

# **Data Link Layer (L2 Routing) Implementation (TCP/ IP Stack Functionality)**

A Mid-Term Report  
in Partial Fulfilment of the Requirements  
for the Course of  
**Minor Project - I**  
In  
Third year – Fifth Semester of  
**Bachelor of Technology**  
Specialization in  
**DevOps and CCVT.**

Under the guidance of

**Ms. Avita Katal**  
**Assistant Professor – Senior Scale**  
**Department of Virtualization**

By

**500067035**  
**500067409**  
**500068293**

**R171218058**  
**R110218131**  
**R110218107**

**Kshitiz Saini**  
**Saloni Saxena**  
**Pratyusha Agarwal**



**UNIVERSITY WITH A PURPOSE**

Department of Cybernetics and Virtualization

**SCHOOL OF COMPUTER SCIENCE**

**UNIVERSITY OF PETROLEUM AND ENERGY STUDIES**

Bidholi Campus, Energy Acres, Dehradun – 248007

**July, 2020**



## School of Computer Science

University of Petroleum & Energy Studies, Dehradun

### **Project Proposal Approval Form (2019-2020)**

**Minor**

1

**Project Title:**

Data Link Layer (L2 Routing) Implementation (TCP/ IP Stack Functionality)

#### **Abstract**

In today's world, fast and efficient communication on the network between the sender and receiver is very important. For this communication, data is converted into packets and sent over the network using various routing algorithms. In a network or over multiple networks, routing refers to the process of determining a path for a packet to travel from. L2 Routing is based on the concept of Data Link layer and happens when data is to be transferred between the same subnet. In this project, we aim to implement the functionality of Data Link Layer like creating ARP tables, L2 Routing, MAC learning & L2 Switching.

*Keywords: Network Routing, TCP/IP, Network Graph, ARP, L2 Routing, L2 Switching, Packet, Socket Programming, Topology, MAC Learning*

# **TABLE OF CONTENT**

	<b>TOPIC</b>	<b>PAGE NO.</b>
1)	Introduction	1-2
2)	Literature Review	3
3)	Problem Statement	4
4)	Objectives	5
5)	Methodology	6
6)	Implementation	7-11
	6.1 Implementing Generic Graph	7
	6.2 Implementing Network Topology	8
	6.3 Integrating CLI	9
	6.4 Adding Communication Properties	10
	6.5 Data Flow Diagram	11
7)	System Requirements	12
8)	Code	12
9)	Pert Chart	13
10)	References	14

# 1. INTRODUCTION

This project focuses on implementing Layer 2 i.e. Data-Link Layer of TCP/IP Stack. The project mainly focuses on Graph Data Structures highlighting the features of **Generic Graph Topology**. The project will use routers, switches, links as well as nodes to depict the overall functionality of network topology. The entire packet journey (on sending and receiving machines) through the Data Link Layer of TCP/IP Model along with a minimal use of **Socket Programming** will be shown in this project.

The project focuses on the practical implementation of L2 Routing, L2 Switching [2] and VLAN based forwarding [2] where the concepts of ARP Resolution, MAC Learning and Forwarding will be used. Alongside, implementing the concept of Routing and Switching Algorithms, GL Threads, Timers, CLI Integration will also be in consideration.

Networks are treated as graphs  $G(V, E)$  consisting of  $V$  vertices/nodes and  $E$  edges.

- The graphs are bi-directional (undirected) because data can flow on both sides.
- Weighted graphs are used where weight indicates distance/cost etc.
- Connected graph is used as all nodes in a network are connected with each other.

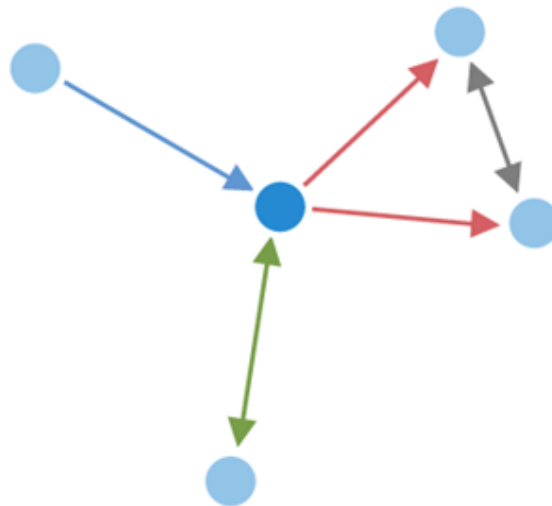


Figure 1 [4] Network  $G(V, E)$  with  $V = 5$  and  $E = 5$

Figure 1 depicts a graph containing 5 vertices and 5 edges. It is a connected and undirected graph that can be used to represent a network.

Routing Algorithm [1] determines the path selected by a packet to reach its destination. It must decide within each intermediate router which output channel/s are to be selected for incoming packets.

Switching Algorithm [3] determines how network resources are allocated for data transmission, i.e. how and when the input channel is connected to the output channel selected by the routing algorithm. It is the actual mechanism that removes data from input channels and places them on output channels.

## **2. LITERATURE REVIEW**

- 1)** In [1], the author David C. Plummer has talked about Address Routing Protocol and its importance in transmitting the packet to a target host. The author explained the concept how ARP allows dynamic distribution of information to build tables that is needed to convert the given address into 48 bits ethernet address. In packet generation journey the concept of converting <protocol type, target protocol address> to 48-bit ethernet address is discussed in detail and also when to transmit or discard the packet. In packet reception, author has discussed about the algorithm through which a packet goes when address resolution packet is received. The author also discussed about network monitoring and debugging and some related issues.
- 2)** In [2], the authors Mario Ernesto, Gomez-Romero, Mario Reyes-Ayala, Edgar Alejandro Andrade-González, Jose Alfredo Tirado-Mendez have explained the physical and logical architecture of the network to be designed and implemented having VLAN. The main goal of this paper is to increase the security level of the LAN, in order to reduce the access to undesirable sites and to avoid the presence of hackers in the net, for the same purpose they have explained how VLANs are connected to port number and then the connection of VLANs with MAC as it eliminates the concept of connection with port numbers. The paper also focuses on the layer 3 based VLAN and policy based VLAN. As a result of this work, the solution implemented can be changed according to current organization requirements. This is especially useful, because the workstations can be easily relocated if necessary.
- 3)** In [3], the authors F. Balay and MorethanIP explained what L2 Switching is and how it can be used to connect number of Ethernet LANs to form a large network, the architecture of an L2 Switch was explained as a System on Programmable Chip (SPOC). MorethanIP focuses on the functional description of the L2 Switch and F. Balay explained concepts like Frame Switching, MAC Address Learning, lookup table entry ageing, port migration as well as hash code and table management. At last, the authors concluded that L2 Switch is an essential building block in today's telecommunication system.

### **3. PROBLEM STATEMENT**

The transmission of data from one device to another without the help of TCP/IP stack is only feasible for a geographically small area but when it comes to worldwide solutions, it is not very appropriate. The problem without the TCP/IP Layer is to transmit data through the shortest path using features like creating ARP Tables and MAC Learning for an optimal resolution. Another problem is to create a network to share the data and a new medium has to be used for transmission every time.

## **4. OBJECTIVES**

The main objective is to implement the data link layer.

Sub-objectives for the project are:

- Developing a generic graph topology
- Tackling the issues of thrashing, collision and reducing broadcast domain.
- Implementing ARP, L2 routing, MAC learning and L2 switching.



## 5. METHODOLOGY

Agile methodology of software development will be followed for the proposed project. The project is divided into 12 sprints where the sprint 7 and 8 will consist of parallel development by different members of the team. Each sprint is provided ample time to complete itself as well as to maintain the product's backlog (if any). The project can accommodate changes if required at any stage of the project. The sprints 1, 2 and 3 are specifically for requirement analysis and designing of the project. One sprint is specifically designed for setting the environment like maintaining the Version Control (Git in our case) and MakeFile. Each development sprint is followed by Unit Testing and an Integration Testing at the end. Sprints are also designed for the reviewing as well as retrospection part. After all the sprints the project is expected to complete by November 30.

The algorithms of sub-processes are defined below:

### ARP Resolution:

- 1) ARP Request message is broadcasted to all devices on the network.
- 2) Nodes determine if they are the intended target.
  - a) If nodes are the intended target, they proceed to STEP 3
  - b) If they are not the intended target, they discard the request and proceed to STEP 4
- 3) Intended device sends ARP Response to the sender in unicast mode.
- 4) END

### L2 Switching:

- 1) When the frame reaches the L2 switch, it updates its table with the MAC address of the source machine corresponding to the port number at which the frame is received.
- 2) L2 switch forwards the packet.
  - a) If it does not contain the MAC address of the destination machine corresponding to any port, it floods the packet.
  - b) If it contains the MAC address of the destination machine, step 3 is followed.
- 3) Frame is received only to the intended machine.
- 4) END

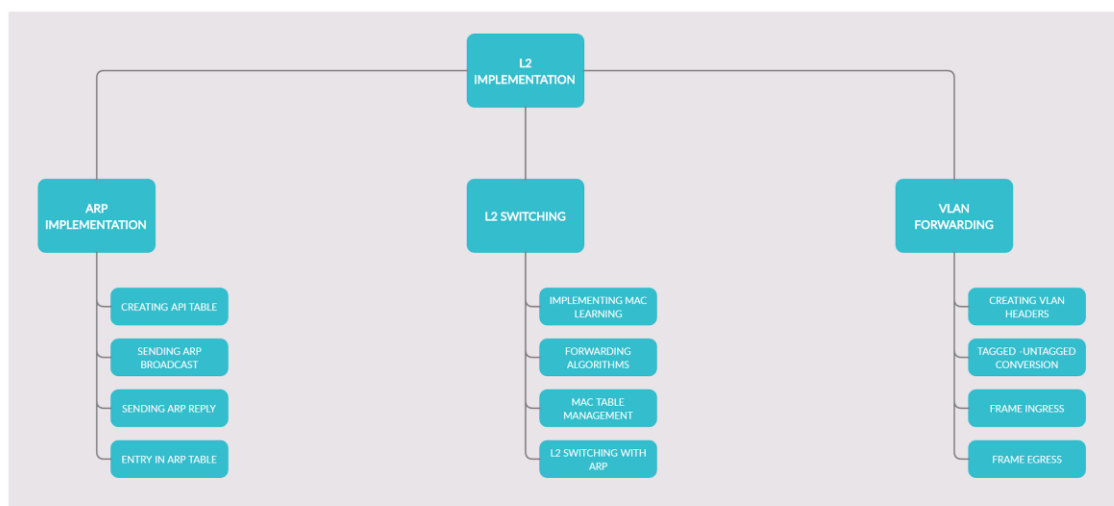


Figure 2 Different phases of the development part

## 6. IMPLEMENTATION

### 6.1 Generic Graph

We have used Linked List and Graph data structures to represent the Network Topology. We are using the Generic Graphs in our case which is a data structure that can be made once and can be updated as many times according to ones need.

The functionality of network graph defined is:

- Graph is defined as collection of network nodes.
- Each network node has its own interface.
- Interfaces are connected to each other via links.

```
kshiltzaini113@kshiltzaini113: ~/repo/TCP-IP-Stack-Functionality/Project_Implementation
kshiltzaini113@kshiltzaini113:~/repo/TCP-IP-Stack-Functionality/Project_Implementation$ make
gcc -g -c main.c -o main.o
gcc -g -c -I linkedlist linkedlist/linkedlist.c -o linkedlist/linkedlist.o
gcc -g -c -I . networkgraph.c -o networkgraph.o
gcc -g -c -I . network_topology.c -o network_topology.o
gcc -g main.o linkedlist/linkedlist.o networkgraph.o network_topology.o -o main.sh
kshiltzaini113@kshiltzaini113:~/repo/TCP-IP-Stack-Functionality/Project_Implementation$ ./main.sh
Network Topology Name : Minor-1 Example-1

Network Node Name : router_2
Local Node : router_2, Interface Name : eth0/3, Neighbour Node : router_1, Traversal Cost : 1
Local Node : router_2, Interface Name : eth0/5, Neighbour Node : router_0, Traversal Cost : 1

Network Node Name : router_1
Local Node : router_1, Interface Name : eth0/1, Neighbour Node : router_0, Traversal Cost : 1
Local Node : router_1, Interface Name : eth0/2, Neighbour Node : router_2, Traversal Cost : 1

Network Node Name : router_0
Local Node : router_0, Interface Name : eth0/0, Neighbour Node : router_1, Traversal Cost : 1
Local Node : router_0, Interface Name : eth0/4, Neighbour Node : router_2, Traversal Cost : 1

kshiltzaini113@kshiltzaini113:~/repo/TCP-IP-Stack-Functionality/Project_Implementation$
```

Figure 3 Generic Graph Implementation

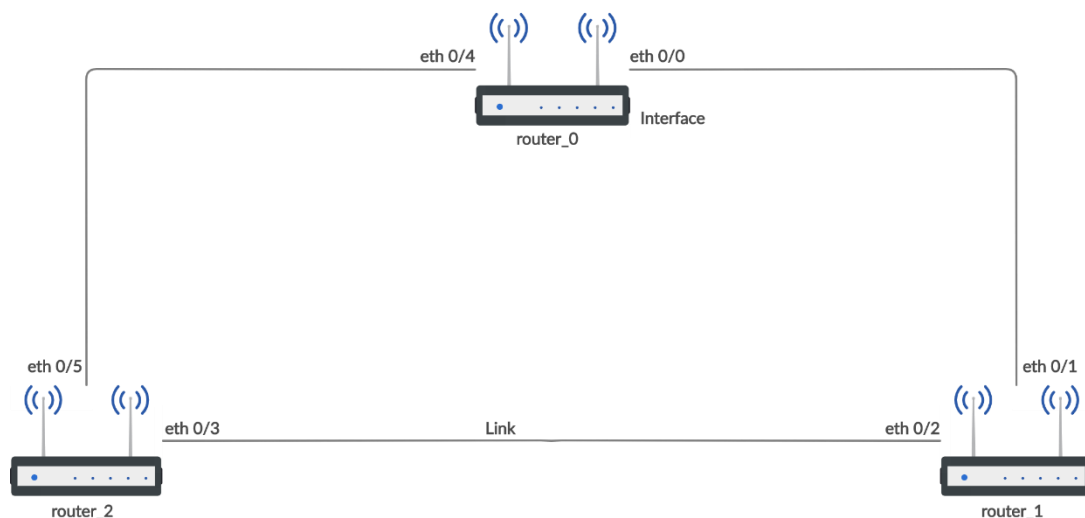


Figure 4 Generic Graph Diagram

## 6.2 Network Topology

After the implementation of Generic Graph was complete, we have used user-defined structures to add network properties like IP, MAC and Subnet Mask.

The functionality of Network properties is:

- Loopback Address provides a unique address to a device.
- Each node is provided with its own unique network properties.
- Links are established between nodes and they have their own network properties.

```
kshltizsaini113@kshltizsaini113: ~/repo/TCP-IP-Stack-Functionality/Project_Implementation
kshltizsaini113@kshltizsaini113:~/repo/TCP-IP-Stack-Functionality/Project_Implementation$ make
gcc -g -c main.c -o main.o
gcc -g -c -I linkedlist/linkedlist.c -o linkedlist/linkedlist.o
gcc -g -c -I . networkgraph.c -o networkgraph.o
gcc -g -c -I . network_topology.c -o network_topology.o
gcc -g -c -I . network.c -o network.o
gcc -g main.o linkedlist/linkedlist.o networkgraph.o network_topology.o network.o -o main.sh
kshltizsaini113@kshltizsaini113:~/repo/TCP-IP-Stack-Functionality/Project_Implementation$ ./main.sh
Topology Name : Minor-1 Example-1
Node Name : router_2
Node Flags : 2 Loopback Address : 122.1.1.2/32
Interface Name : eth0/3
  Neighbour Node : router_2, Local Node : router_1, Traversal Cost : 1
  IP Address : 30.1.1.1/24 MAC Address : 114:111:117:116:101:114
Interface Name : eth0/5
  Neighbour Node : router_2, Local Node : router_0, Traversal Cost : 1
  IP Address : 40.1.1.1/24 MAC Address : 114:111:117:116:101:114
Node Name : router_1
Node Flags : 2 Loopback Address : 122.1.1.1/32
Interface Name : eth0/1
  Neighbour Node : router_1, Local Node : router_0, Traversal Cost : 1
  IP Address : 20.1.1.1/24 MAC Address : 114:111:117:116:101:114
Interface Name : eth0/2
  Neighbour Node : router_1, Local Node : router_2, Traversal Cost : 1
  IP Address : 30.1.1.1/24 MAC Address : 114:111:117:116:101:114
Node Name : router_0
Node Flags : 2 Loopback Address : 122.1.1.0/32
Interface Name : eth0/0
  Neighbour Node : router_0, Local Node : router_1, Traversal Cost : 1
  IP Address : 20.1.1.1/24 MAC Address : 114:111:117:116:101:114
Interface Name : eth0/4
  Neighbour Node : router_0, Local Node : router_2, Traversal Cost : 1
  IP Address : 40.1.1.1/24 MAC Address : 114:111:117:116:101:114
kshltizsaini113@kshltizsaini113:~/repo/TCP-IP-Stack-Functionality/Project_Implementation$
```

Figure 5 Network Topology Implementation

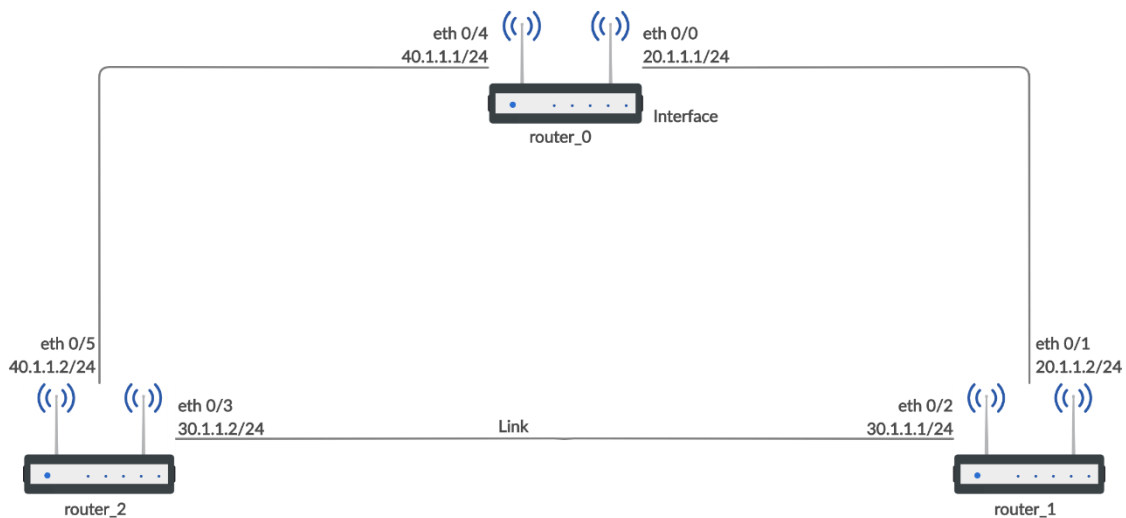


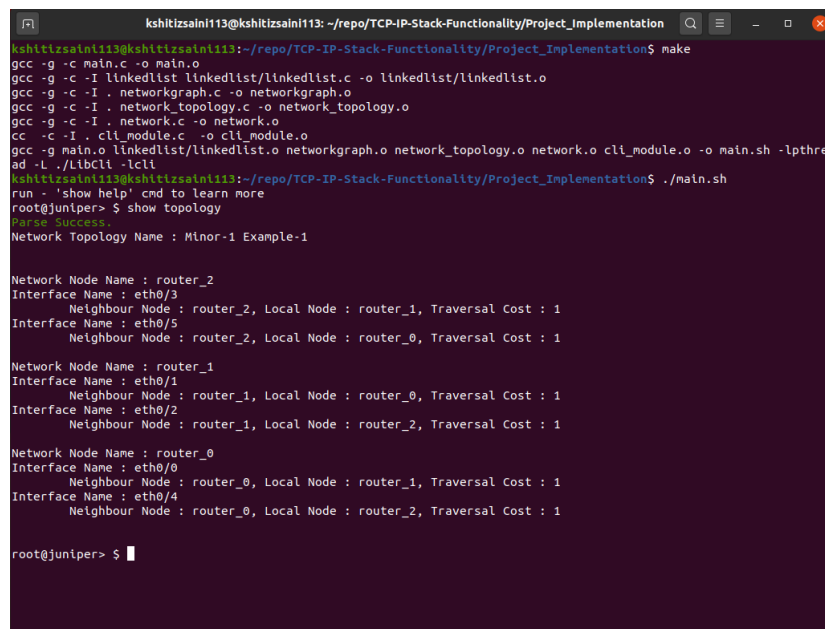
Figure 6 Network Topology Diagram

## 6.3 CLI

CLI was setup using the LibCli library to remove the hepatic issues of menu driven programs and to support sever side systems also. Even CLI based programs consumes less memory.

CLI is responsible for:

- Validating command syntax.
- Validating if leaf node values are compatible with given leaf node expectations.
- Sending command to the project along with leaf node values for command processing.



```
kshiltzaini113@kshiltzaini113: ~/repo/TCP-IP-Stack-Functionality/Project_Implementation
gcc -g -c main.c -o main.o
gcc -g -c -I linkedlist linkedlist/linkedlist.c -o linkedlist/linkedlist.o
gcc -g -c -I . networkgraph.c -o networkgraph.o
gcc -g -c -I . network_topology.c -o network_topology.o
gcc -g -c -I . network.c -o network.o
cc -c -I . cli_module.c -o cli_module.o
gcc -g main.o linkedlist/linkedlist.o networkgraph.o network_topology.o network.o cli_module.o -o main.sh -lpthread -L ./libcli -lcli
kshiltzaini113@kshiltzaini113:~/repo/TCP-IP-Stack-Functionality/Project_Implementation$ ./main.sh
run - 'show help' cmd to learn more
root@juniper> $ show topology
Parse Success.
Network Topology Name : Minor-1 Example-1

Network Node Name : router_2
Interface Name : eth0/3
  Neighbour Node : router_2, Local Node : router_1, Traversal Cost : 1
Interface Name : eth0/5
  Neighbour Node : router_2, Local Node : router_0, Traversal Cost : 1

Network Node Name : router_1
Interface Name : eth0/1
  Neighbour Node : router_1, Local Node : router_0, Traversal Cost : 1
Interface Name : eth0/2
  Neighbour Node : router_1, Local Node : router_2, Traversal Cost : 1

Network Node Name : router_0
Interface Name : eth0/0
  Neighbour Node : router_0, Local Node : router_1, Traversal Cost : 1
Interface Name : eth0/4
  Neighbour Node : router_0, Local Node : router_2, Traversal Cost : 1

root@juniper> $
```

Figure 7 Implementation of CLI

## 6.4 Communication Setup

Communication properties are added to the system for making different nodes available nodes communicate with each other. We have used a minimal socket programming for the same. A view of the system depicting the functionality is defined as,

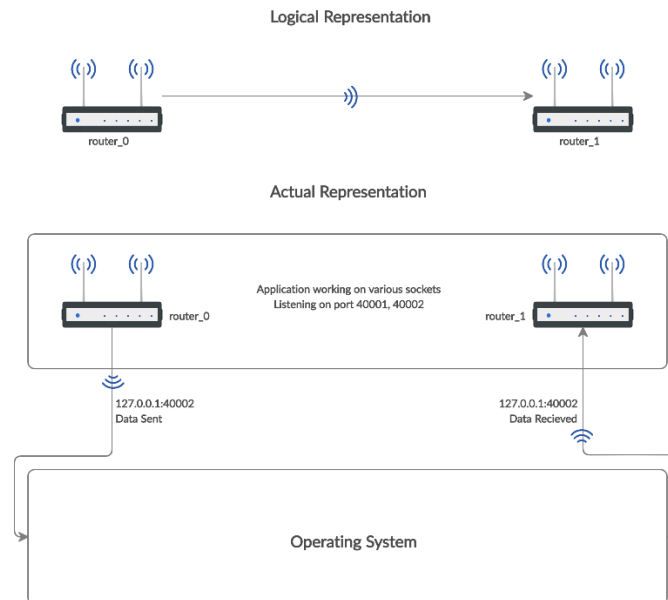
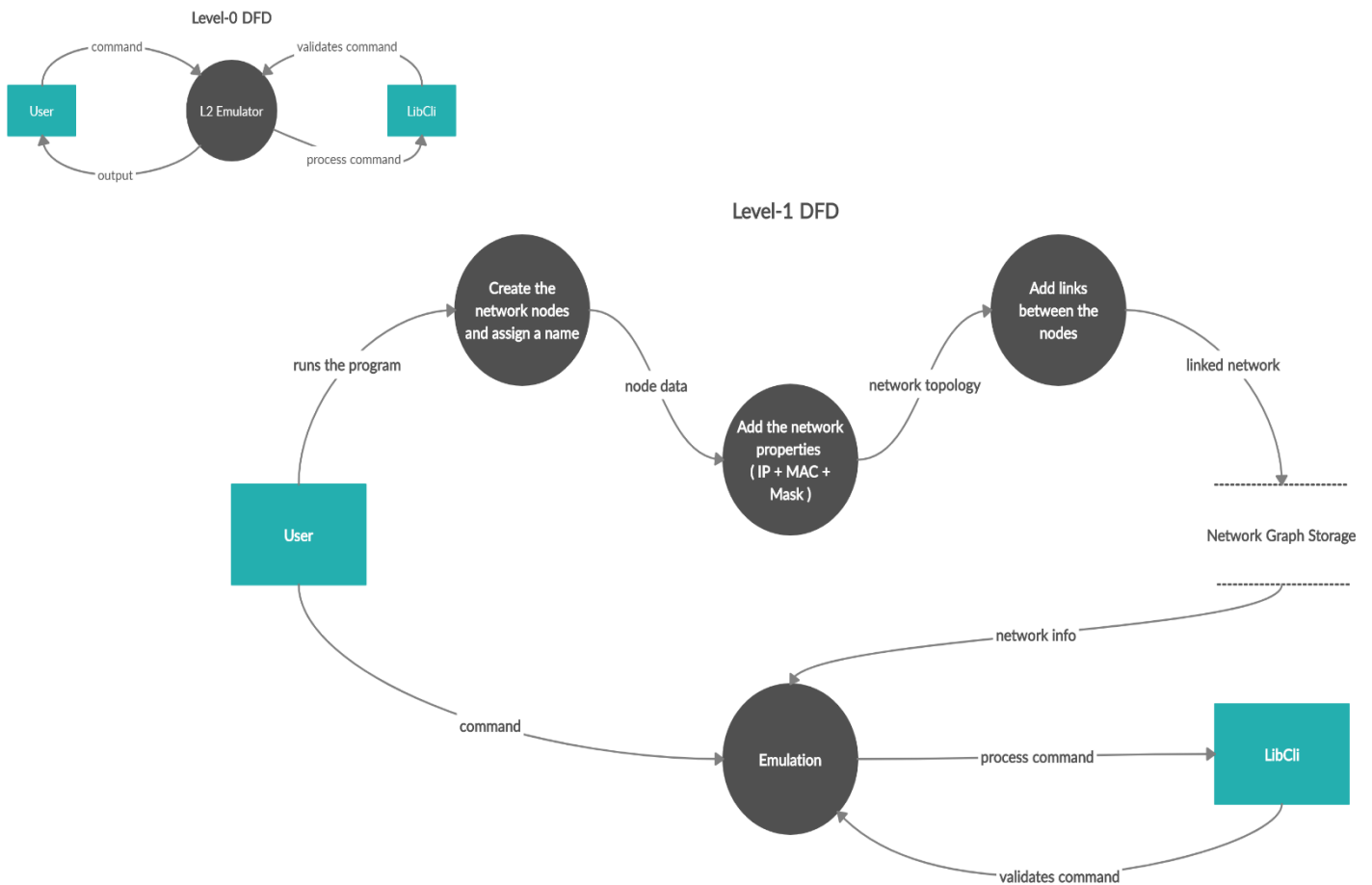


Figure 8 Logical-Actual View Communication

Communication Algorithm:

- 1) Assigning unique UDP port number to each node of the topology.
- 2) Opening a UDP socket for all port numbers.
- 3) Listening on all UDP sockets.
- 4) Nodes communicates by sending data to designation node with IP=127.0.0.1.
- 5) Underlying OS echoes back all data back to the application.
- 6) Handing over the data to the destination port number.
- 7) Using auxiliary information, recipient interface name is known.
- 8) Actual frame is received to the destination node on a given interface.

## 6.5 Data Flow Diagram



## **7. SYSTEM REQUIREMENTS**

### **1. Software**

- GCC Compiler
- Any flavour of Linux

### **2. Hardware**

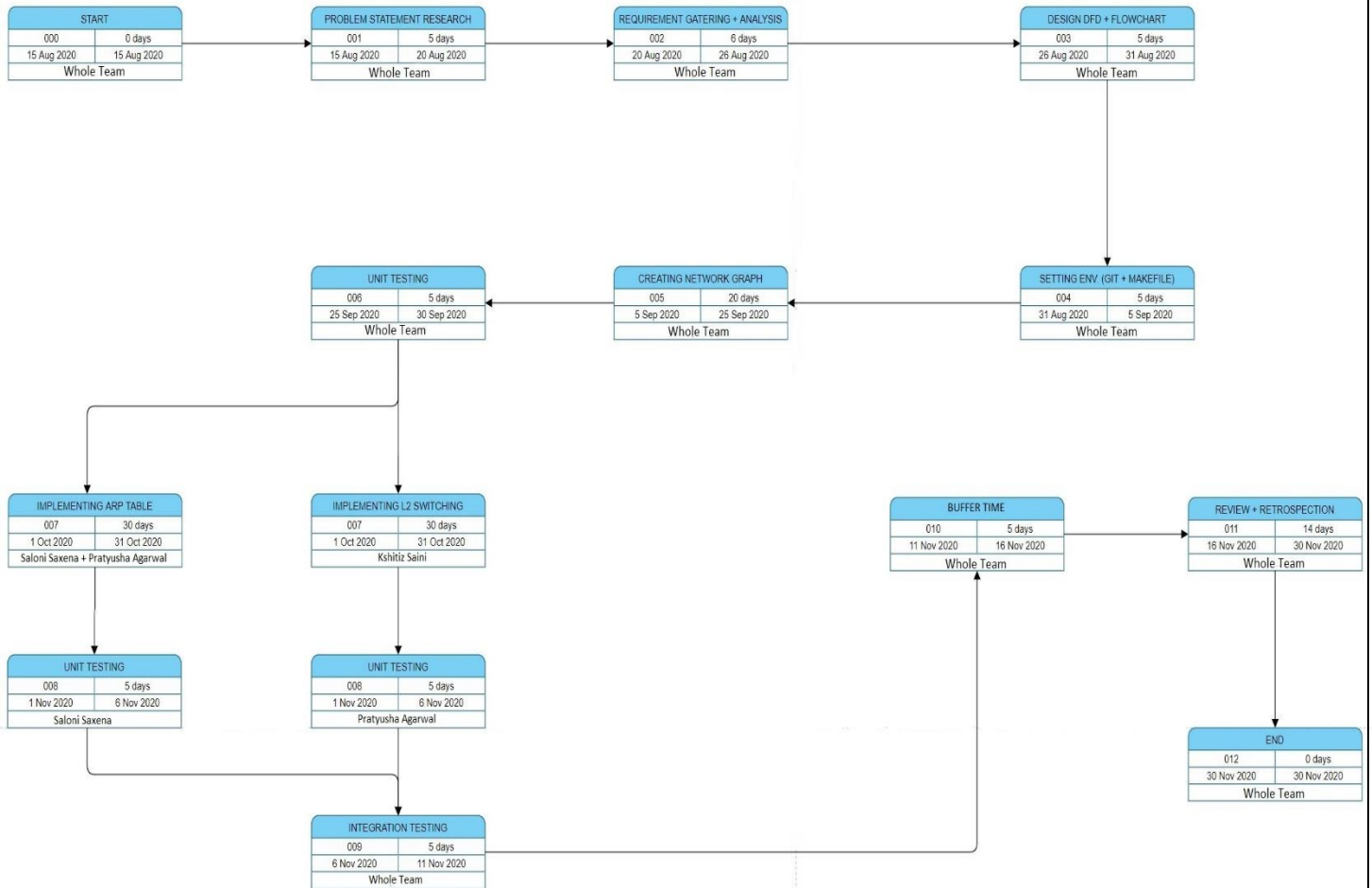
- 512 MB RAM
- i3 5th Generation or above processor

## **8. CODE**

The code for the project is available on GitHub on the repository:

[https://github.com/kshitizsaini113/TCP-IP-Stack-Functionality/tree/master/Project\\_Implementation](https://github.com/kshitizsaini113/TCP-IP-Stack-Functionality/tree/master/Project_Implementation)

## 9. PERT CHART





## **10. REFERENCES**

- [1] C Plummer, “An Ethernet Address Resolution Protocol” : Massachusetts Institute of Technology (MIT), Internet Engineering Task Force, 1982
- [2] M. Ernesto Gomez-Romero, M. Reyes-Ayala, E. Alejandro, Andrade-Gonzalez, J. Alfredo and Tirade-Mendez , “Design and Implementation of a VLAN” : Metropolitan Autonomous University, Cinvestav Applied Computer Conference, 2011
- [3] F. Balay and MorethanIP , “L2 Switch Implementation with programmable logic devices” : Germany
- [4] Socilyzer, “Interpreting Social Network Diagram”: Graph Interpretation, Socilyzer

### **Synopsis Draft verified by**

**Ms. Avita Katal**

Assistant Professor-Senior Scale  
Department of Virtualization  
School of Computer Science



**Project Guide**  
**(Ms. Avita Katal)**

**Dr. Monit Kapoor**

Head of Department  
Department of Cybernetics  
School of Computer Science



**HOD**  
**(Dr. Monit Kapoor)**