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: zeeshan sajid_____ Student ID#:_14289_____

Class and Section:_MsCs_____

Section: Remote Invocation

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

(6)

A1)Three primitives

DO Operation

The do Operation method sends a request message to the server whose Internet address and port are mentioned in the remote reference given as an argument.

Get Request

get Request operation is used by a server to obtain service requests that are carried out by the clients.

SendReply

When the server has invoked the specified operation, it then uses send reply to send the reply message to the client. When the reply of the message is received by the client the original *do Operation* is unblocked and execution of the client program continues. .

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

A2)Difference B/W RPC and RMI

The basic difference is that RMI supports object oriented programming while RPC on the other hand supports procedural programming, the parameters passed to RPC are ordinary data structures where as RMI transits objects as parameters, RPC protocol generates more overhead then RMI ,in RPC request could not be possible because the two processes have different address space but in case of RMI it is possible, in RPC in-out parameters are passed which means that the value paseed

and the out put must be of the same data type where as in RMI there is no need to pass in-out parameters.

Section: Indirect Communication

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

A3) Space uncoupling

Is that in which the sender does not know or need to know the identity of the receivers and Because of this space uncoupling the system developer has many degrees of freedom in dealing with change participants can be replaced, updated, replicated or migrated.

Time uncoupling

Is that in which the sender and receiver can have independent lifetimes. the sender and receiver do not need to exist at the same time to communicate. This has important benefits for example, in more volatile environments where senders and receivers may come and go.

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

A4)Group communication(single multicast operation) points

The important feature of group communication is that a process issues only a single multicast operation to send a message to each group of processes instead of sending multiple operation to individual processes. Group communication is an important building block for distributed systems and particularly reliable distributed systems with key areas of application including, the reliable dissemination of information to potentially large numbers of clients, including in the financial industry, where institutions require accurate and up-to date access to a wide variety of information sources, support for system monitoring and management, including for example load balancing strategies. support for a range of fault-tolerance strategies, including the consistent update of replicated or the implementation of highly available servers. support for collaborative applications, where again events must be disseminated to multiple users to preserve a common user view – for example, in multiuser games. The use of a single multicast operation instead of multiple send operations amounts to much more than a convenience for the programmer: it enables the implementation to be efficient in its utilization of bandwidth. The use of a single multicast operation is also important in terms of delivery guarantees. If a process issues multiple independent send operations to individual processes.

Section: OS Support

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

A5) Network OS

A network operating system is made up of software's and is associated with protocols that allow different computer networks to work or to be used together. It retains autonomy in managing their own processing resources. With a network operating system a user can remotely log into another computer using secure shell ssh and run processes there while the operating system manages the processes running at its own node it does not manage processes across the nodes. A Network operating system is that in which users are never concerned with where their programs run or the location of any resources. There is a single system image. The operating system has control over all the nodes in the system, and it transparently locates new processes at whatever node suits its scheduling policies. The network operating system enables users to run their favorite word processors and other standalone applications.

Distributed OS

A distributed operating system is centralized OS but it runs on multiple independent systems. The users of the environment are aware of multiplicity of machines. The chief advantages of a Distributed -based operating system are its extensibility and its ability to enforce modularity behind memory protection boundaries. In addition, a relatively small kernel is more likely to be free of bugs than one that is larger and more complex. . The advantage distributed OS is centralized design with which operations can be invoked. In distributed systems components can have variety and differences in Networks, Computer hardware, Operating systems, Programming languages and implementations by different developers. Concurrency is a property of a system representing the fact that multiple activities are executed at the same time. In a distributed system hardware, software, network anything can fail. The system must be designed in such a way that it is available all the time even after something has failed.

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

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

A6) Process and threads

Threads are directly supported by OS any application can me programmed and updated to function as multithreaded. An application that contains threads can be supported with in a single process the process performs thread creation scheduling and management in the process space. A thread a basic unit of operating system that allocates processor time a thread can execute any part of the process code including the parts that are currently being executed by any other thread. The simplest term is a process. The process is an executing program. A process can run more then one thread. the term process means what we have called a thread. The reader may encounter in the literature the terms heavyweight process, where an execution environment is taken to be included, and lightweight process, where it is not.

System Virtulization

system virtualization refers to the use of software to allow system hardware to run multiple programs of different operating systems concurrently, allowing you to run different applications requiring different operating systems on one computer system at the same time. The concept stems from the observation that modern computer architectures have the necessary performance to support potentially large numbers of virtual machines and multiplex resources between them.

Multiple instances of the same operating system can run on the virtual machines or a range of different operating systems can be supported. The virtualization system allocates the physical processors and other resources of a physical machine between all virtual machines that it supports. On server machines, an organization assigns each service it offers to a virtual machine and then optimally allocates the virtual machines to physical servers. Unlike processes, virtual machines can be migrated quite simply to other physical machines, adding flexibility in managing the server infrastructure. This approach has the potential to reduce investment in server computers and to reduce energy consumption, a key issue for large server farms. Virtualization is very relevant to the provision of cloud computing. virtualization, allow users of the cloud to be provided with one or more virtual machines for their own use. System virtualization is implemented by a thin layer of software on top of the underlying physical machine architecture; this layer is referred to as a virtual machine monitor or hypervisor.

Section: Distributed Objects and Components

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

Middleware

The middle layers lies between the operating system and the applications

Issues

Todays business world has changed to a different environment of never ending contact through mobile phones and other hand handled devices. This means that there are countless apps in use today by billions of people who are constantly talking texting networking or working all these apps cannot handle the workload by themselves these must fall into the middleware that is bridging the gap between apps and OS. Instead of failures and unwanted waiting, the required is either reconfiguration to get back the wanted resource automatically or degradation if they are unavailable Reconfiguration and operating under less than optimal conditions both have two points of focus: individual and aggregate behavior. Moreover, there is a need for interoperability of control and management mechanisms .Middleware applications are the bridge that make your computer do what you want them to do with your systems without it the data base the client side and the internal and important messaging and many processes would be useless. The application and infrastructure of the middleware architecture is highly distributed which means that as we test or manage one should do it in a way that it is comprehensive in scope the whole system should be taken rather than components for looking at the integration . Another challenge for mangers and architects is the continuous monitoring of middleware from front end application to back end .which requires keen knowledge on how the complete infrastructure should work but also the identification of what problems may exist in this environment. it is too often the case that a substantial percentage of the effort expended to develop applications goes into building ad hoc and proprietary middleware substitutes, or additions for missing middleware functionality. As a result, subsequent composition of these ad hoc capabilities is either infeasible or prohibitively expensive. One reason why redevelopment persists is that it is still often relatively easy to pull together a minimalist ad hoc solution, which remains largely invisible to all except the developers. Unfortunately, this approach can yield substantial recurring downstream costs,

particularly for complex and long-lived distributed systems of systems. As with the adoption of middleware-based packages and shielding applications from heterogeneity, there has been much progress in the area of interoperability. Thus far, however, interoperability concerns have focused on data interoperability and invocation interoperability. Little work has focused on mechanisms for controlling the overall behavior of integrated systems, which is needed to provide "control interoperability." There are requirements for interoperable control capabilities to appear in individual resources first, after which approaches can be developed to aggregate these into acceptable global behavior. It is important to have a clear understanding of the QoS information so that it becomes possible to Identify the users requirements at any particular point in time and Understand whether or not these requirements are being met. It is also essential to aggregate these requirements, making it possible to form decisions policies, and mechanisms that begin to address a more global information management organization. Meeting these requirements will require flexibility on the parts of both the application components and the resource management strategies used across heterogeneous systems of systems. A key direction for addressing these needs is through the concepts associated with managing adaptive behavior, recognizing that not all requirements can be met all of the time, yet still ensuring predictable and controllable end-to-end behavior