CSL801

RIZVI EDUCATION SOCIETY'S

Rizvi College of Engineering

Department of Computer Engineering

Distributed Computing Lab

2024-25

NAME:

CLASS:

ROLL NO. / UIN:

RIZVI EDUCATION SOCIETY'S

Rizvi College of Engineering

Department of Computer Engineering

Lab File

Class & Semester: *B.E (Sem VIII)*

Subject: Distributed Computing Lab (DCL)

Name: ……

Roll No.: ……

UIN: ……

Document Revised on *02 / 01 / 2024.*

RIZVI EDUCATION SOCIETY'S

Rizvi College of Engineering

Department of Computer Engineering

Certificate

This is to certify that ……. of fourth year Computer Engineering Department has performed all the experiments of Distributed Computing Lab with satisfactory results.

……..

Subject In-charge

INSTITUTE VISION

To become a leading entity in transforming the diverse class of learners into innovators, analyzers and entrepreneurs competent to develop eco-friendly sustainable solutions and work in multi-disciplinary environment to meet the global challenges and contribute towards nation building.

DEPARTMENT VISION

To become a center of excellence in the field of Computer Engineering to transform diverse class of learners into skilled professionals with ethical values capable of contributing towards nation building.

PROGRAM OUTCOMES

PO1. **Engineering knowledge**
PO2. **Problem analysis**
PO3. **Design/development of solutions**
PO4. **Conduct investigations of complex problems**
PO5. **Modern tool usage**
PO6. **The engineer and society**
PO7. **Environment and sustainability**
PO8. **Ethics**
PO9. **Individual and teamwork**
PO10. **Communication**
PO11. **Project management and finance**
PO12. **Life-long learning**

PROGRAM SPECIFIC OUTCOMES

PSO1. **Open-Source Tools**: To encourage the students to work using open-source software's & tools in diversified areas of computer science.

PSO2. **Industry Readiness**: To enable the students to acquire the necessary skill set required to develop, test, install, deploy, and maintain a complete software system for business and other applications, that makes them industry ready.

INSTITUTE MISSION

IM1. **Impart Core Fundamental principles:** To impart core fundamental principles of engineering and science, through conventional and innovative methods which will help the learners acquire skillset to create solutions to complex engineering problems**.**

IM2. Bridge the Technical Skill Gap: To bridge the industry – academia gap through curriculum enrichment activities for industry readiness**.**

IM3. Inculcate Professional Etiquettes and Ethics: To groom the learners through well-planned training & placement activities laced with professional etiquettes and ethics leading to their holistic development thus enabling them to acquire distinguished positions in prominent organizations and inculcating lifelong learning capability.

IM4. Research and Development: To provide modern infrastructure and the necessary resources, for planning and implementing innovative research ideas, leading to entrepreneurship.

DEPARTMENT MISSION

DM1. **Quality Technical Education**: To provide quality technical education with the help of modern resources for finding solutions to complex problems of computer engineering.

DM2. **Leadership & Entrepreneurial skills**: To inculcate moral and ethical values while acquiring leadership, entrepreneurial skills and overall personality development.

DM3. **Professional Skills and Lifelong Learning**: To inculcate professional skills and lifelong learning for further education as well as acquiring distinguished positions in leading software industries.

DM4. **Research and Development**: To encourage creative thinking for planning and implementing innovative ideas leading to meaningful research and development considering economical, societal and environmental factors.

PROGRAM EDUCATIONAL OBJECTIEVES

PEO1. **Successful Career**: To build a successful career in leading industries related to the field of Computer Engineering wherein the engineer will be able to provide the necessary solutions to the challenges witnessed in existing and new business models.

PEO2. **Leadership Qualities**: To exhibit the qualities of team spirit, leadership, and problem-solving skills to achieve top positions in the organization or to enhance entrepreneurial skills.

PEO3. **Adaptability to New Technology**: To be able to adapt to new technologies and platforms and share their knowledge with their peers in the allied fields.

PEO4. **Research & Higher Studies**: To develop proficiency in computer engineering and related fields to be able to work in multi-disciplinary areas with a strong focus on innovation and research.

Lab Objectives

1. To understand basic underlying concepts of forming distributed systems.
2. To learn the concept of clock Synchronization
3. To learn Election Algorithm.
4. To explore mutual exclusion algorithms and deadlock handling in the distributed system
5. To study resource allocation and management.
6. To understand the Distributed File System

Lab Outcomes

Learner will be able to…

1. Develop test and debug using Message-Oriented Communication or RPC/RMI based client-server programs.
2. Implement techniques for clock synchronization.
3. Implement techniques for Election Algorithms.
4. Demonstrate mutual exclusion algorithms and deadlock handling.
5. Implement techniques of resource and process management.
6. Describe the concepts of distributed File Systems with some case studies.

Rubrics

Following rubrics will be used to assess the work submitted by the students.

1. For Experiment

Criteria 1: Delivery			
Submitted practical in stipulated time			
4	**3**	**2**	**1**
Excellent	Good	Average	Below Average
Submitted previous lab writeups on next coming practical.	Submitted after one week of due date.	Submitted after two weeks of due date.	Submitted after three weeks of due date.
Criteria 2: Understanding			
How well the topic is understood to the students based on question asked to them at the time of checking			
4	**3**	**2**	**1**
Excellent	Good	Average	Below Average
Answered all the questions asked with appropriate answers	Answered all the questions but some answers were not up to mark.	Answered some of the questions correctly & some incorrectly.	Answers were not satisfactory.
Criteria 3: Readability /Presentation			
Easy to understand for readers			
4	**3**	**2**	**1**
Excellent	Good	Average	Below Average
The code is exceptionally well organized with comments / heading and very easy to follow.	The code is fairly easy to read without comments / headings	The code is readable only by someone who knows what it is supposed to be doing.	The code is poorly organized and very difficult to read.
Criteria 4: Discipline			
Discipline refers to rule following behavior, to regulation, order, control and authority.			
4	**3**	**2**	**1**
Excellent	Good	Average	Below Average
Always following rules and regulations in order maintain discipline in a laboratory.	Most of the time following rules and regulations in order maintain discipline in a laboratory.	Sometime following rules and regulations in order maintain discipline in a laboratory.	Not following rules and regulations in order maintain discipline in a laboratory.

2. For Assignments

Criteria 1: Delivery			
Submitted Assignment in stipulated time			
4	**3**	**2**	**1**
Excellent	Good	Average	Below Average
Submitted on time	Submitted after one week of due date.	Submitted after two weeks of due date.	Submitted after three weeks of due date.
Criteria 2: Understanding			
How well the topic is understood to the students based on question asked to them at the time of checking			
4	**3**	**2**	**1**
Excellent	Good	Below Average	Below Average
Answered all the questions asked with appropriate answers	Answered all the questions but some answers were not up to mark.	Answered some of the questions correctly & some incorrectly.	Answers were not satisfactory.
Criteria 3: Readability /Presentation			
Easy to understand for readers			
4	**3**	**2**	**1**
Excellent	Good	Average	Below Average
The theory is exceptionally well organized with heading or diagram and very easy to follow.	The theory is well organized with heading but no diagrams.	The theory is readable only by someone who knows what it is supposed to be doing with some heading or diagram.	The theory is poorly organized and very difficult to read.
Criteria 4: Discipline			
Discipline refers to rules following behavior, to regulation, order, control and authority.			
4	**3**	**2**	**1**
Excellent	Good	Average	Below Average
Always following rules and regulations in order maintain discipline in a classroom.	Most of the time following rules and regulations in order maintain discipline in a classroom.	Sometime following rules and regulations in order maintain discipline in a classroom.	Not following rules and regulations in order maintain discipline in a classroom.

Lab Guidelines

- Students are advised to come to the laboratory on time.
- Sign in the laboratory login register, write the TIME-IN, and occupy the computer system allotted to you by the faculty.
- Execute your task in the laboratory and record the results / output in the lab observation notebook and get certified by the concerned faculty.
- Avoiding Plagiarism, it is a form of dishonesty, and it can be avoided by being honest while writing.
- All the students should be polite and cooperative with the laboratory staff, must maintain the discipline and decency in the laboratory.
- Computer labs are established with sophisticated and high-end branded systems, which should be utilized properly.
- Students / Faculty must keep their mobile phones in SWITCHED OFF mode during the lab sessions. Misuse of the equipment, misbehaviors with the staff and systems etc., will attract severe punishment.
- Students must take the permission of the faculty in case of any urgency to go out; if anybody found loitering outside the lab / class without permission during working hours will be treated seriously and punished appropriately.
- Students should LOG OFF/ SHUT DOWN the computer system before he/she leaves the lab after completing the task (experiment) in all aspects. He/she must ensure the system / seat is kept properly.

INDEX

EXPERIMENTS

Experiment No. 1

Title: *Write a program to demonstrate Datagram Socket for chat application using java.*

Rubric	***Score (0 to 4)***
Delivery	
Understanding	
Readability	
Discipline	
Total	

Performed On: _____

Sign: _____

Experiment No. 1

Distributed Computing Lab (DCL)

Aim: Write a program to demonstrate Datagram Socket for chat application using java.

Software: Jdk 1.8.0, Eclipse.

Pre-Lab Questions:

1. What is Datagram Socket?
2. What are the steps involved in using Socket?

Post-Lab Questions:

1. Which constructor of Datagram Socket class is used to create a datagram socket and binds it with the given Port Number?
2. What does the java.net. InetAddress class represent?

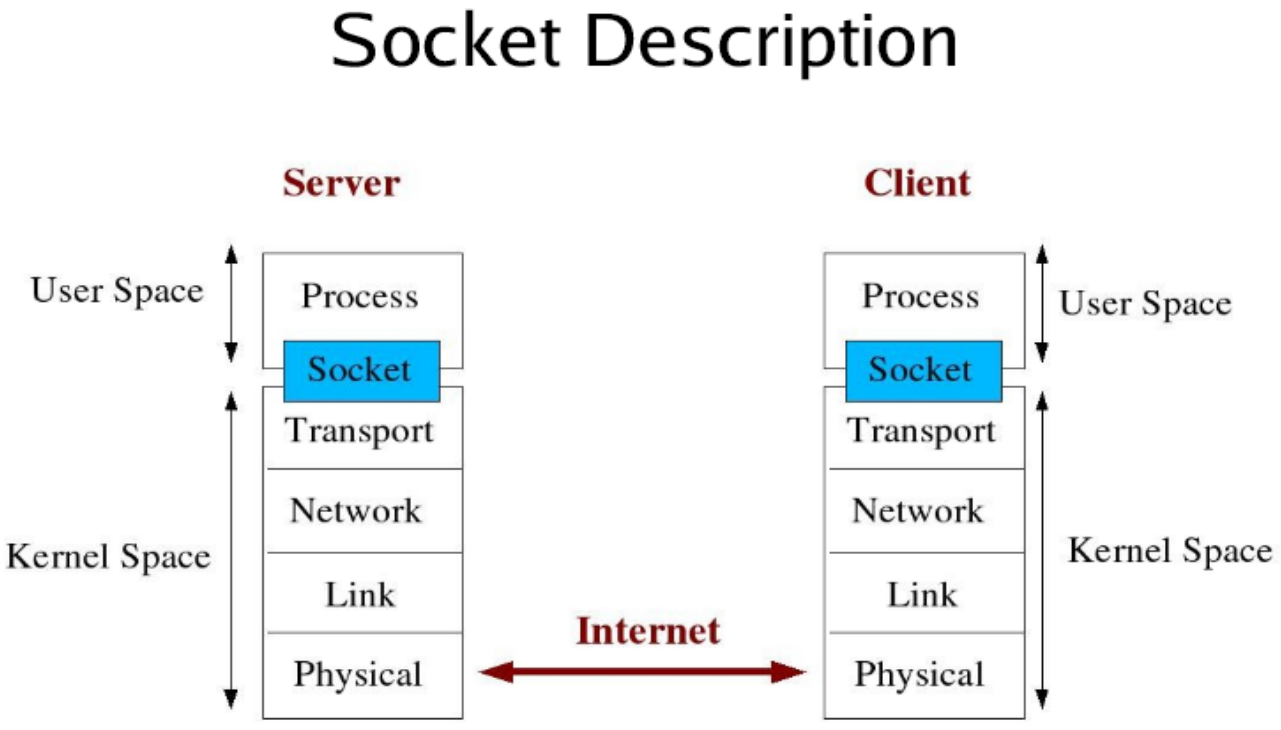
Theory:

What is Datagram Socket Programming?

The server continuously receives datagram packets over a datagram socket. Each datagram packet received by the server indicates a client request for a quotation. When the server receives a datagram, it replies by sending a datagram packet that contains a one-line "quote of the moment" back to the client.

The client application in this example is fairly simple. It sends a single datagram packet to the server indicating that the client would like to receive a quote of the moment. The client then waits for the server to send a datagram packet in response.

Socket is an interface between an application process and transport layer. The application process can send/receive messages to/from another application process (local or remote) via a socket.



There are a few steps involved in using sockets:

- Create the socket.
- Identify the socket.
- On the server, wait for an incoming connection.
- On the client, connect to the server's socket.
- Send and receive messages.
- Close the socket.

Program:

Conclusion:

……
……
……

CO's Covered:

……
……
……

Experiment No. 2

Title: *Write a Program to demonstrate Inter-process Communication in Client-Server Environment Using Java.*

Rubric	***Score (0 to 4)***
Delivery	
Understanding	
Readability	
Discipline	
Total	

Performed On: _____

Sign: _____

Experiment No. 2

Distributed Computing Lab (DCL)

Aim: Write a Program to demonstrate Inter-process Communication in Client-Server Environment using Java.

Software: jdk 1.8.0, Eclipse.

Pre-Lab Questions:

1. What is Inter-process communication?
2. What are the models of IPC?

Post-Lab Questions:

1. Differentiate between synchronous and asynchronous communication?
2. What is event synchronization?

Theory:

Inter-process communication (**IPC**) refers specifically to the mechanisms an operating system provides to allow processes it manages to share data. Typically, applications can use IPC categorized as clients and servers, where the client requests data and the server responds to client requests. Many applications are both clients and servers, as commonly seen in distributed computing.

Inter-process communication differs from communication among threads within the same program in several ways:

1. Different processes execute in different address spaces. (They have separate heaps.)
2. Different processes may execute on different machines.
3. Different processes may execute in different administration domains (and may not trust each other.

These differences have certain implications for the way communication must occur between different processes.

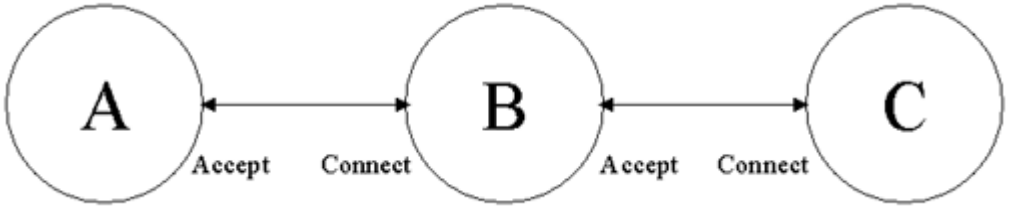
1. Local memory addresses are meaningless on the "other side" if transmitted. So, any data must be passed by value or if a reference is required, special mechanisms must be in place to resolve the references (they can't be ordinary addresses.)
2. The network must be involved when processes live on different machines.
3. Care must be taken to ensure that sensitive data is not compromised by untrusted applications involved in the communication (security) or by applications snooping on the wire (encryption).

   Let's start by spending some time discussing briefly how communication occurs in the network. Then we'll see how JAVA supports using this form of communication. We'll only touch on the security issue in passing.
   If we take a simplified view of the internet, it might look something like this:

- A collection of hosts (computers) at the edges of the network, where
- Each computer is connected to a **local area network (LAN)**, and
- Every host has a unique address (called an **IP address -- Internet Protocol address**)

Transmission Control Protocol (TCP)

- A sender transmits packets, but also saves them on the side in a buffer
- When a receiver gets a packet, it sends an acknowledgement (ACK) to the sender for that packet.
- If the sender doesn't get an ACK after a certain period of time, it retransmits the packet.
- To improve **throughput** multiple packets (say n) are sent before waiting for the first ACK and when the first ACK comes back, the next packet (n+1) can be sent and so on. This is called a "sliding window protocol".
- Packets are delivered to the receiving application in the same order they were sent by the sender. (FIFO)



Here, B is a client of A and a server for C.

Program:

Conclusion:

……

……

……

CO's Covered:

……

……

……

Experiment No. 3

Title: *Write a program to demonstrate the steps of Remote Method Invocation (RMI) application using java.*

Rubric	***Score (0 to 4)***
Delivery	
Understanding	
Readability	
Discipline	
Total	

Performed On: _____

Sign: _____

Experiment No. 3

Distributed Computing Lab (DCL)

Aim: Write a program to demonstrate the steps of Remote Method Invocation (RMI) application using java.

Software: jdk.1.8.0, Eclipse.

Pre-Lab Questions:

1. What is RPC and RMI?
2. What are the steps involved in RMI execution process?

Post-Lab Questions:

1. Explain Marshalling and unmarshalling?
2. What are the steps involved to make work a RMI program?

Theory:

Remote Method Invocation (RMI)

Java's Remote Method Invocation (commonly referred to as RMI) is used for client and server models. RMI is the object-oriented equivalent to RPC (Remote procedure call). The Java Remote Method Invocation (RMI) system allows an object running in one Java Virtual Machine (VM) to invoke methods on an object running in another Java VM. RMI provides for remote communication between programs written in the Java programming language.

RMI is defined to use only with the Java platform. If you need to call methods between different language environments, use CORBA. With CORBA a Java client can call C++ server and/or a C++ client can call a Java server. With RMI that cannot be done.

RMI uses stub and skeleton object for communication with the remote object. A remote **object** is an object whose method can be invoked from another JVM.

Stub

The stub is an object, acts as a gateway for the client side. All the outgoing requests are routed through it. It resides at the client side and represents the remote object. When the caller invokes method on the stub object, it does the following tasks:

1. It initiates a connection with remote Virtual Machine (JVM),
2. It writes and transmits (marshals) the parameters to the remote Virtual Machine (JVM),
3. It waits for the result
4. It reads (unmarshals) the return value or exception, and
5. It finally, returns the value to the caller.

Skeleton

The skeleton is an object, acts as a gateway for the server-side object. All the incoming requests are routed through it. When the skeleton receives the incoming request, it does the following tasks:

1. It reads the parameter for the remote method.
2. It invokes the method on the actual remote object, and
3. It writes and transmits (marshals) the result to the caller.

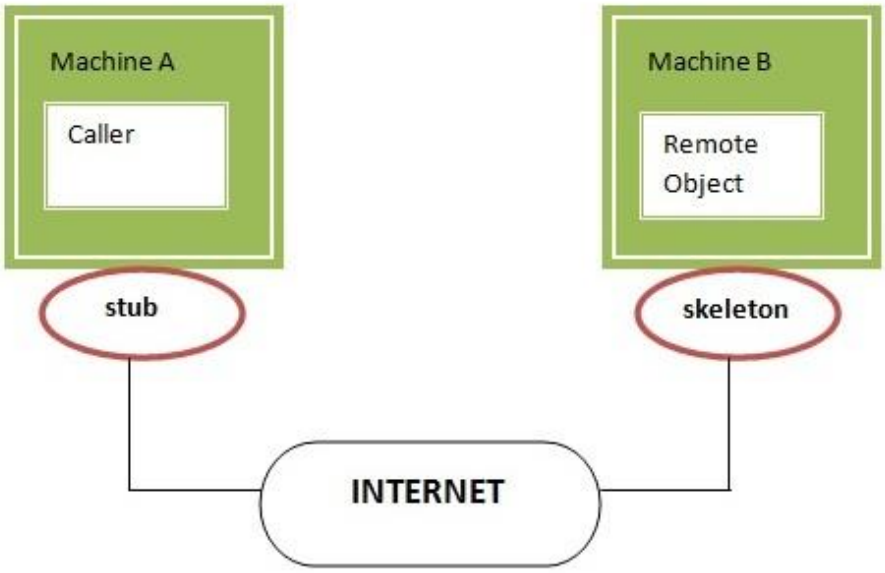


Figure: RMI Implementation

Program:

Conclusion:

……

……

……

CO's Covered:

……

……

……

Experiment No. 4

Title: *Write a Program to Demonstrate Group Communication using Java.*

Rubric	***Score (0 to 4)***
Delivery	
Understanding	
Readability	
Discipline	
Total	

Performed On: _____

Sign: _____

Experiment No. 4

Distributed Computing Lab (DCL)

Aim: Write A Program to Demonstrate Group Communication using Java.

Software: jdk1.8.0, Eclipse.

Pre-Lab Questions:

1. What is group communication?
2. What is master and slave?

Post-Lab Questions:

1. What is Server Socket?
2. What is port?

Theory:

Java Groups is a group communication toolkit written entirely in Java. It is based on IP multicast, but extends it with

1. reliability and
2. Group membership.

Reliability includes.

- lossless transmission of a message to all recipients (with retransmission of missing messages)
- fragmentation of large messages into smaller ones and reassembly at the receiver's side
- ordering of messages, e.g. messages m1 and m2 sent by P will be received by all receivers in the same order, and not as m2, m1 (FIFO order)
- Atomicity: a message will be received by all receivers, or none.

Group Membership includes:

- Knowledge of who the members of a group are and
- Notification when a new member joins, an existing member leaves, or an existing member has crashed.

The table below shows where Java Groups fits in. In unicast communication, where one sender sends a message to one receiver, there is UDP and TCP. UDP is unreliable, packets may get lost, duplicated, may arrive out of order, and there is a maximum packet size restriction. TCP is also fragmenting packets that are too big and present messages to the application in the order in which they were sent. In the multicast case, where one sender sends a message to many receivers, IP Multicast extends UDP: a sender sends messages to a multicast address and the receivers have to join that multicast address to receive them. Like in UDP, message transmission is still unreliable, and there is no notion of membership (who has currently joined the multicast address).

	Unreliable	Reliable
Unicast	UDP	TCP
Multicast	IP Multicast	Java Groups

Java Groups extends reliable unicast message transmission (like in TCP) to multicast settings. As described above it provides reliability and group membership on top of IP Multicast. Since every application has different reliability needs, Java Groups provides a flexible protocol stack architecture that allows users to put together custom-tailored stacks, ranging from unreliable but fast to highly reliable but slower stacks. As an example, a user might start with a stack only containing IP Multicast. To add loss-less transmission, he might add the NAKACK protocol (which also weeds out duplicates). Now messages sent from a sender are always received by the recipients, but the order in which they will be received is undefined. Therefore, the user might choose to add the FIFO layer to impose per/sender ordering. If ordering should be imposed over all the senders, then the TOTAL protocol may be added. The Group Membership Service (GMS) and FLUSH protocols provide for group membership: they allow the user to register a callback function that will be invoked whenever the membership changes, e.g. a member joins, leaves or crashes. In the latter case, a failure detector (FD) protocol is needed by the GMS to announce crashed members. If new members want to obtain the current state of existing members, then the STATE_TRANSFER protocol has to be present in this custom-made stack. Finally, the user may want to encrypt messages, so he adds the CRYPT protocol (which encrypts/decrypts messages and takes care of re-keying using a key distribution toolkit).

Using composition of protocols (each taking care of a different aspect) to build reliable multicast communication has the benefit that

- users of a stack only pay for the protocols they use and
- Since protocols are independent of each other, they can be modified, replaced or new ones can be added, improving modularity and maintainability.

The Java Groups API

The API of Java Groups is very simple (similar to a UDP socket) and is always the same, regardless of how the underlying protocol stack is composed. To be able to send/receive messages, a *channel* has to be created. The reliability of a channel is specified as a string, which then causes the creation of the underlying protocol stack. The example below creates a channel and sends/receives 1 message:

To join a group, **connect ()** is called. It returns when the member has successfully joined the group named "My Group", or when it has created a new group (if it is the first member).

Then a message is sent using the **Send ()** method. A message contains the receiver's address ('null' = all group members), the sender's address ('null', will be filled in by the protocol stack before sending the message) and a byte buffer. In the example, the String object "Hello world" is set to be the message's contents. It is serialized into the message's byte buffer.

Since the message is sent to all members, the sender will also receive it (unless the socket option 'receive own multicasts' is set to 'true'). This is done using the **Receive ()** method.

Finally, the member disconnects from the channel and then closes it. This results in a notification being sent to all members who are registered for membership change notifications.

Program:

Conclusion:

……

……

……

CO's Covered:

……

……

……

Experiment No. 5

Title: *Write A Program to Demonstrate Berkeley Clock Synchronization Algorithm.*

Rubric	***Score (0 to 4)***
Delivery	
Understanding	
Readability	
Discipline	
Total	

Performed On: _____

Sign: _____

Experiment No. 5

Distributed Computing Lab (DCL)

Aim: Write A Program to Demonstrate Berkeley Clock Synchronization Algorithm.

Software: jdk1.8.0, Eclipse.

Pre-Lab Questions:

1. What are logical and physical clock?
2. Explain Berkeley clock synchronization algorithm?

Post-Lab Questions:

1. What is Query-poll?
2. What is response and adjust operation?

Theory:

The Berkeley algorithm is a method of clock synchronization in distributed computing which assumes no machine has an accurate time source. It was developed by Gusella and Zatti at the University of California, Berkeley in 1989 and like Cristian's algorithm is intended for use within intranets.

Algorithm:

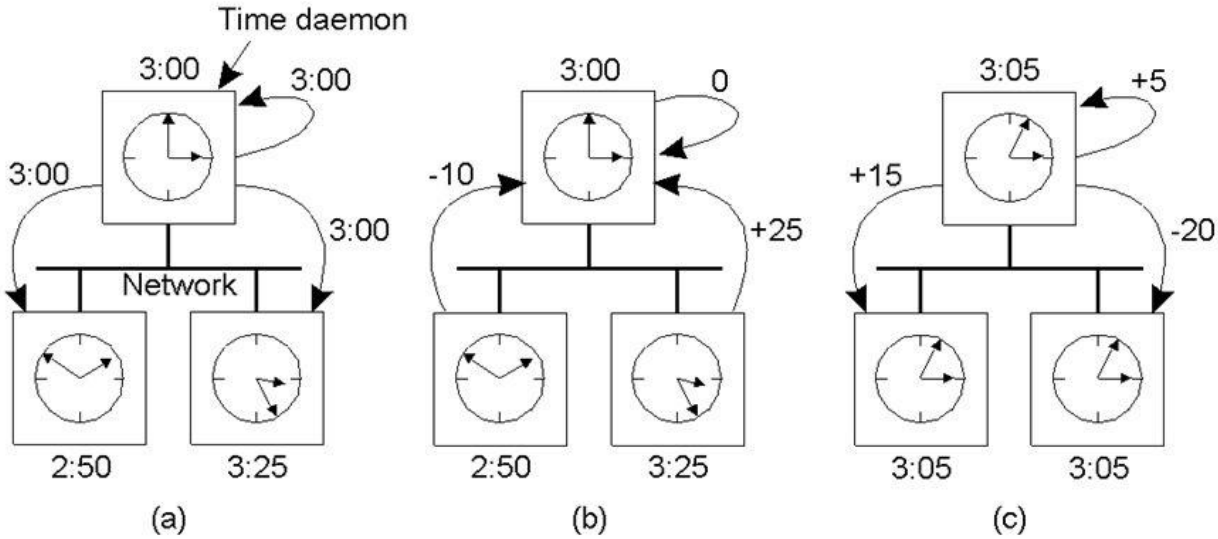
Unlike Cristian's algorithm, the server process in the Berkeley algorithm, called the master, periodically polls other slave processes. Generally speaking, the algorithm is:

1. A master is chosen via an election process such as Chang and Roberts's algorithm.
2. The master polls the slaves who reply with their time in a similar way to Cristian's algorithm.
3. The master observes the round-trip time (RTT) of the messages and estimates the time of each slave and its own.
4. The master then averages the clock times, ignoring any values it receives far outside the values of the others.
5. Instead of sending the updated current time back to the other process, the master then sends out the amount (positive or negative) that each slave must adjust its clock. This avoids further uncertainty due to RTT at the slave processes.

With this method the average cancels out individual clock's tendencies to drift. Gusella and Zatti released results involving 15 computers whose clocks were synchronized to within about 20-25 milliseconds using their protocol. Computer systems normally avoid rewinding their clock when they receive a negative clock alteration from the master. Doing so would break the property of monotonic time, which is a fundamental assumption in certain algorithms in the system itself or in programs such as make. A simple solution to this problem is to halt the clock for the duration specified by the master, but this simplistic solution can also

cause problems, although they are less severe. For minor corrections, most systems slow the clock (known as "clock slew"), applying the correction over a longer period of time.

The Berkeley Algorithm



a) The time daemon asks all the other machines for their clock values
b) The machines answer
c) The time daemon tells everyone how to adjust their clock

11

Program:

Conclusion:

……

……

……

CO's Covered:

……

……

……

Experiment No. 6

Title: *Write a program to demonstrate Bully Election Algorithm.*

Rubric	***Score (0 to 4)***
Delivery	
Understanding	
Readability	
Discipline	
Total	

Performed On: _____

Sign: _____

Experiment No. 6

Distributed Computing Lab (DCL)

Aim: Write a program to Demonstrate Bully Election Algorithm.

Software: jdk1.8.0, Eclipse.

Pre-Lab Questions:

1. What is Election algorithm?
2. Explain Bully Election algorithm?

Post-Lab Questions:

1. Compare different Election algorithm?
2. What is coordinator?

Theory:

The Bully Algorithm was devised by Garcia-Molina in 1982. When a process notices that the coordinator is no longer responding to requests, it initiates an election. Process P, holds an election as follows:

1) P sends an ELECTION message to all processes with higher numbers.
2) If no one responds, P wins the election and becomes coordinator.
3) If one of the higher-ups' answers, it takes over. P's job is done.

At any moment, a process can get an ELECTION message from one of its lower-numbered colleagues. When such a message arrives, the receiver sends an OK message back to the sender to indicate that it is alive and will take over. The receiver then holds an election, unless it is already holding one. Eventually, all processes give up but one and that one is the new coordinator. It announces its victory by sending all processes a message telling them that starting immediately it is the new coordinator.

If a process that was previously down comes back up, it holds an election. If it happens to be the highest-numbered process currently running, it will win the election and will take over the coordinator's job. Thus the biggest guy in town always wins, hence the name "Bully Algorithm".

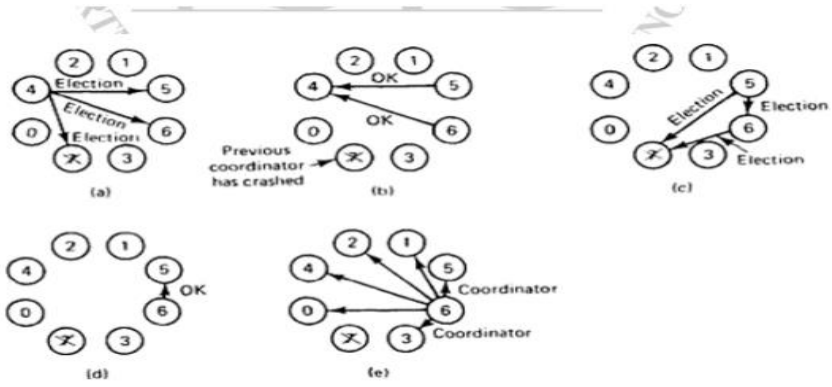


Fig: Bully Election Algorithm

Program:

Conclusion:

……

……

……

CO's Covered:

……

……

……

Experiment. No. 7

Title: *Write a Program to Demonstrate Token Ring Algorithm for Distributed Mutual Exclusion.*

Rubric	***Score (0 to 4)***
Delivery	
Understanding	
Readability	
Discipline	
Total	

Performed On: _____

Sign: _____

Experiment No. 7

Distributed Computing Lab (DCL)

Aim: Write a Program to Demonstrate Token Ring Algorithm for Distributed Mutual Exclusion.

Software: jdk1.8.0, Eclipse.

Pre-Lab Questions:

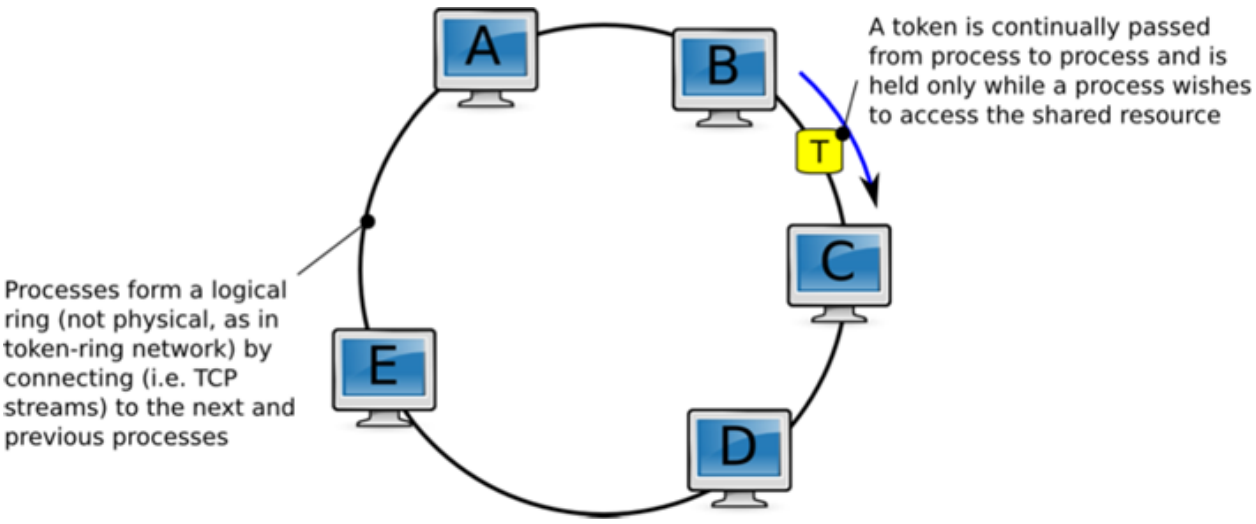
1. What is Distributed mutual exclusion?
2. What is Token Ring algorithm?

Post-Lab Questions:

1. What are different problems of token?
2. Compare different distributed mutual exclusion algorithm.

Theory:

Token ring is one among mutual exclusion algorithm in distributed systems that does not involve with a server or coordinator. A token is passed from one node to another node in a logical order. A node received token has a proper right to enter the critical section. If a node being holding token does not require to enter critical section or access a shared resource, it just simply passes the token to the next node.



A completely different approach to achieving mutual exclusion in a distributed system. Here we have a bus network (for e.g. Ethernet).A logical ring is constructed in which each process is assigned a position in the ring. The ring positions may be allocated in numerical order of network addresses or some other means.

It does not matter what the ordering is.All that matters is that each process knows who is next in line after itself. When the ring is initialized, process *A* (*say*) is given a ***token***. The token circulates around the ring. It is passed from process *k* to process *k+1* in point-to-point messages. When a process acquires the token from its neighbour. After it has exited, it passes the token along the ring. It is not permitted to enter a second CS using the same token. If a process is handed the token by its neighbour and is not interested in entering a CS.

Token Ring algorithm:

- Initialization
  - Process 0 gets token for resource R
- Token circulates around ring
  - From P_i to $P_{(i+1)}$mod N
- When process acquires token
  - Checks to see if it needs to enter critical section
  - If no, send token to neighbor
  - If yes, access resource

Hold token until done

- Only one process at a time has token
  - Mutual exclusion guaranteed
- Order well-defined
  - Starvation cannot occur
- If token is lost (e.g. process died)
  - It will have to be regenerated
- Does not guarantee FIFO order

Program:

Conclusion:

……

……

……

CO's Covered:

……

……

……

Experiment No. 8

Title: *Write a program to Simulate Load Balancing Algorithm Using Java.*

Rubric	***Score (0 to 4)***
Delivery	
Understanding	
Readability	
Discipline	
Total	

Performed On: _____

Sign: _____

Experiment No. 8

Distributed Computing Lab (DCL)

Aim: Write A Program to Simulate Load Balancing Algorithm Using Java.

Software: jdk1.8.0, Eclipse.

Pre-Lab Questions:

1. What is Election algorithm?
2. Explain Bully Election algorithm?

Post-Lab Questions:

1. Compare different Election algorithm?
2. What is coordinator?

Theory:

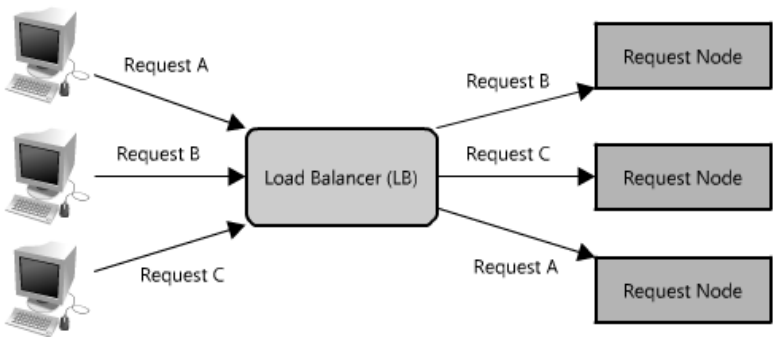
LOAD BALANCING:

Load balancing is the way of distributing load units (jobs or tasks) across a set of processors which are connected to a network which may be distributed across the globe. The excess load or remaining unexecuted load from a processor is migrated to other processors which have load below the threshold load. Threshold load is such an amount of load to a processor that any load may come further to that processor. In a system with multiple nodes there is a very high chance that some nodes will be idle while the other will be overloaded. So, the processors in a system can be identified according to their present load as heavily loaded processors (enough jobs are waiting for execution), lightly loaded processors (less jobs are waiting) and idle processors (have no job to execute). By load balancing strategy it is possible to make every processor equally busy and to finish the works approximately at the same time.

A load balancing operation consists of three rules. These are location rule, distribution rule and selection rule. The selection rule works either in preemptive or in non- preemptive fashion. The newly generated process is always picked up by the non-preemptive rule while the running process may be picked up by the preemptive rule. Preemptive transfer is costly than non-preemptive transfer which is preferable. However preemptive transfer is more excellent than non-preemptive transfer in some instances.

Benefits of Load Balancing:

1. Load balancing improves the performance of each node and hence the overall system performance. Load balancing reduces the job idle time.
2. Small jobs do not suffer from long starvation.
3. Maximum utilization of resources
4. Response time becomes shorter.
5. Higher throughput
6. Higher reliability
7. Low cost but high gain
8. Extensibility and incremental growth


Request A
Request B
Request C
Load Balancer (LB)
Request B
Request C
Request A
Request Node
Request Node
Request Node

Program:

Conclusion:

……

……

……

CO's Covered:

……

……

……

Experiment No. 9

Title: *Write A Program to Demonstrate Steps of CORBA Application Using Java.*

Rubric	***Score (0 to 4)***
Delivery	
Understanding	
Readability	
Discipline	
Total	

Performed On: _____

Sign: _____

Experiment No. 9

Distributed Computing Lab (DCL)

Aim: Write a Program to Demonstrate Steps of CORBA Application Using Java

Software: jdk1.8.0, Eclipse.

Pre-Lab Questions:

1. What is CORBA? What does it do?
2. What are the CORBA application development processes?

Post-Lab Questions:

1. Explain what is CORBA good for?
2. Which protocol is used for invoking methods on CORBA objects over the internet?

Theory:

A First CORBA Application

This section introduces the JServer CORBA application development process. It tells you how to write a simple but useful program that runs on a client system, connects to Oracle using IIOP, and invokes a method on a CORBA server object that is activated and runs inside an Oracle8i Java VM. This section addresses only the purely mechanical aspects of the development process. Application developers know that for large-scale applications the design is a crucially important step. See "For More Information" for references to documents on CORBA design.

The CORBA application development process has seven phases:

1. Design and write the object interfaces.

2. Generate stubs and skeletons, and other required support classes.

3. Write the server object implementations.

4. Use the client-side Java compiler to compile both the Java code that you have written, and the Java classes that were generated by the IDL compiler. Generate a JAR file to contain the classes and any other resource files that are needed.

5. Publish a name for the directly accessible objects with the Cos Naming service, so you can access them from the client program.

6. Write the client side of the application. This is the code that will run outside of the Oracle8i data server, on a workstation or PC.

7. Compile the client code using the JDK Java compiler.

8. Load the compiled classes into the Oracle8i database, using the load java tool and specifying the JAR file as its argument. Make sure to include all generated classes, such as stubs and skeletons. (Stubs are required in the server when the server object acts as a client to another CORBA object.)

Program:

Conclusion:

……

……

……

CO's Covered:

……

……

……

Experiment No. 10

Title: *Write a program to implement Chandy -Misra_Hass distributed deadlock detection algorithm.*

Rubric	***Score (0 to 4)***
Delivery	
Understanding	
Readability	
Discipline	
Total	

Performed On: _____

Sign: _____

Experiment No. 10

Distributed Computing Lab (DCL)

Aim: Write a program to implement Chandy -Misra_Hass distributed deadlock detection algorithm.

Software: jdk1.8.0, Eclipse

Pre-Lab Questions:

1. What is deadlock management?
2. How Chandy -Misra_Hass algorithm works?

Post-Lab Questions:

1. What will happen if process P_1 requests one additional instance of resource type A and two instances of resource type C?

Theory:

Chandy-Misra-Haas's distributed deadlock detection algorithm is an edge chasing algorithm to detect deadlock in distributed systems.

In edge chasing algorithm, a special message called probe is used in deadlock detection. A probe is a triplet (i, j, k) which denotes that process Pi has initiated the deadlock detection and the message is being sent by the home site of process Pj to the home site of process Pk.

The probe message circulates along the edges of WFG to detect a cycle. When a blocked process receives the probe message, it forwards the probe message along its outgoing edges in WFG. A process Pi declares the deadlock if probe messages initiated by process Pi returns to itself.

Other terminologies used in the algorithm:

Dependent process:

A process Pi is said to be dependent on some other process Pj, if there exists a sequence of processes Pi, Pi1, Pi2, Pi3…, Pim, Pj such that in the sequence, each process except Pj is blocked and each process except Pi holds a resource for which previous process in the sequence is waiting.

Locally dependent process:

A process Pi is said to be locally dependent on some other process Pj if the process Pi is dependent on process Pj and both are at same site.

Data structures:

A boolean array, dependenti. Initially, dependenti[j] is false for all value of i and j. dependenti[j] is true if process Pj is dependent on process Pi.

Algorithm:

Process of sending probe:

1. If process Pi is locally dependent on itself then declare a deadlock.

2. Else for all Pj and Pk check following condition:

- (a). Process Pi is locally dependent on process Pj
- (b). Process Pj is waiting on process Pk
- (c). Process Pj and process Pk are on different sites.

If all of the above conditions are true, send probe (i, j, k) to the home site of process Pk.

On the receipt of probe (i, j, k) at home site of process Pk:

1. Process Pk checks the following conditions:

- (a). Process Pk is blocked.
- (b). dependentk[i] is false.
- (c). Process Pk has not replied to all requests of process Pj

If all of the above conditions are found to be true then:

1. Set dependentk[i] to true.

2. Now, If k == i then, declare the Pi is deadlocked.

3. Else for all Pm and Pn check following conditions:

- (a). Process Pk is locally dependent on process Pm and
- (b). Process Pm is waiting upon process Pn and
- (c). Process Pm and process Pn are on different sites.

4. Send probe (i, m, n) to the home site of process Pn if above conditions satisfy.

Thus, the probe message travels along the edges of transaction wait-for (TWF) graph and when the probe message returns to its initiating process then it is said that deadlock has been detected.

Performance:

Algorithm requires at most exchange of m(n-1)/2 messages to detect deadlock. Here, m is number of processes and n is the number of sites.

The delay in detecting the deadlock is O(n).

Advantages:

There is no need for special data structure. A probe message, which is very small and involves only 3 integers and a two dimensional boolean array dependent is used in the deadlock detection process.

At each site, only a little computation is required and overhead is also low

Unlike other deadlock detection algorithm, there is no need to construct any graph or pass nor to pass graph information to other sites in this algorithm.

Algorithm does not report any false deadlock (also called phantom deadlock).

Disadvantages:

The main disadvantage of distributed detection algorithms is that all sites may not be aware of the processes involved in the deadlock which makes resolution difficult. Also, proof of correction of the algorithm is difficult.

It may detect a false deadlock if there is a delay in message passing or if a message is lost. This can result in unnecessary process termination or resource preemption.

It may not be able to detect all deadlocks in the system, especially if there are hidden deadlocks or if the system is highly dynamic.

It is complex and difficult to implement correctly. It requires careful coordination between the processes, and any errors in the implementation can lead to incorrect results.

It may not be scalable to large distributed systems with a large number of processes and resources. As the size of the system grows, the overhead and complexity of the algorithm also increase.

Program:

Conclusion:

……

……

……

CO's Covered:

……

……

……

Experiment No. 11

***Title:* Distributed Shared Memory**

Rubric	***Score (0 to 4)***
Delivery	
Understanding	
Readability	
Discipline	
Total	

Performed On: _____

Sign: _____

Experiment No. 11

Distributed Computing Lab (DCL)

Aim: Distributed Shared Memory

Software: jdk1.8.0, Eclipse

Pre-Lab Questions:

1. What is Distributed Shared Memory (DSM), and how does it differ from traditional shared memory systems?
2. What are the primary goals of using DSM in distributed systems?

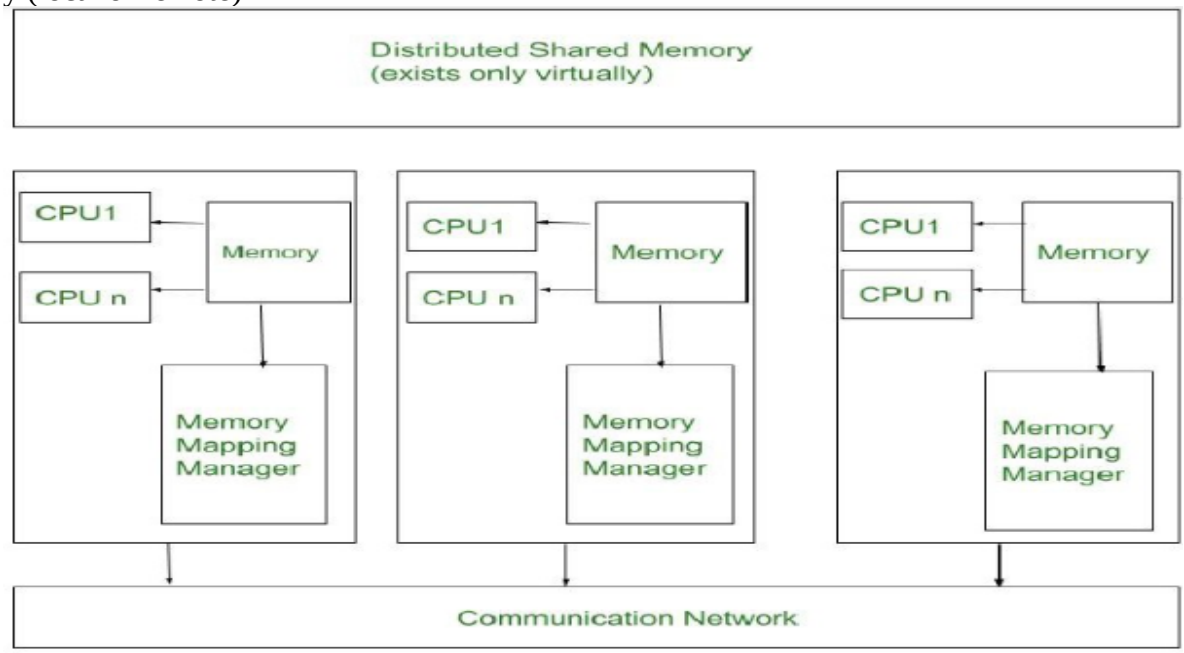
Post-Lab Questions:

1. What approach did you use to implement the DSM system in the lab?
2. How did you ensure consistency across the shared memory in your implementation?

Theory:

Distributed Shared Memory (DSM) implements the distributed systems shared memory model in a distributed system, that hasn't any physically shared memory. Shared model provides a virtual address area shared between any or all nodes. To beat the high forged of communication in distributed system. DSM memo, model provides a virtual address area shared between all nodes. systems move information to the placement of access. Information moves between main memory and secondary memory (within a node) and between main recollections of various nodes.

Every Greek deity object is in hand by a node. The initial owner is that the node that created the object. Possessions will amendment as the object moves from node to node. Once a method accesses information within the shared address space, the mapping manager maps shared memory address to physical memory (local or remote)



DSM permits programs running on separate reasons to share information while not the software engineer having to agitate causation message instead underlying technology can send the messages to stay the DSM consistent between computes. DSM permits programs that won't to treat constant laptop to be simply tailored to control on separate reason. Programs access what seems to them to be traditional memory.
Hence, programs that Pine Tree State DSM square measure sometimes shorter and easier to grasp than programs that use message passing. But DSM isn't appropriate for all things. Client-server systems square measure typically less suited to DSM, however, a server is also wont to assist in providing DSM practicality for information shared between purchasers.

Architecture of Distributed Shared Memory (DSM):

Every node consists of 1 or additional CPU's and a memory unit. High-speed communication network is employed for connecting the nodes. A straightforward message passing system permits processes on completely different nodes to exchange one another.

Memory mapping manager unit:

Memory mapping manager routine in every node maps the native memory onto the shared computer storage. For mapping operation, the shared memory house is divided into blocks. Information caching may be a documented answer to deal with operation latency. DMA uses information caching to scale back network latency. the most memory of the individual nodes is employed to cache items of the shared memory house.

Memory mapping manager of every node reads its native memory as an enormous cache of the shared memory house for its associated processors. The bass unit of caching may be a memory block. Systems that support DSM, information moves between secondary memory and main memory also as between main reminiscences of various nodes.

Communication Network Unit:

Once method access information within the shared address house mapping manager maps the shared memory address to the physical memory. The mapped layer of code enforced either within the operating kernel or as a runtime routine.

Physical memory on every node holds pages of shared virtual–address house. Native pages area unit gift in some node's memory. Remote pages in some other node's memory.

Program:

Conclusion:

……

……

CO's Covered:

……

……

……

ASSIGNMENTS

Assignment No. 01

Rubric	***Score (0 to 4)***
Delivery	
Understanding	
Readability	
Discipline	
Total	

Performed On: _____

Sign: _____

Assignment No. 02

Rubric	***Score (0 to 4)***
Delivery	
Understanding	
Readability	
Discipline	
Total	

Performed On: _____

Sign: _____