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.

Re-Mid Semester Assignment
Course: - Distributed Computing

Deadline: - Mentioned on SIC Marks: - 30

Program: - MS (CS) Dated: 13 June 2020

Student Name: _AWAID ULLAH__ Student ID#_12714__

Class and Section:__MSCS_____

Question1: Discuss how the MMOG's as a Distributed System solves certain challenges due to its distributed architecture.
(6)

Question2: Among the trends of Distributed Systems discussed in C1-Lec2, which trend in your opinion will be most dominant in the future and why?
(6)

Question3: Among the challenges of Distributed Systems discussed in C1-Lec2, which problem in your opinion will accompany distributed systems into the future and why?
(6)

Question4: The design of distributed systems can be described and discussed in three ways i.e Physical Model, Architectural Model and Fundamental Model. Describe the example of distributed system in Question1 with respect to these three models. (6)

Question5: For the purpose of Inter Process Communication (IPC) in distributed systems, in what situation you will use UDP and TCP and why?
(6)

**Question1**: Discuss how the MMOG's as a Distributed System solves certain challenges due to its distributed architecture.

**Ans:** Massive multiplayer online games provide an immersive experience, and a large number of users interact with the persistent virtual world via the Internet. The best examples of such games include Sony's Ever Quest II and Finland's EVE Online

CCP game. The complexity of these worlds has greatly increased, and now includes complex gaming areas (such as EVE, online games are composed of a universe with more than 5,000 star systems) and many different social and economic systems. The number of players is also increasing, the system can support more than 50,000 simultaneous online players (due to the need for response time, the MMOG project is the main challenge of the distributed system technology. Other challenges include spreading the event to multiple players in real time and keeping it shared the world's consensus and many solutions for game design. Massively multiplayer online games: The largest online game, EVE Online, uses a client-server architecture to store a single copy of the state of the world on a centralized server, and can be accessed by the client program running on the player. The server that supports a large number of clients is a mature and complex entity, consisting of a cluster architecture with hundreds of computer nodes. The centralized architecture helps the management of the virtual world, and a single copy can also promote the consistency of concerns. Then, the goal is to ensure a rapid response by optimizing the network protocol and ensuring a rapid response to incoming events. To support this, by assigning each "star system" to a specific computer in the cluster divides the load. A heavily loaded star system has its own dedicated computer, while other star systems share a computer. Incoming events are routed to the correct computer in the cluster to track players in the galaxy Mobile. Other MMOGs use a more distributed architecture, where the universe is partitioned on a (possibly very large) number of servers, and these servers can also be distributed by geographic location. Then, based on the current usage model and the network delay on the server (For example, depending on the geographic location) users are dynamically assigned specific servers. This architectural style adopted by Ever Quest can naturally be expanded by adding new servers. Most commercial systems use one of the two models described above. But

the researchers now also looking for a peer-to-peer technology, where each participant contributes resources (storage and processing) to host the game.

**Question2:** Among the trends of Distributed Systems discussed in C1-Lec2, which trend in your opinion will be most dominant in the future and why?

**ANS:** Distributed systems are undergoing an important period of change, which can be attributed to many trends of influence: the emergence of ubiquitous network technologies; the emergence of ubiquitous computing, coupled with the desire to support user mobility in distributed systems; the growing demand for multimedia services; distributed multimedia systems treat distributed systems as utilities
Pervasive networking and the modern Internet
The Internet is an interconnected collection of many different types of computer networks, such as Wi-Fi, WiMAX, Bluetooth, and third-generation mobile phone networks. The end result is that the network has become a ubiquitous resource, and devices can be connected anytime, anywhere (if needed). The Internet allows programs running anywhere to send messages to programs elsewhere. The Internet is also a very large distributed system. It enables users to use services such as the World Wide Web, e-mail and file transfers wherever they are. The service package is open-it can be expanded by adding server computers and new types of services. The figure shows a collection of intranets, which are subnets operated by enterprises and other organizations, usually protected by firewalls. The role of the firewall is to protect the intranet by preventing unauthorized messages from coming out or entering. The firewall is implemented by filtering incoming and outgoing messages. An Internet Service Provider (ISP) is a company that provides broadband and other types of connections to individual users and small organizations so that they can access services anywhere on the Internet. The internal networks are linked together through a backbone network. The backbone network uses satellite connections, fiber optic cables and other high-bandwidth circuits, and has a high-capacity network link. Please note that some organizations may not want to connect their internal network to the Internet at all. For example, police and other security and law enforcement agencies may have at least some internal intranets isolated from the outside world (probably the most effective firewall-there is no physical connection when resources need to be shared between internal and external users, by preventing legitimate access to services Firewalls may also have problems in distributed systems. As a result, firewalls often require better mechanisms and strategies to supplement them. The implementation of the Internet and the services it supports has led to the development of practical solutions to many distributed system problems.

**Mobile and ubiquitous computing:**
Device miniaturization and technological advancements in wireless networks have increasingly led to the integration of small portable computing devices into distributed systems. These devices include: laptop computers. Portable devices such as smart phones, GPS-compatible devices, pagers (PDAs) and digital cameras. Wearable devices, such as smart watches with functions similar to PDAs. Built-in appliances in washing machines, high-fidelity audio systems, automobiles and refrigerators. In mobile computing, users who are far from the "home" intranet (intranet at work or at home) can always access resources through the devices they carry. They can continue to access the Internet (home intranet); more and more, regulations require users to use nearby resources, such as printers or even sockets, while traveling. The latter is also called location or context sensitive computing. Ubiquitous computing is the use of many small and inexpensive computing devices that exist in users' physical environments (including homes, offices, and even natural environments). The term "ubiquitous" is intended to imply that small computing devices will eventually become so common among everyday objects that they are hardly noticed. In other words, their computer behavior will be transparently and tightly linked to their body functions. Only when there are computers everywhere can they communicate with each other. Ubiquity overlaps with mobile computing, because in principle mobile users can benefit from computers anywhere. But they are usually different. Ubiquitous computing can benefit users, and they can stay in unique environments such as homes or hospitals.
**Distributed multimedia systems**.
Another important trend is the need to support multimedia services in distributed systems. Multimedia support can be effectively defined as the ability to support multiple media types in an integrated manner. Distributed systems support the storage, transmission, and representation of what are commonly referred to as discrete media types (such as images or text messages). The distributed multimedia system should be able to perform the same function on continuous media types such as audio and video. The key feature of continuous media types is that their integrity depends on the real-time relationship between the elements of the reserved media type. For example, in video, it is necessary to maintain the number of frames per second, and for real-time streaming, it is necessary to maintain the maximum delay or delay provided to deliver the image
**Distributed computing as a utility**
Many companies advocate the vision of using distributed resources as commodities or public services, analogous to other public services such as water or electricity. Using this model, resources are provided by the service provider and effectively rented out, not owned by the end user. The model is applicable to physical resources (storage and processing) and logical services. Users can access complex data centers and even IT infrastructure using the types of services now provided by companies such as Amazon and Google. In fact, users can be served through

virtual (virtual operating systems) rather than physical nodes. You can also use this method to provide software services on the Internet worldwide. Many companies now offer a full range of services for effective leasing, including email and calendar services. For example, Google recommends that the Google Apps Blade server is the least computational item, which includes processing and storage capacity (main memory). The blade system consists of a potentially large number of blade servers contained in the blade chassis. The chassis provides other items such as power, cooling, persistent storage (disk), network, and display. With this solution, a single blade server can be much smaller than a standard PC, and production costs are also lower. The overall goal of the cluster computer is to provide a series of cloud services, including high-performance computing, mass storage (data center) and richer application services (such as Web search).

**Question3:** Among the challenges of Distributed Systems discussed in C1-Lec2, which problem in your opinion will accompany distributed systems into the future and why?

**ANS: Heterogeneity**

Internet users access services on heterogeneous computers and network collections. Heterogeneity (diversity and difference) applies to networks; equipment; bones; programming languages; implementation. The heterogeneity of the Internet is obscured by the use of Internet protocols. The representation of the data type may be different on different hardware. Different programming languages use different representations for characters and data structures (such as arrays and records). These differences must be considered so that programs written in different languages can communicate with each other. Unless they use common standards, programs written by different developers cannot communicate with each other.

**Openness**

The distributed system mainly depends on the extent to which new resource sharing services can be added and provided to various client programs. Only by publishing the specifications and documents of the main software interfaces can the opening be realized. This process is similar to standardized interfaces. The challenge for designers is to solve the complexity of a distributed system composed of many components designed by different people. By adding computers to the network, the system can be expanded at the hardware level, and the software level can be expanded by introducing new services and re-implementing the old services, allowing applications to share resources. Another advantage of open systems is that they are independent of various vendors.

**Security**

The information resources available and maintained in the distributed system are of great value to its users. Therefore, their safety is very important. The security of

information resources consists of three components: confidentiality (preventing unauthorized personnel from leaking), integrity (preventing tampering or damage), and availability (preventing media from being disturbed) resources, although firewalls can be used to provide protection against external threats Protection, but they cannot ensure that intranet users use resources correctly, nor can they ensure that resources are properly used on the firewall. The Internet is not protected by a firewall.

Security

Following two security challenges:

Denial of service attacks:

A user may wish to disrupt a service for some reason. This can be achieved by bombarding the service with such a large number of pointless requests that the serious users are unable to use it.

Security of mobile code:

Consider someone who receives an executable program as an electronic mail attachment: it may seem to display an interesting picture but in reality it may access local resources.

**Scalability**

From small intranets to the Internet, distributed systems can operate effectively on many different scales. If the system is still effective with a significant increase in the number of resources and the number of users, the system is described as scalable. The number of computers and servers on the Internet has greatly increased.

Fault management

A failure in a distributed system is a partial failure, that is, some components fail, while other components continue to operate. Therefore, managing failures is particularly difficult.

Simultaneous access

In a distributed system, multiple clients can try to access shared resources at the same time. The process of managing shared resources can only accept one client request at a time. But this method limits the process. As a result, services and applications often allow multiple client requests to be processed simultaneously. Multiple threads can run simultaneously in an object. In this case, their operations on the object may conflict and produce inconsistent results. Any object that represents a shared resource in a distributed system should be responsible for ensuring that it works properly in a concurrent environment using threads

**Transparency**

Transparency is the concealment of the separation of components in a distributed system by users and application programmers, so the system is regarded as a whole, not a collection of independent components. The meaning of transparency has a significant impact on the design of system software. To illustrate the transparency of access, please consider a graphical user interface with folders. This

interface is the same regardless of whether the files in the folder are local or remote. As an example of lack of access transparency, consider a distributed system that does not allow you to access files on remote computers unless you use an ftp program.

**Service quality**

The main non-functional attributes of the system that affect the quality of service of customers and users are reliability, security and performance. In the design of most computer systems, reliability and security issues are critical. The performance aspects of quality of service were initially defined in terms of responsiveness and throughput, but have been redefined in terms of the ability to meet punctuality guarantees. For example, for movie services, continuous video images must be displayed to users within certain specified time limits.

**Question4:** The design of distributed systems can be described and discussed in three ways i.e Physical Model, Architectural Model and Fundamental Model. Describe the example of distributed system in Question1 with respect to these three models.

ANS: There are three important complementary ways to describe and discuss the design of distributed systems:

**Physical model**

The physical model is the most definite way to describe the system. They capture the hardware of the system from the perspective of computers (and other devices, such as mobile phones) and their interconnected networks.

**Architectural model**

The architectural model describes the system in terms of calculation and communication tasks performed by the computing elements of the system. The computing element is a single computer or a collection of it supported by an appropriate network interconnection.

**Fundamental Model**

The basic model uses abstract views to examine all aspects of the distributed system. Three important aspects of distributed systems: interaction model, failure model and; security model,

**Physical model** The physical model is a representation of the underlying hardware elements of a distributed system, which is free from the specific details of the computer and network technology used. Basic physical model: A distributed system is a system in which hardware or software components located on a networked computer communicate and coordinate their actions only by transmitting messages. This leads to the smallest physical model of a distributed system, which is a set of scalable computer nodes interconnected by a computer network for the required delivery of messages. In addition to this basic model, there are three generations of distributed systems. Early distributed systems, Internet-wide distributed systems, contemporary distributed systems,

**Architecture model:** A system structure based on individually specified components and their interrelationships. The purpose is to ensure that the structure meets current and future possible requirements. The main concern is to make the system reliable, manageable, adaptable and profitable. The architectural design of a building has similar aspects-it not only determines its appearance, but also its overall structure and architectural style (classic, modern), and provides a consistent frame of reference for the design.

Architectural element: Communication entity: From a system perspective, the entity communicating in a distributed system is usually a process, combined with an appropriate inter process communication paradigm, but from a programming perspective, this is not yet not enough, more problem-oriented abstractions have been proposed. Objects: Objects are introduced to allow and encourage the use of object-oriented methods in distributed systems. Components: Because distributed objects have many important problems, the use of component technology has emerged. Components look like objects because they provide problem-oriented abstractions for building distributed systems and can also be accessed through interfaces. The difference is that in addition to interfaces, components also make all dependencies explicit and provide contracts. The construction of the system is more complete. This allows third parties to develop components, and also promotes a more pure combination approach to build distributed systems by eliminating hidden dependencies.

**Fundamental Model**

Basic model the basic model should contain only the basic elements that must be considered in order to understand and reason about certain aspects of system behavior. The purpose of this model is to explain all relevant assumptions on the system we are modeling. Given these assumptions, generalize what is possible or impossible. The generalization can take the form of general algorithms or guaranteed desired attributes. Interaction model: A distributed system consists of many processes that interact in complex ways. For example: multiple server processes can cooperate with each other to provide services, such as the domain name system. Simple programs are controlled by algorithms, and the steps in these algorithms are executed strictly in order and executed as a single process. A distributed system composed of multiple complex processes. Their behavior and state can be described by distributed algorithms, and the steps can be the transmission of messages between them. Messages are transmitted between processes to pass information between them and coordinate their activities.

**Question5:** For the purpose of Inter Process Communication (IPC) in distributed systems, in what situation you will use UDP and TCP and why?

**ANS:** Messaging between a pair of messages can be supported by two message communication operations (send and receive), one process sends the message to the target, and the other process sends the message to the target. This activity may

involve synchronizing the two processes. Synchronously, the sending and receiving processes are synchronized with each message. In this case, sending and receiving are blocking operations. Each time a send is sent, the sending process (or thread) is blocked until the corresponding receipt is sent. Each time a process (or thread) sends a received message, the received message will hang until the message arrives. In asynchronous, the use of the sending operation is not blocked, and once the message has been copied to the local buffer and the transmission of the message continues in parallel with it, the sending process can continue. Sending process. Receive operations can have blocking and non-blocking variables.

**UDP and TCP.**

The socket comes from BSD UNIX, but it also exists in most other versions of UNIX, including Linux as well as Windows and Macintosh OS. IPC involves transferring messages between a socket in one process and a socket in another process. For a process to receive a message, it must link its socket to a local port and one of the Internet addresses of the computer that is running. Messages sent to a specific Internet address and port number can only be received by the process whose socket is associated with that Internet address and port number.

**UDP**

Datagrams sent by UDP can be transmitted without confirmation or retry. If it fails, the message may not arrive. To send or receive messages, the process must first create a socket that is linked to the Internet address and local port of the local host. The server binds its socket to the server port-it is a port known by the client so that the client can send messages to it. The client binds its socket to any available local port. In addition to the message, the receive method also returns the sender's Internet address and port so that the receiver can send a reply. The following are some issues related to datagram communication: Message size: The receiving process must specify the specific size of the message. If the message is too large, it will be truncated on arrival. Any application that requires messages larger than the maximum number of messages should divide it into fragments of this size. Blocking: Sockets usually provide non-blocking transmission and blocking reception for datagram communication. After the send operation passes the message to the basic UDP and IP protocols, the send operation returns, and these protocols are responsible for transmitting the message to its destination. This method will receive blocks until a datagram is received, unless a timeout has been defined on the socket. If the process has other tasks to do while waiting for the message, it should use a separate thread. Expiration time: The server waiting for the server to receive the client's request can use the permanently suspended receiving information. However, in some programs, it is not appropriate to wait indefinitely for the process of calling the receiving operation in the case where the sending process may crash or the expected message may be lost. In order to meet these requirements, a failure time can be set on the socket. Choosing the right time

interval is difficult, but it must be large relative to the time required to send the message. Receive from all content: The receive method does not specify the source of the message. Instead, the call to receive sends a message to its socket from any source. The receive method returns the sender's Internet address and local port, so that the recipient can verify the source of the message. The datagram socket can be connected to a remote port and a specific Internet address, in which case the socket can only send and receive messages from that address. Failure model for UDP datagrams: Messages may be deleted from time to time due to checksum errors or because there is no available buffer space at the source or destination. Sometimes the mail may be sent by the sender's order. Applications that use UDP datagrams must provide their own checks to obtain the required reliable communication quality. Use of UDP For some applications, you can use services that may cause occasional omission failures. For example, DNS and VOIP are implemented through UDP. UDP datagrams are sometimes an attractive option because they are not affected by the overhead costs associated with guaranteed messaging. There are three main sources of overhead: the need to store state information at the source and destination; the transmission of additional messages; and the delay of the sender.

**TCP**

The TCP API from BSD 4.x UNIX provides an abstraction of the byte stream where data can be written and read from. The abstraction of the flow masks the following network characteristics: Message size: The basic implementation of the TCP flow determines the amount of data to be collected before transmitting data in the form of one or more packets. Upon arrival, the data will be transferred to the application as required. Lost messages: The TCP protocol uses an acknowledgement scheme to reliably deliver messages. Flow control: The TCP protocol attempts to match the speed of processes that read and write in the flow. Message repetition and classification: The message identifier is associated with each IP packet, which allows the recipient to detect and reject duplicates, or reorganize messages that did not arrive in the order of the sender. Message target: A pair of communication processes establish a connection before they can communicate on the stream. After the connection is established, these processes only need to read and write to the stream without using the Internet address and port. Establishing a connection involves connecting the client to the server's request, and then accepting the server's request to the client before any communication can take place. For a single client server request and response, this may represent a considerable overhead. When a pair of processes establish a connection, one of them plays the role of client and the other plays the role of server, but then they can become peers. The client role involves creating a stream socket linked to any port, and then creating a connection request to request a connection to the server on its server port. The server role includes creating a listening socket linked to the server port and waiting for the client to request a

connection. The listening socket maintains a queue of incoming connection requests. In the socket model, when the server accepts the connection, a new stream socket will be created for the server to enable it to communicate with the client, while keeping its socket on the server port to listen for Connection requests from other clients. A pair of client and server sockets are connected by a pair of streams, one in each direction. Therefore, each socket has an input stream and an output stream. One of these two processes can send information to the other by writing to its output stream, and the other process can get information by reading from its input stream. When the application closes the socket, it indicates that it no longer writes data to its output stream. All data from the output buffer will be sent to the other end of the stream and placed in the queue of the target socket to indicate that the stream has been interrupted. The target process can read data from the queue, but any further reading after the queue is empty will indicate the end of the stream. When a process ends or fails, all its sockets will eventually be closed, and any process that attempts to communicate with it will find that its connection has been disconnected. The following are some outstanding issues related to streaming communications: Data element matching: Two communications processes must agree on the content of data transmitted on the stream. For example, if a process writes an in in the stream and then a double, the reader at the other end must read an in and then a double. When a pair of processes cannot properly cooperate when using a stream, the reading process may encounter errors in interpreting data, or may crash due to insufficient data in the stream. Blocking: Data written to the stream remains in the queue on the target socket. When the process attempts to read data from the input channel, it either retrieves the data from the queue or suspends until the data becomes available. If the socket on the other end queues as much data as the protocol allows, then the process of writing data to the stream can be prevented by the TCP flow control mechanism. Thread: When the server accepts the connection, it usually creates a new thread to communicate with the new client. The advantage of using a separate thread for each client is that the server can block while waiting for input without delaying other clients. Failure model • to meet the integrity of reliable communications, TCP flows use checksums to detect and reject damaged packets and