Important Instructions:

1) **Open this MS-Word document and start writing answers below each respective question given on page 2.**
2) **Answers the question in the same sequence in which they appear.**
3) **Provide to the point and concrete answers. Some of the questions are open ended and therefore must be answered using your own opinion and thoughts but backed with logical reasons.**
4) **First read the questions and understand what is required of you before writing the answer.**
5) **Attempt the paper yourself and do not copy from your friends or the Internet. Students with exactly similar answers or copy paste from the Internet will not get any marks for their assignment.**
6) **You can contact me for help if you have any doubt in the above instructions or the assignment questions.**
7) **All questions must be attempted.**
8) **Do not forget to write your name, university ID, class and section information.**
9) **Rename you answer file with your university ID# before uploading to SIC.**
10) **When you are finished with writing your answers and are ready to submit your answer, convert it to PDF and upload it to SIC unzipped, before the deadline mentioned on SIC.**

Spring Semester 2020 Final Exam
Course: - Distributed Computing

Deadline: - Mentioned on SIC **Marks: - 50**

Program: - MS (CS) **Dated: 24 June 2020**

Student Name: RAHMAT ULLAH **Student ID#:14337**

Class and Section: 4th Semester

Section: Remote Invocation

Q1. Describe briefly the purpose of the three communication primitives in request-reply protocols. (6)

ANS:

The purpose of request-reply protocols is to match request to replies. Moreover, it provides certain delivery guarantees. It has three communication primitives. The *doOperation, getRequest* and *sendReply*.

The purpose of *doOperation* is to invoke remote operations. Its arguments specify the remote server and which operations to invoke.

The purpose of *getRequest* is to acquire service requests.

The purpose of *sendReply* is to send reply message to the clients.

****

Q2. Explain the technical difference between RPC and RMI? (4)

ANS:

The basic difference between RPC and RMI is that RPC supports procedural programming, on the other hand, RMI supports object-oriented programming.

Moreover, RPC is a library and OS dependent platform whereas RMI is a java platform.

Similarly, RPC supports procedural programming whereas RMI supports object oriented programming.

In addition, RPC is less efficient in comparison of RMI because it creates more overhead whereas RMI is more efficient than RPC because it creates less overhead than RPC.

Finally, RPC does not provide any security while RMI provides client level security.

****

Section: Indirect Communication

Q:3 In contrast to Direct Communication, which two important properties are present in Indirect Communication? (6)

ANS:

In contrast to Direct Communication, the two important properties which are present in Indirect Communication are Time Uncoupling and Space Uncoupling.

Time Uncoupling property of Indirect Communication allows a sender to send a message even if the receiver is still not available. The message is stored and picked up at a later moment.

Similarly, with the property of Space Uncoupling, a sender has freedom to send a message but does not know or need to know whom it is sending to.

In short, the two important properties present in Indirect Communication lead to relaxing the requirement of the sender and receiver to be available or exist in the same time and know each other.

****

Q:4 Provide three reasons as why group communication (single multicast operation) is more efficient than individual unicast operation? (9)

ANS:

Amongst several benefits of group communication, following are the key three reasons.

1. **Managing Group Membership:** The central concept is of the group membership whereby processes may join or leave the group. Processes can send a message to this group and have it propagated to all members of the group. Hence message is sent to all the members of the group by a single operation instead of issuing multiple send operations to individual processes.
2. **Detecting Failures and Convenience for the Programmer:** The use of single multicast operation instead of multiple send operations amounts to much more than a convenience for the programmer. This does not require to detecting the individual failures. Only if the group is managed, the individual failure does not affect the group which is a key reason for the efficiency of group communication.
3. **Providing Reliability and Ordering guarantees with respect to time:** Group communication sends the message no more than once over any communication link which provides the reliability of the message being sent to each recipient and guarantees the order in which the message is to be received by sending it over the distribution tree. This not only provides reliability but also ensures the minimization of total time taken to deliver the message to all destinations as compared to individual or unicast operation.

****

Section: OS Support

Q5. Differentiate a between a network OS and distributed OS. (6)

ANS:

The main difference between network OS and distributes OS is that distributed OS are for some specific uses only. They are not used generally. Whereas the network OS are generally used by the end users to run their applications and meet their needs.

Moreover, the network OS allows users to have a degree of autonomy and have interactive responsiveness of their machines where the work of one user is totally in his/her hand and does not affect the other whereas the distributed OS does not lead the end user to be autonomous as the programs of one user can affect the other.

In other words, both the Network Operating System and Distributed Operating System have a common hardware base but the difference lies in software. Some more of the differences are as:

A network OS is made up of software and associated protocols that allow a set of computer network to be used together while a distributed OS is an ordinary centralized operating system but runs on multiple independent CPUs.

In network OS, Environment users are aware of multiplicity of machines whereas in distributed OS, Environment users are not aware of multiplicity of machines.

Control over file placement is done manually by the user in network OS while in distributed OS, the control lover file placement can be done automatically by the system itself.

****

Q6. Describe briefly how the OS supports middleware in a distributed system by providing and managing (6)

a) **Process and threads**
b) **System Virtualization**

ANS:

The task of middleware is to provide a high-level programming abstraction for the development of distributed systems. The OS supports middleware to do so with the following ways.

A) **<u>How the OS supports middleware in a distributed system by providing and managing Process and Threads:</u>**

The operating system running at a node supports middleware in distributed system by providing and managing process and threads associated with user level services such as communication libraries. Through the management of Process and threads, the OS provides ease to manage local hardware resources for processing, storage and communication. Middleware utilizes a combination of these local resources to implement its mechanism for remote invocations between objects or processes at the nodes. The OS enables the middleware to deliver distributed resources sharing to users. Kernel and server processes are the components that manage the resources and present clients with an interface to the resources by providing and managing process and Threads.

B) **<u>How the OS support middleware in a distributed system by providing and managing System Virtualization</u>**

They OS support middleware for System Virtualization by providing multiple virtual machines over underlying physical machine architectures. The OS enables the middleware to

support potentially large numbers of virtual machines and multiplex resources between them. The virtualization system allocates the physical processor(s) and other resources of a physical machine between all virtual machines that it supports. Hence on server machines, an organization assigns each service it offers to a virtual machine and then optimally allocates the virtual machine to physical servers. This way, the middleware get supported by the OS in a distributed system by providing and managing system virtualization.

****

Section: Distributed Objects and Components

Q7. Write in your own words the issues with Object (distributed) oriented middlewares. (13)

ANS:

As a matter of fact, an increasing number of next-generation applications will be developed as distributed "systems of systems," which include many interdependent levels, such as network/bus interconnects, local and remote end systems, and multiple layers of common and domain-specific middleware. There are number of issues with Object (distributed) oriented middleware. Some of them are the following as per my own understanding of the topic.

Implicit Dependencies: object interfaces do not describe what the implementation of the objects depend on , making object-based systems difficult to develop and subsequently manage.

Programming Complexity: Programming distributed object middleware leads to a need to master many low-level details associated with middleware implementations.

Lack of Separation of Distribution Concerns: Application developers are obliged to consider details of concerns such as security, failure handling and concurrency, which are largely similar from one application to another.

Moreover, Predictability of Operating Characteristics, which is that it is difficult to predict accurate performance of the working of middlewares.
The other issue I understand is Controllability of Operating Characteristics, which states that it is difficult for the operating system to control all the characteristics of Object oriented middlewares.
Moreover, the adaptability of Operating Characteristics for applications is another key concern. They may not be adaptable with all components.
I believe that all the above issues are associated with respect to some features such as;
Time,
Quantity of information,
Accuracy,
Confidence, and
Synchronization.
All these issues become highly volatile in systems of systems, due to the dynamic interplay of the many interconnected parts. These parts are often constructed in a similar way from smaller parts. Given the complexity of the issues with Object Oriented middlewares, various tools and techniques are needed to configure and reconfigure these issues hierarchically so they can adapt to a wider variety of situations and result in smooth operation.
****