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: Muhammad Faisal **Student ID#: 15139**

Class and Section: MSCS-iv FALL-18

Section: Remote Invocation

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

Answer. The request reply protocol is based on a trio of Communication primitives i.e. do Operation, get request and send reply

- The *do Operation* method sends a request message to the server whose Internet address and port are specified. The caller of *do Operation* is blocked until the server performs the requested operation and transmits a reply message to the client process.
- *Get Request* is used by a server process to acquire service requests
- When the server has invoked the specified operation, it then uses send reply to send the reply message to the client.

  When the reply message is received by the client the original do operation is unblocked & execution of the client program continuous.

The main purpose of the communication primitives in request reply protocol is to provide convenient communication between client and server.

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

Answer. The technical differences between RPC and RMI are as follow

- RPC support procedural programming while RMI support object oriented programming
- In RPC, ordinary data structure is passed to remote procedure while in RMI objects are passed to the remote method.
- RPC is less efficient while RMI is more efficient
- RPC invoke functions while RMI invoke methods.
- RPC does not have the capability to allow usage of design pattern while RMI have this capability due to object oriented.
- RPC is faster than RMI, since RMI involves execution of java bytecode.

Section: Indirect Communication

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

Answer. Space uncoupling: The sender does not know or need to know the identity of the receiver, and vice versa.

Time uncoupling: The sender and the receiver can have independent life times. the sender and receiver do not need to exist at the same time to communicate

Indirect communication is often used in distributed systems where change is anticipated.

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

Answer. Group communication offer a service whereby a message is sent to a group and then this message is delivered to all members of the group.

group communication is efficient due to the following reasons.

- **Enhanced Efficiency:** Controls network traffic and reduces server and CPU loads
- **Optimised Performance:** Eliminates traffic redundancy
- **Distributed Applications:** Makes multipoint applications possible

Section: OS Support

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

Answer. The main differences between Network operating system and distributed system are as follow

- A network operating system is made up of software and associated network protocols that allow a set of computer network to be used together. However, a distributed operating system is an ordinary centralized operating system but runs on multiple independent CPUs.
- In Network OS, Performance is badly affected if certain part of the hardware starts malfunctioning. However distributed OS is fault tolerant.
- Network operating systems are loosely coupled systems and are used in heterogeneous computers. While distributed OS is tightly coupled systems usually used in multiprocessors or homogeneous computers.
- Network OS have less transparency as compared to distributed OS.
- The network OS is easily implemented as compared to the distributed operating system.
- Network OS have high scalability as compared distributed OS.
- In network OS User manually selects a computer while in distributed System selects computers dynamically and automatically.
- In network OS different machine may have different OS while in Distributed it has single system wide.
- In network OS Each computer manages its local resources on the other hand in distributed OS it manages its resources Global Central or distributed

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

a) Process and threads

b) System Virtualization

Answer. A)

- ➢ A process consists of an execution environment, the unit of resource management `
  - An address space `
  - Activity (thread) synchronization and communication resources such as semaphores and communication interfaces
  - Higher-level resources such as open files and windows

- ➢ A Thread is an activity, execution
  - Threads within same process can share resources of the execution environment
  - An execution environment provides protection from threads outside it

B) In operating system virtualizations when the application does not interfere with another one even though they are functioning in the same computer. The kernel of an operating system allows more than one isolated user-space instance to exist. These instances call as software containers, which are virtualizations engines**.**

The operating system of the computer manages all the software and hardware of the computer. With the help of the operating system, several different computer programs can run at the same time. This is done by using the CPU of the computer. With the combination of few components of the computer which is coordinated by the operating system, every program runs successfully.

Section: Distributed Objects and Components

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

Answer. Issues with object-oriented middleware are:

Implicit dependencies

- ➢ Internal (encapsulated) behavior of the object is hidden, e.g. an object may communicate with other object or may use other services
- ➢ Not only a clear interface definition is needed but also the dependencies the object has on other objects in the distributed configuration.

Interaction with the middleware

- ➢ Programmers is exposed to many relatively low-level details associated with the middleware architecture – need to further simplifications
- ➢ Clean separation of concern is needed between code related to operation In a middleware framework and code associated with the application.

Lack of separation of distribution concerns

- ➢ Programmers have the explicitly deal with non-functional to issue such as security, coordination and replication.
- ➢ The complexities of dealing with the distributed system services should be hidden wherever possible from the programmer.

.**No support for deployment**

- Technologies such as Java RMI and COBRA does not support for the deployment of the deployed arbitrary distributed configurations.
- Middleware platforms should provide intrinsic support for deployment so that distributed software can be installed and deployed in the same way as software for a single machine.