can cause the protocol to fail and have a connection set up by accident when no one wants it.

4. Explain the issue of regulating the sending rates to obtain desirable bandwidth controlling the congestion? Additive Increase Multiplicative Decrease (AIMD)? [6]

Ans -

Consider what happens from some starting allocation if both user 1 and user 2 additively increase their respective bandwidths over time. For example, the users may each increase their sending rate by 1 Mbps every second. Eventually, the operating point crosses the efficiency line and both users receive a congestion signal from the network. At this stage, they must reduce their allocations. However, an additive decrease would simply cause them to oscillate along an additive line. This situation is shown in Fig. 6-24. The behavior will keep the operating point close to efficient, but it will not necessarily be fair.

Similarly, consider the case when both users multiplicatively increase their bandwidth over time until they receive a congestion signal. For example, the users

may increase their sending rate by 10% every second. If they then multiplicatively decrease their sending rates, the operating point of the users will simply oscillate along a multiplicative line. This behavior is also shown in Fig. 6-24. The multiplicative line has a different slope than the additive line. (It points to the origin, while the additive line has an angle of 45 degrees.) But it is otherwise no better. In neither case will the users converge to the optimal sending rates that are both fair and efficient.

AIMD

Now consider the case that the users additively increase their bandwidth allocations and then multiplicatively decrease them when congestion is signaled. This behavior is the AIMD control law, and it is shown in Fig. 6-25. It can be seen that the path traced by this behavior does converge to the optimal point that is both fair and efficient. This convergence happens no matter what the starting point, making AIMD broadly useful. By the same argument, the only other combination, multiplicative increase and additive decrease, would diverge from the optimal point.