

# NETWORK SIMULATION TOOLKIT

Enrollment Number – 101270

Name of Student – Aman Batra

Name of Supervisor – Mr. Ravindara Bhatt



MAY-2014

Submitted in partial fulfillment of the Degree of  
Bachelor of Technology

DEPARTMENT OF COMPUTER SCIENCE  
JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY  
WAKNAGHAT

## **TABLE OF CONTENTS**

<b>Chapter No.</b>	<b>Topics</b>	<b>Page No.</b>
	Certificate	ii
	Acknowledgement	iii
	Summary	iv
Chapter-1	Introduction	5-6
	1. Introduction to Network Simulator	
	2. Uses and Functions of Simulator	
	3. System Specifications & Requirements	
Chapter-2	Background Theory	7-26
	1. Networking Devices	
	(a) Routers	
	(b) Switches	
	2. Networking Protocols	
	2.1 Internet Control Message Protocol (ICMP)	
	2.2 User Datagram Protocol (UDP)	
	2.3 Internet Protocol version 4 (IPv4)	
	2.4 Address Resolution Protocol (ARP)	
	2.5 Telnet	
Chapter-3	Details of the Simulator	27-30
	1. Front-End Simulation	
	2. Back-End Simulation	
Chapter-4	Simulator Architecture and Implementation	31-49
	Conclusion	50
	References	51

## **CERTIFICATE**

This is to certify that the work titled “**NETWORK SIMULATION TOOLKIT**” submitted by “**Aman Batra**” in partial fulfillment for the award of degree of Bachelor of Technology in Computer Science of Jaypee University of Information Technology, Waknaghat has been carried out under my supervision. This work has not been submitted partially or wholly to any other University or Institute for the award of this or any other degree or diploma.

**Signature of Supervisor** .....

**Name of Supervisor**     Mr. Ravindara Bhatt

**Designation**                 Assistant Professor

**Date**                                 .....

## **ACKNOWLEDGEMENT**

I express my sincere gratitude to my respected project supervisor Sr. Lect. Mr. Ravindara Bhatt, Department of Computer Science And Engineering, Jaypee University of Information Technology, Wagnaghat under whose supervision and guidance this work has been carried out. His whole hearted involvement, advice, support and constant encouragement throughout, have been responsible for carrying out this project work with confidence. I am also grateful to him for providing me with required infrastructural facilities that have been highly beneficial to me in undertaking the above mentioned project.

I am sincerely grateful to Brig. S.P. Ghrera, Professor and Head of Department of Computer Science and Engineering, Jaypee University of Information Technology, Wagnaghat for providing all necessary facilities for the successful completion of my project.

I would also like to thank the laboratory staff of Department of Computer Science and Engineering for their timely help and assistance.

**Signature of Student** .....

**Name of Student** Aman Batra

**Date** .....

## SUMMARY

A **Network Simulator** is software tool that can analyze how a virtual computer network behaves and performs without the presence of a computer network in the real world. In a network simulator, a network topology can be modeled using devices like routers, switches, connecting cables etc. and then analysis of network performance takes place. Basically, a user can simulate any network topology which can be created using the topology editing tool.

In the topology editing tool, a network topology can be made using the basic elements of the network like routers (Linux and Cisco routers), switches (L2, L3 and L4 switches), PCs and Ethernet cables. Then, after we can save the topology as an XML file, we need to run the backend simulator with the same configuration file as opened in the frontend GUI simulator. The configuration file is same as the topology XML file.

After opening the configuration file in both i.e. the backend and the frontend, just go to the simulator , and make a connection to the server where the backend process is running (for the same system, the IP address to be provided would be 127.0.0.1) and mention the same port number on which the backend is being run.

After the connection to the server is made, just open the built-in telnet session or a separate putty window for all the PCs and routers. After configuring the network nodes, enabling IP forwarding, creating static routes on each router and a default route on each PC, start transmitting data packets from end to another or from one node to another.

Then the data packets can be captured over the network and can be seen visually and hence, simulation of network topology can be analyzed.

## **1. INTRODUCTION**

A **network simulator** is software tool that can analyze how a virtual computer network behaves and performs without the presence of a computer network in the real world. In a network simulator, a network topology can be modeled using devices like routers, switches, connecting cables etc. and then analysis of network performance takes place. Basically, a user can simulate or customize any network topology which can be created using the topology editing tool. A Simulator normally comes with support and documentation for the most popular networking protocols and networks in use today.

### **USES of Network Simulator:**

A Network Simulator can assist in the following way:

- Network Simulators are inexpensive as compared to the cost involved in real network setup and give efficient performance.
- It helps a person to test and analyze any particular case that may be tough or relatively expensive to emulate using the real hardware.
- It also helps a person to analyze new network protocols or changes made to an existing protocol in a network topology simulation.
- It also helps one to design complex and hierarchical networks using various network elements.
- One can also simulate network protocols like TCP, IMAP, POP3, IMGP, ARP etc. using a simulator and also can test and analyze various standard simulation results.

The Simulator which has been developed is a simple graphical type of network simulator. It assists one in creating a virtual network topology consisting of Linux and Cisco routers, switches and cables. A virtual network topology can be instantly created in the front-end simulation GUI. The network devices present in the virtual network topology can be configured through command window or Telnet client. The front-end simulation GUI is designed to view and capture data packets transmitted in the virtual network topology.

It is a multiplatform application written in Java (version 7).

### **System Specifications and Requirements:**

- Java Run Time Environment & Development Kit version 7
- **Telnet Client:** You can use the built-in client but if you don't like it you can download putty on Windows or just use telnet in case of Linux/Unix system.
- **Xterm Terminal:** Standard terminal emulator for X Window System. Eg. MobaXterm Personal Edition.

## **2. BACKGROUND THEORY**

There are a variety of networking concepts used in this project including several protocols, routing knowledge, networking devices and many more which are briefly explained below:

### **2.1 Networking Devices**

#### ***a) Routers***

A Router is a part of networking hardware which serves important functions in a network. It is a cross point node in a network which reads the destination address marked in an incoming packet, to consult its internal information to identify an outgoing link to which the packet is to be forwarded and then to forward the packet. A network link that connects two routers is limited by how much data it can transfer per unit of time commonly referred to as the bandwidth or capacity of a link. A network then carries traffic on its link and through its routers to the eventual destination.



A router also decides which data messages must be processed first when there will be network traffic and routers will be filled with multiple queues. Routers also maintain a routing table in



case of any error or data packet loss. This routing table also help routers to make decision for processing packets in case of network traffic.

There is a special type of routers named Cisco routers used in this project. These routers help create a more intelligent, responsive, and integrated network, based on adaptive and agile technologies.

### b) *Switches*

A switch is a device that gathers the signals from devices that are connected to it, and then regenerates a new copy of each signal. It connects individual devices on an Ethernet network so that they can communicate with one another. It momentarily connects the sending and receiving devices so that they can use the entire bandwidth of the network without interference.

Switches have **two** benefits:

- They provide each pair of communicating devices with a fast connection.
- They segregate the communication so that it does not enter other portions of the network.



These benefits are particularly useful if your network is congested and traffic pools in particular areas.

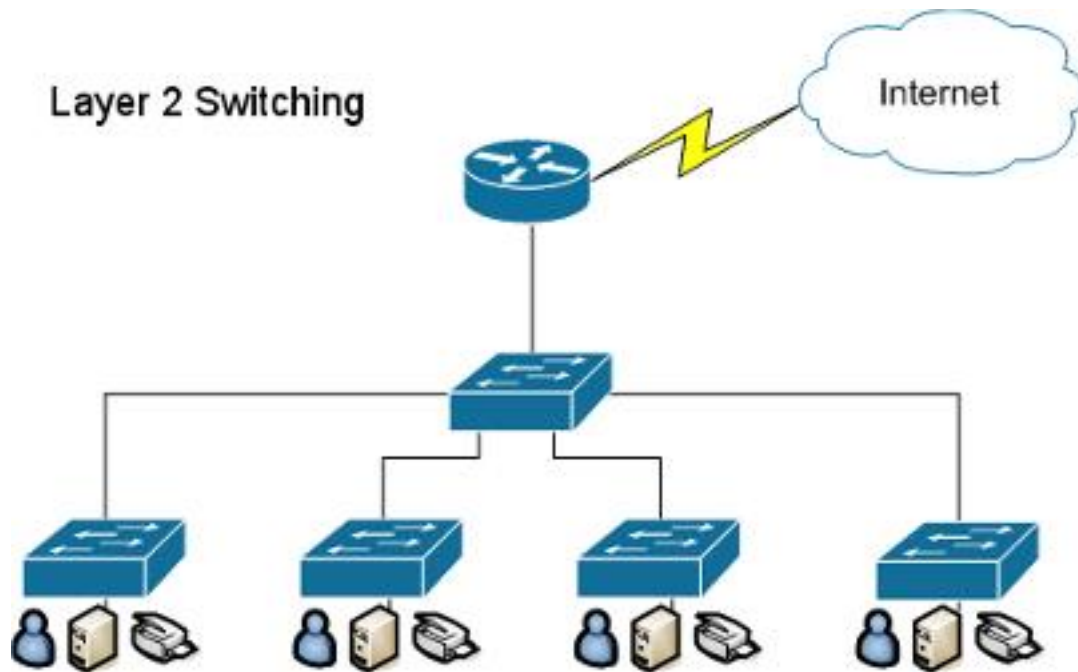
A switch learns the MAC-addresses of the computers NIC's connected to his ports, writing them down in a MAC-address table. When a switch receives a data packet on one port, it reads out the destination MAC-address from the packet header and then forwards the packet to the port on which the NIC having this address is connected to. Thus reducing the network traffic a very good amount.

The only traffic transported out of every port of the switch, apart from the port the switch is receiving this packet, is the broadcast traffic the computers connected to the switch are causing. This 'one-to-one' communication still does not need IP addresses; it is based on MAC-addresses only. Therefore, a switch is a OSI-layer 2 device too. It cannot be responsible for rejecting any network traffic and therefore cannot be a part of the trouble shooting for users not are able to DCC or trying to use any other networking services.

### ***(i) Layer 2 Switching***

A switch connects segments of a network and makes filtering decisions about which data is allowed to travel between the segments and which data stays on a single segment. But while a bridge may connect only two segments, a switch can connect many segments. One switch can do the work of many bridges.

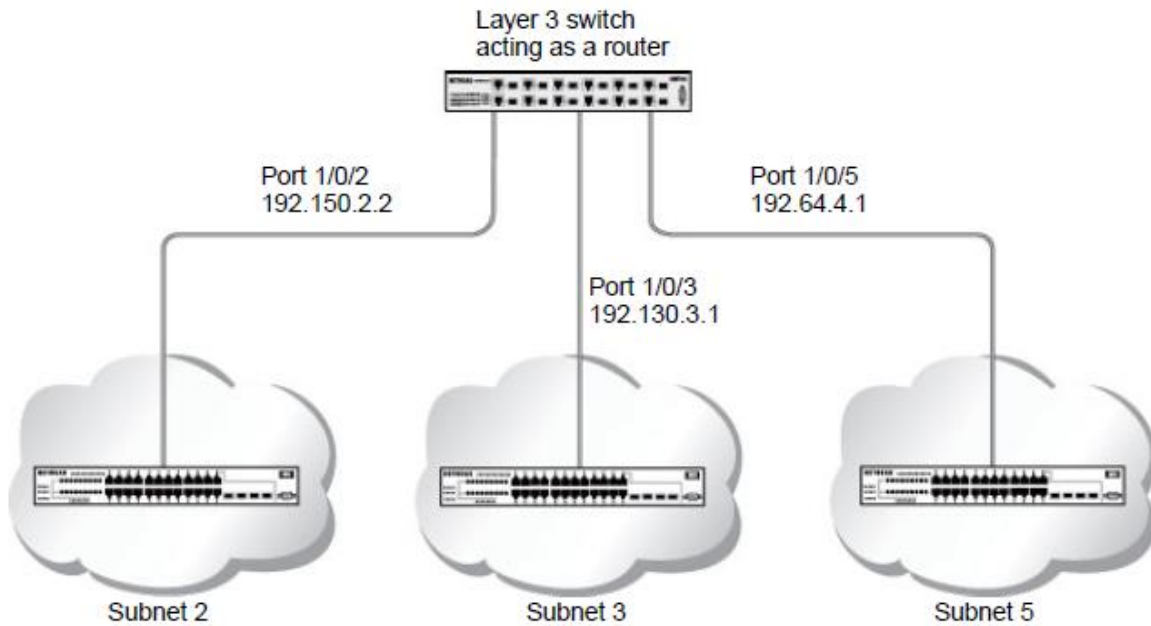
Layer 2 switches operate at the data link of the OSI model. Layer 2 is responsible for error checking and transmitting data across the physical media. MAC addressing source and destination protocols are Layer 2 protocols.



Layer 2 switches use the MAC address of data packets to determine where those packets should go. When you connect a switch to a network, it immediately listens for all the devices on all the segments that are connected to the switch's ports. It learns the MAC addresses of all of those devices and creates a segment or forwarding table.

### *(ii) Layer 3 Switching*

It is also known as packet switching (routing). Switches packets based on IP addressed instead of Ethernet MAC addresses. Older switches enabled IP level routing in software by forwarding a packet to the processor which in turn performed the required routing operations. Currently, IP routing is performed in hardware, at interface line rate, similarly to Layer-2 (Ethernet) switching. The processor is only used to process specific routing protocols and configure IP routing tables in the hardware.



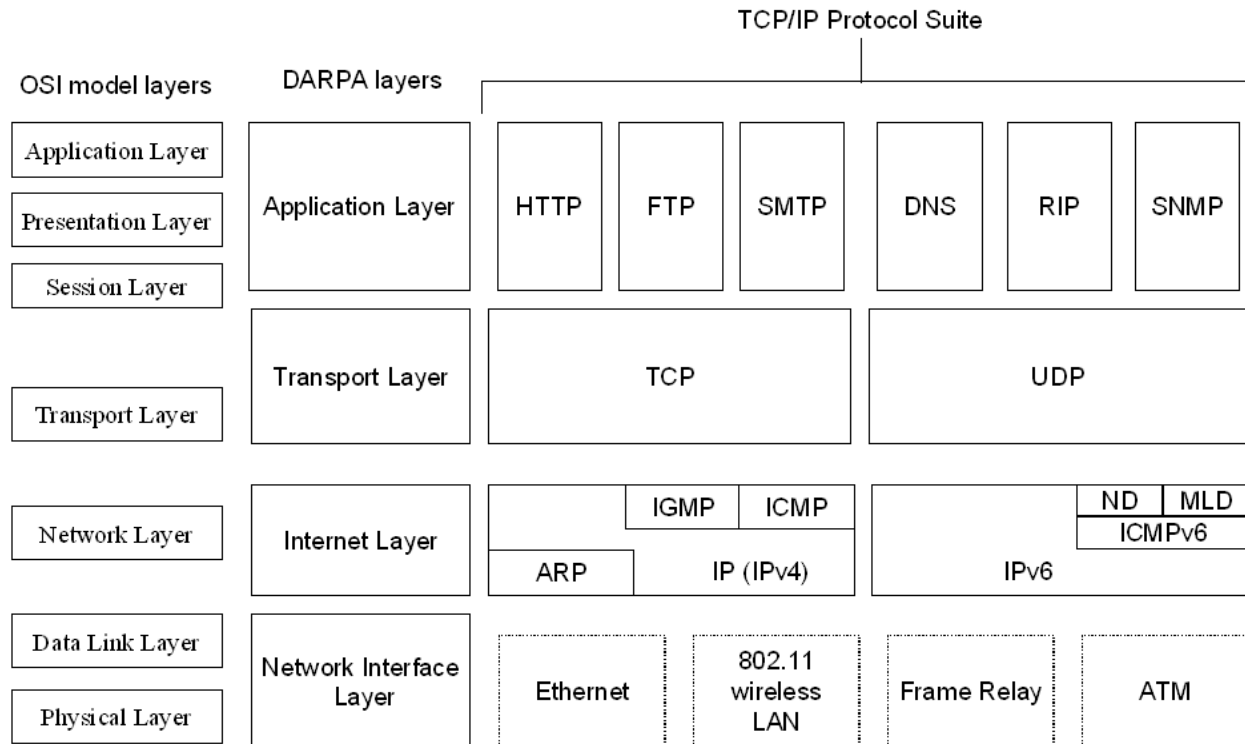
## 2.2 Networking Protocols

### 2.2.1 Internet Control Message Protocol (ICMP)

The Internet Control Message Protocol (ICMP) is a mechanism used by hosts and gateways to send notification of datagram problems back to the sender. ICMP sends query and error reporting messages.

Since the IP provides unreliable and connectionless datagram delivery, it was designed this way to make efficient use of network resources. The IP protocol is a best-effort delivery service that delivers a datagram from its original source to its final destination. However, it has two deficiencies: **lack of error control** and **lack of assistance mechanisms**.

The Internet Control Message Protocol (ICMP) has been designed to compensate for the above two deficiencies. It is a companion to the IP protocol.



**Position of Networking Protocols with accordance of each OSI model layer**

## Types of Messages

ICMP messages are divided into two broad categories: **error-reporting messages** and **query messages**.

**The error-reporting messages** report problems that a router or a host (destination) may encounter when it processes an IP packet.

**The query messages**, which occur in pairs, help a host or a network manager get specific information from a router or another host. For example, nodes can discover their neighbors. Also, hosts can discover and learn about routers on their network, and routers can help a node redirect its messages.

## ICMP Message Types

Type	Description and ICMP Message Types
0	Echo Reply (Ping Reply, used with Type 8, Ping Request)
3	Destination Unreachable
4	Source Quench
5	Redirect
8	Echo Request (Ping Request, used with Type 0, Ping Reply)
9	Router Advertisement (Used with Type 9)
10	Router Solicitation (Used with Type 10)
11	Time Exceeded
12	Parameter Problem
13	Timestamp Request (Used with Type 14)
14	Timestamp Reply (Used with Type 13)

15	Information Request (obsolete) (Used with Type 16)
16	Information Reply (obsolete) (Used with Type 15)
17	Address Mask Request (Used with Type 17)
18	Address Mask Reply (Used with Type 18)

This table is adapted from ‘**Data Communication & Networking by Forouzan**’

### **Ping: Echo Request and Reply—Types 8 and 0**

We can use **ping** here to see how it uses ICMP packets. The source host sends ICMP echo-request messages (type: 8, code: 0); the destination, if alive, responds with ICMP echo-reply messages. The **ping** program sets the identifier field in the echo-request and echo-reply message and sets the sequence number from 0; this number is incremented by 1 each time a new message is sent. Note that *ping* can calculate the round-trip time. It inserts the sending time in the data section of the message. When the packet arrives, it subtracts the arrival time from the departure time to get the round-trip time (RTT).

### **Destination Unreachable—Type 3**

When a router cannot route a datagram or a host cannot deliver a datagram, the datagram is discarded and the router or the host sends a destination-unreachable message back to the source host that initiated the datagram. Note that destination-unreachable messages can be created by either a router or the destination host.

### **0 = Network Unreachable**

This message indicates that the router cannot find the destination network (does not exist or has failed) or has no route to this network. In other words, the router cannot deliver or forward an IP datagram to the destination network. This could be the result of a network that is beyond the maximum distance limitation for the routing protocol in use and is therefore considered unreachable (too far). When a client attempts to connect to a host on a network that is unreachable, a gateway generates this message to alert the source host of the problem.

### **1 = Host Unreachable**

This error comes when the destination host should be directly reachable, but does not respond to ARP Requests.

### **2 = Protocol Unreachable**

This error comes when the protocol in the protocol field of the IP header is not supported at the destination.

### **3 = Port Unreachable**

This error comes when the transport protocol at the destination host cannot pass the datagram to an application.

### **4 = Fragmentation is needed, but don't-fragment bit set**

This error message occurs when the IP datagram must be fragmented, but the DF bit in the IP header is set.



### **5 = Source Route Failed**

This error message occurs when the router finds that the next node in the source route that does not exist on the network.

### **6 = Destination Network Unknown**

This error message occurs when the router is unable to deliver the datagram because destination address is unknown.

### **7 = Destination Host Unknown**

This error message occurs when the router is unable to deliver the datagram because destination node is unknown.

### **8 = Source Host Isolated (obsolete)**

### **9 = Destination Network Administratively Prohibited**

This error message occurs when there is prohibition by the administrator for delivering the datagram to the destination address via router.

### **10 = Destination Host Administratively Prohibited**

This error message occurs when there is prohibition by the administrator for delivering the datagram to the destination node via router.

### **Source Quench—Type 4**

There is no communication between the source host, which produces the datagram, the routers, which forward it, and the destination host, which processes it. One of the ramifications of this absence of communication is the lack of flow control. The source-quench message in ICMP was designed to add a kind of flow control to the IP. When a router or host discards a datagram due

to congestion, it sends a source-quench message to the sender of the datagram. This message has two purposes. First, it informs the source that the datagram has been discarded. Second, it warns the source that there is congestion somewhere in the path and that the source should slow down the sending process.

### **Redirect—Type 5**

When a router needs to send a packet destined for another network, it must know the IP address of the next appropriate router. The same is true if the sender is a host. Both routers and hosts, then, must have a routing table to find the address of the router or the next router. The hosts usually use static routing. When a host comes up, its routing table has a limited number of entries. It usually knows the IP address of only one router, the default router. For this reason, the host may send a datagram, which is destined for another network, to the wrong router. In this case, the router that receives the datagram will forward the datagram to the correct router. However, to update the routing table of the host, it sends a redirection message to the host.

### **Time Exceeded—Type 11**

The time-exceeded message is generated in two cases: As we know, routers use routing tables to find the next hop that must receive the packet. If there are errors in one or more routing tables, a packet can travel in a loop or a cycle, going from one router to the next or visiting a series of routers endlessly. As each datagram contains a field called **time to live** that controls this situation.

When a datagram visits a router, the value of this field is decremented by 1. When the time-to-live value reaches 0, after decrementing, the router discards the datagram. However, when the datagram is discarded, a time-exceeded message must be sent by the router to the original source. Second, a time-exceeded message is also generated when not all fragments that make up a message arrive at the destination host within a certain time limit.

### **Parameter Problem—Type 12**

Any ambiguity in the header part of a datagram can create serious problems as the datagram travels through the Internet. If a router or the destination host discovers an ambiguous or missing value in any field of the datagram, it discards the datagram and sends a parameter problem message back to the source.

### **Router Advertisement and Solicitation—Types 9 and 10**

A host can broadcast (or multicast) a router-solicitation message. The router or routers that receive the solicitation message broadcast their routing information using the router-advertisement message. A router can also periodically send router-advertisement messages even if no host has solicited.

Note that when a router sends out an advertisement, it announces not only its own Presence but also the presence of all routers on the network of which it is aware.

### **Timestamp Request and Reply—Types 13 and 14**

Two machines (hosts or routers) can use the timestamp request and timestamp reply messages to determine the round-trip time needed for an IP datagram to travel between them. It can also be used to synchronize the clocks in two machines.

### **Information Request and Reply—Types 15 and 16**

A host node can make a request for information like to what type of network it is connected.

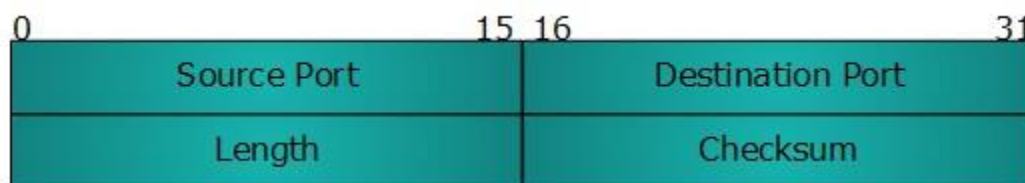
## Address Mask Request and Reply—Types 17 and 18

A host may know its IP address, but it may not know the corresponding mask. To obtain its mask, a host sends an address-mask-request message to a router on the LAN. If the host knows the address of the router, it sends the request directly to the router. If it does not know, it broadcasts the message. The router receiving the address-mask-request message responds with an address-mask-reply message, providing the necessary mask for the host. This can be applied to its full IP address to get its subnet address.

### 2.2.2 User Datagram Protocol (UDP)

The User Datagram Protocol (UDP) is called a connectionless, unreliable transport protocol. It does not add anything to the services of IP except to provide process-to process communication instead of host-to-host communication. Also, it performs very limited error checking. UDP is a very simple protocol using a minimum of overhead. If a process wants to send a small message and does not care much about reliability, it can use UDP. Sending a small message by using UDP takes much less interaction between the sender and receiver than using TCP.

#### UDP Header:



A UDP header contains information about the following fields:

- **Source Port Number:** This 16-bit port number used by the process running on the source host.
- **Destination Port:** This 16-bit port number used by the process running on the destination host.

- **Length:** This 16-bit field defines the total length of the user datagram i.e. the header plus the data.
- **Checksum:** This 16-bit field is used to detect errors over the entire user datagram i.e. the header plus the data.

### 2.2.3 Internet Protocol Version 4 (IPv4)

An Internet Protocol version 4 address is a 32-bit address that uniquely and universally defines the connection of a device (for example, a computer or a router) to the Internet. It is a widely used Internet protocol available on the Internet. It is a connectionless protocol. IP Version 4 was the first version to experience widespread deployment.

IP Version 4 addresses are unique. They are unique in the sense that each address defines one and only one connection to the Internet. Two devices on the Internet can never have the same address at the same time.

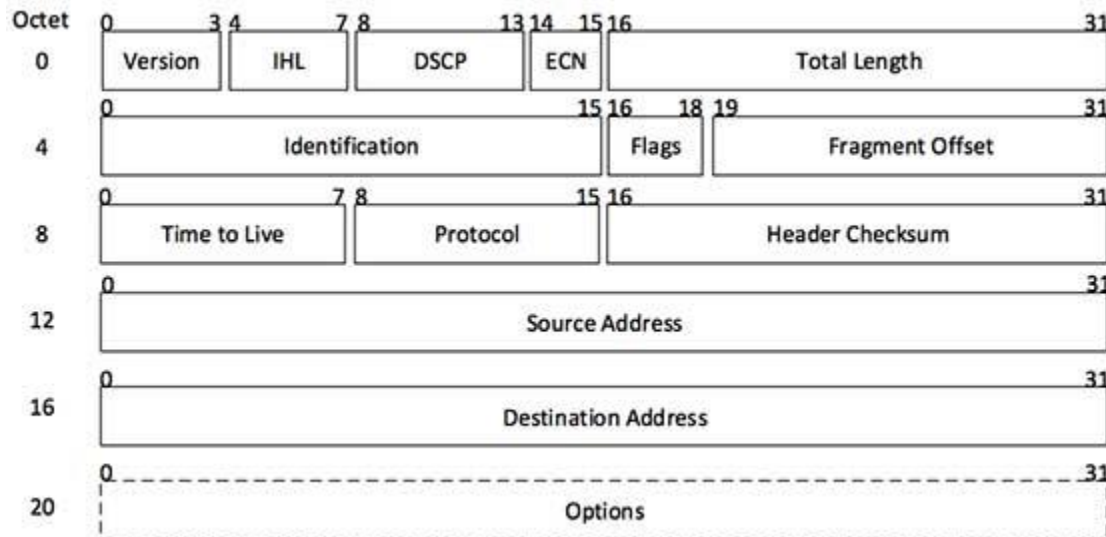
IP Version 4 uses 32-bit addresses, which means that the address space is  $2^{32}$  or 4,294,967,296 (more than 4 billion). This means that, theoretically, if there were no restrictions, more than 4 billion devices could be connected to the Internet.

This Internet Protocol takes data segments from the Transport layer and divides it into small chunks known as packets. An IP packet contains data unit which has been received from the above layer and it adds its own header information to it.



(IP Encapsulation)

This IP Packet data is defined to be an IP Payload. The IP header contains all the required information to deliver the packet from the source to the destination.



The IP header also includes some important information which includes the Version Number which is 4 in this protocol. The other details are also as follows:

- **Version:** Version Number of Internet Protocol.
- **Header Length:** Tells the end of header and beginning of payload
- **Differentiated Services Code Point:** This is type of service to be requested.
- **Explicit Congestion Notification:** It contains the details about seen congestion in data packet transmission.
- **Total Length:** It tells about the end of an entire IP Packet unambiguously.
- **Identification:** All the fragments of a single datagram have the same identification number.
- **Flags:** 1<sup>st</sup> bit is reserved and must be zero 2<sup>nd</sup> bit tells if data packet does not requires segmentation and last i.e. 3<sup>rd</sup> bit tells about more fragments.
- **Fragment Offset:** It is used for reassembly of data packet after its fragmentation.

- **Time to Live:** It is initially set by sender to 255 which is decremented by each router and data packet is discarded if TTL is zero to avoid infinite loops.
- **Protocol:** Its value indicates what is in the data field. E.g. TCP or UDP
- **Header Checksum:** It checks for error in the header only. If error is found, the packet is discarded so that it does not harm the network.
- **Source Address:** It is a 32-bit IP address of the sender.
- **Destination Address:** It is a 32-bit IP address of the receiver.
- **Options:** It is an optional field. E.g. Timestamp, Source route, Record route.

Since no encryption or authentication is provided by IPv4 and maximum delay time for audio and video transmission provided in IPv4 design, IPv6 protocol was proposed. In IPv6, the Internet protocol was extensively modified to accommodate the unforeseen growth of the Internet. It has larger address space of 128 bit as compared to 32 bit address of IPv4 protocol.

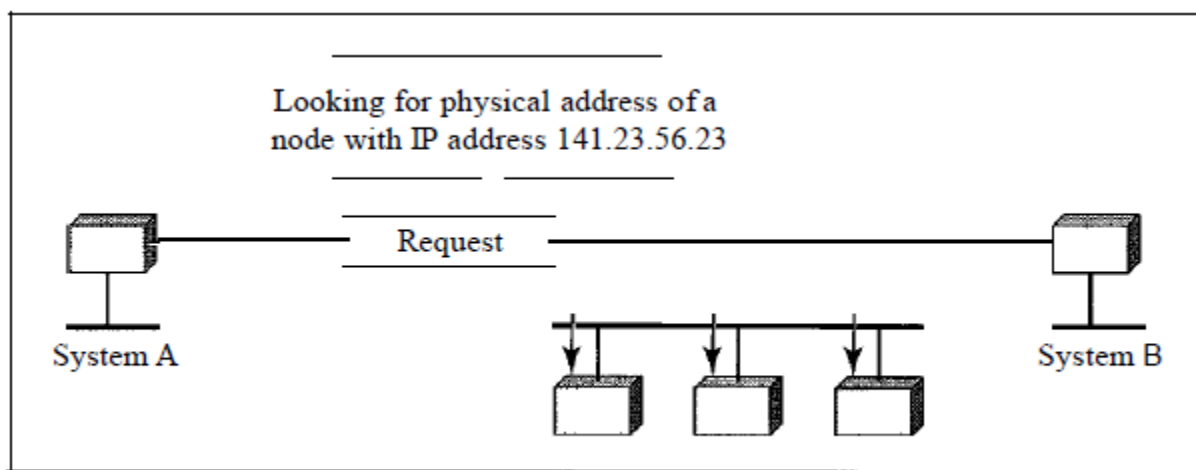
#### 2.2.4 Address Resolution Protocol (ARP)

The Address Resolution Protocol (ARP) is used to associate a logical address with a physical address. On a typical physical network, such as a LAN, each device on a link is identified by a physical or station address, usually imprinted on the network interface card (NIC). ARP is used to find the physical address of the node when its Internet address is known.

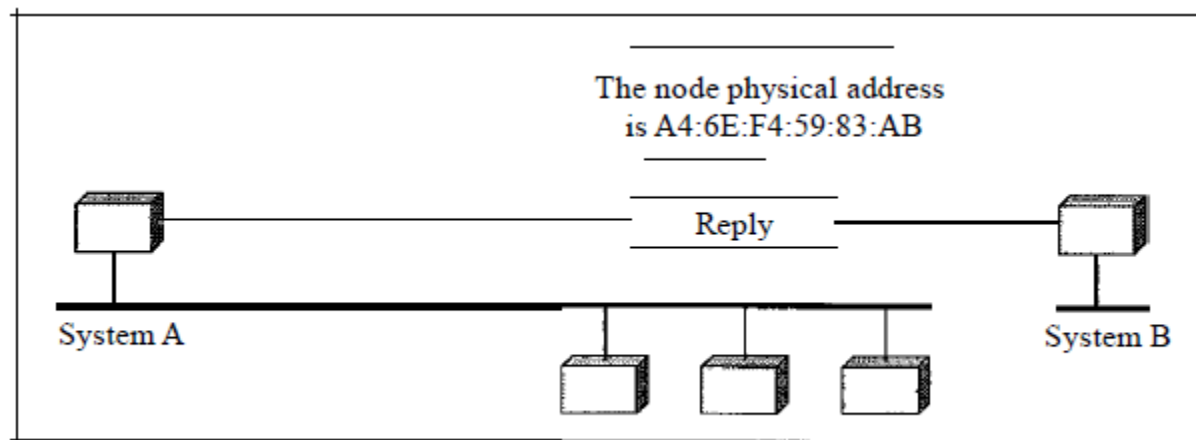
Anytime a host or a router has an IP datagram to send to another host or router, it has the logical address of the receiver. The logical (IP) address is obtained from the DNS if the sender is the host or it is found in a routing table if the sender is a router. But the IP datagram must be encapsulated in a frame to be able to pass through the physical network. This means that the sender needs the physical address of the receiver. The host or the router sends an ARP query packet. The packet includes the physical and IP addresses of the sender and the IP address of the receiver. Because the sender does not know the physical address of the receiver, the query is broadcast over the network.

Every host or router on the network receives and processes the ARP query packet, But only the intended recipient recognizes its IP address and sends back an ARP response packet. The response packet contains the recipient's IP and physical addresses. The packet is unicast directly to the inquirer by using the physical address received in the query packet.

#### ARP Request & Reply Operation:



a. ARP request is broadcast



b. ARP reply is unicast



Internet Protocol (IPv4) over Ethernet ARP packet		
octet offset	0	1
0	Hardware type (HTYPE)	
2	Protocol type (PTYPE)	
4	Hardware address length (HLEN)	Protocol address length (PLEN)
6	Operation (OPER)	
8	Sender hardware address (SHA) (first 2 bytes)	
10	(next 2 bytes)	
12	(last 2 bytes)	
14	Sender protocol address (SPA) (first 2 bytes)	
16	(last 2 bytes)	
18	Target hardware address (THA) (first 2 bytes)	
20	(next 2 bytes)	

<b>22</b>	(last 2 bytes)
<b>24</b>	Target protocol address (TPA) (first 2 bytes)
<b>26</b>	(last 2 bytes)

This table is adapted from ‘**Data Communication & Networking by Forouzan**’

### **Hardware type (HTYPE)**

This 16-bit field defines type of the network on which ARP is running.

### **Protocol type (PTYPE)**

This 16-bit field defines the protocol. ARP can be used with higher level protocol.

### **Hardware length (HLEN)**

This 8-bit field defines the length of the physical address in bytes.

### **Protocol length (PLEN)**

This 8-bit field defines the length of the logical address in bytes.

### **Operation**

This 16-bit field defines the type of packet. It’s either of ARP request or ARP reply.

### **Sender Hardware Address (SHA)**

This variable length field defines the physical address of the sender.

### **Sender Protocol Address (SPA)**

This variable length field defines the logical address of the sender.

**Target Hardware Address (THA)**

This variable length field defines the physical address of the target.

**Target protocol address (TPA)**

This variable length field defines the logical address of the target.

**2.2.5 TELNET**

Telnet is a network protocol which is used in internet or local area networks to provide a client-server interactive communications facility. It is a standard method to communicate with another network host. Telnet provides a standard interface for terminal devices and terminal-oriented processes through a network.

Using the Telnet protocol, a user can login remotely on a local host and execute commands on another distant host computer. It employs a client-server model. Even Today Telnet provides a text-based user interface. Telnet was one of the first Internet applications since the earliest demand was to connect two or more terminals to hosts across networks.

Telnet is one of the most popular Internet applications because of its flexibility like checking E-Mails etc; it does not waste much network resources and Telnet clients are integrated in every UNIX environment and other operating systems.

Telnet is a connection oriented protocol as it uses Transmission Control Protocol (TCP).

Clients connect to the well known destination port 23 on the server side.

### **3. DETAILS OF THE SIMULATOR**

This JAVA simulator built is classified into two parts:

1. **Frontend Simulation** which is further divided into two parts namely:

- Topology Editing Tool
- Network Simulator

2. **Backend Simulation**

It is used for sending and capturing the packets via implementing network routing protocols.

Frontend Simulation is the part whose functioning is visible to the user. In this, the user can edit the topology using the editing tool, where as the simulator works on interaction with the backend simulation process.

In the backend process, the packets are captured and recorded over the network through the event handler.

Topology editing tool is completely independent of anything; it does not need any type of support to run but the simulator needs the backend process to run simultaneously over the same port.

### 3.1 Network Simulator System Requirements

#### *a) Hardware Requirements*

**Processor:** Minimum 1 GHz or faster

**RAM:** 1 GB (32-bit) or 2 GB (64-bit)

**Hard disk space:** At least 10 GB of disk space

#### *b) Software Requirements*

**Operating System:** Windows XP Professional/Vista/7/8 or Linux

**Software and Tools:** Java Development Kit 7, Java Runtime Environment 7, Telnet, Eclipse or Netbeans and MobaXterm Terminal Personal Edition.

### 3.2 Topology Editing Tool

Topology editing tool is a JAVA GUI based editor which is used for the setting of visual virtual connections between the different types of routers, switches and PCs via Ethernet connecting cables. In the editing tool, the connections can be setup by dragging the devices to the editing area. Over there, the user can specify the MAC Address and IP Address of each of the devices. Also the visibility of the MAC Address, IP Address, adding of interfaces, connection status etc can be setup in the properties tab of the device.

Using the side panel that comprises of the devices, moving of devices, alignment along the grid, resizing them can be done using the topology editor's editing section.

Also devices once added can be deleted or renamed according to the user's requirement.

### 3.3 Frontend Simulation

The frontend is run in the DOS window using the following command: **java -jar simulator\_frontend.jar** .Once the backend jar file is running, the user can connect to the server by using the "Connect to Server" and specify the IP address and the port number of the same location over which the backend file is running.

Then, the user can open a putty session or a built-in telnet session over the PCs and routers, and then configure the nodes using a 3-step process:

1. enable IP forwarding on all the routers
2. create static routes on each router
3. create a default route on each PC

Click on the capture button and then start ping or sending messages from one end device to another, and then the user can view the packet status in the event list. In the event list, the user can select the packet and can view the packet details in the below displayed box.

Also, the user can view the previously played event by using the PLAY CONTROLS mentioned in the Simulator GUI.

Also, events once taken place can be deleted from the list, for example the packet capture from one system device to another can be deleted.

Also, events in the GUI can be saved and could be loaded later for transmission of packets from one end to another.

### 3.4 Backend Simulation

The backend is run in the DOS window using the following command: **java -jar simulator\_frontend.jar topology-file.xml**, and the backend process is started for the defined topology file mentioned in the above command.

The backend is used for capturing the packets and is used to send the information to the frontend simulator. As the packets are captured in the DOS window over the network, the statistics are shown in the DOS window as well.

#### **4. SIMULATOR ARCHITECTURE**

This JAVA Network Simulator is categorized in two parts – the front-end simulation and the back-end simulation. The front-end simulation is a graphical user interface based on JAVA that helps a user to create any network topology using the topology editing tool and after the network simulation is started, it displays the packets which are successfully transmitted and which are failed in network using some animations. The back-end simulation runs the simulation based on the information retrieved from the XML file generated by the front-end simulation.

This JAVA Simulator can work in a case where the front-end simulation can run on one computer and the back-end simulation runs on another computer. But till date, both the front-end and back-end are run on the same computer.

##### **Steps to create a new network simulation in this JAVA Simulator**

To create and run a network simulation in the JAVA Simulator developed, follow these steps:

1. Start the front-end simulation.
2. Use the GUI front-end to create a new network topology using topology editing tool.
3. Save that network topology using any name in the form of an XML file.
4. Start the back-end simulation which will use the previously-saved XML file to create the simulation nodes.
5. Connect the front-end network topology to the back-end running simulation so that we can use the front-end GUI to demonstrate functioning of the network in the simulation.

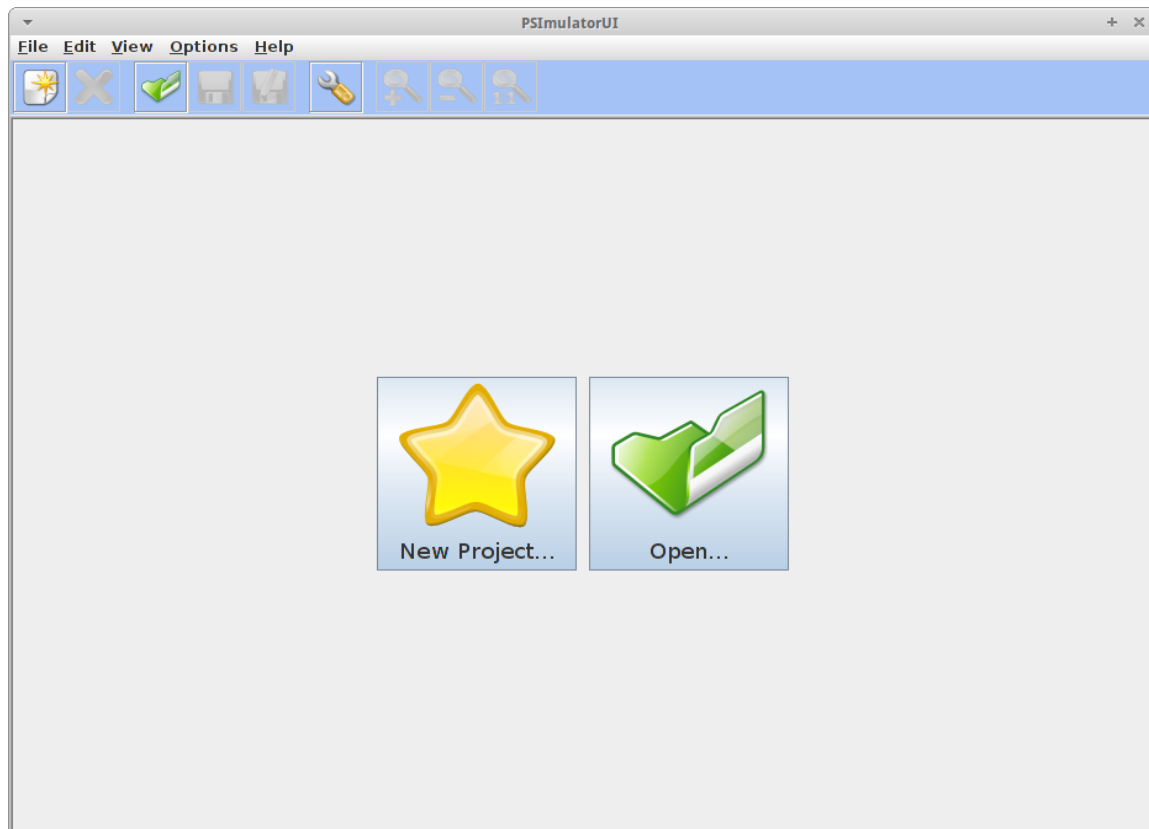


## 4.1 Start Front-End JAVA GUI Network Simulation

To start the front-end network simulation, execute the following command:

```
$java -jar simulator_frontend.jar
```

This will start the front-end GUI network simulator. Using this, we can create a new network topology or open any existing network topology XML file.

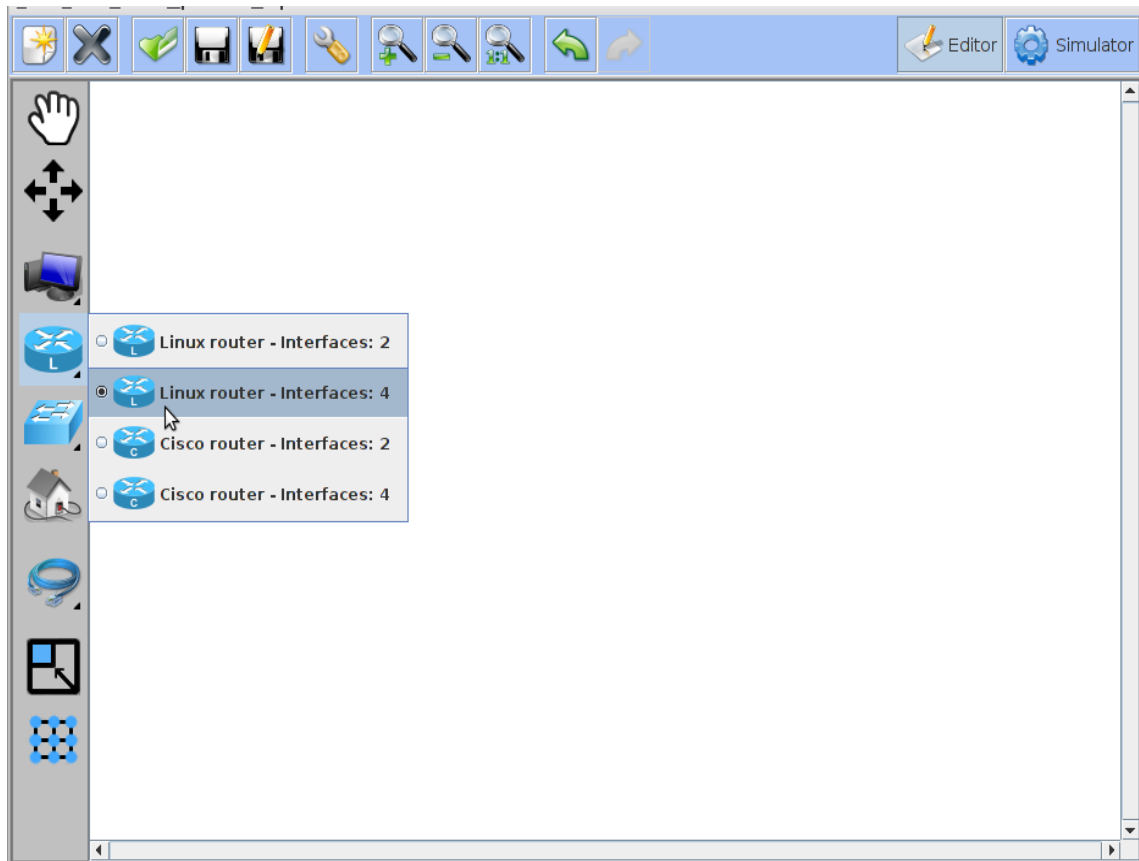


## 4.2 Create a new Network topology

To create a new network topology using the topology editing tool, click on the **New Project** button. We will see a blank editing screen window with some tools like to add network elements such as host PCs, routers, switches and network connections. It also offers some tools to organize elements on the editing window.

## Steps to add Network Nodes on the Editing Screen

To add elements of a network, right click on any element icon e.g. a router and then click on one of its types that appear on the sub-menu.

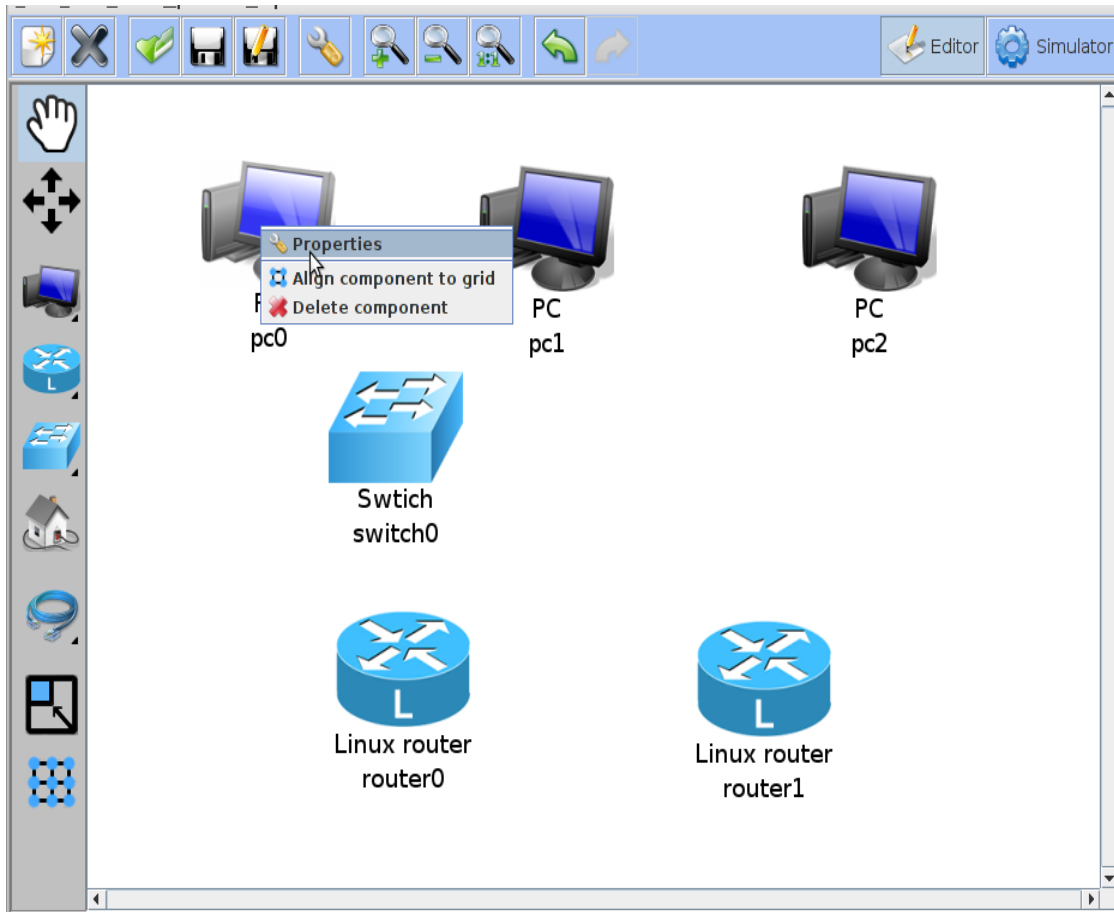


Then, click on the location in the editing screen where we want that node to appear. We may click again to add more routers of same or another type.

We can select network parts already on the editing screen window and position them around the screen.

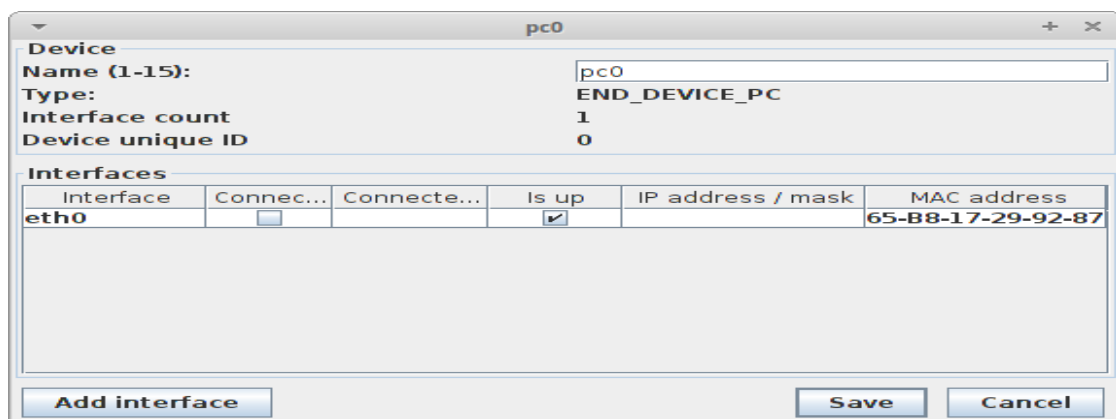
## Configuring the interface of the network elements

To configure the network element, click on the editing screen and select **Properties** from the sub-menu.

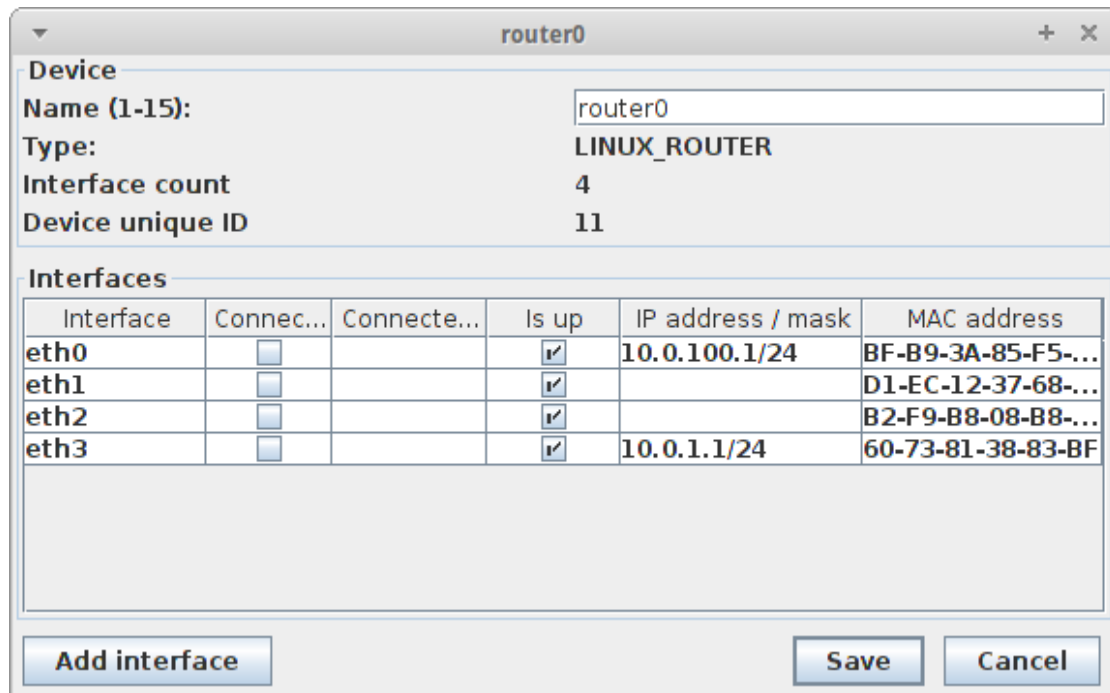


We can rename the node or configure the IP addresses of each network element. We can also add more network elements if we need to use them in the network topology. We can configure those network elements at any time.

Down below, we can see the configuration of a host PC network element:



Down below, we can see the configuration of a router. As we can see, this router has four networking interfaces but we are using only two of those. We are using the lowest number interfaces to connect to the local networks and the highest number interfaces to connect to the other routers.



The screenshot shows a configuration window titled 'router0'. It contains a 'Device' section with the following fields:

- Name (1-15): router0
- Type: LINUX\_ROUTER
- Interface count: 4
- Device unique ID: 11

Below the device section is an 'Interfaces' section containing a table with the following data:

Interface	Connec...	Connecte...	Is up	IP address / mask	MAC address
eth0	<input type="checkbox"/>		<input checked="" type="checkbox"/>	10.0.100.1/24	BF-B9-3A-85-F5-...
eth1	<input type="checkbox"/>		<input checked="" type="checkbox"/>		D1-EC-12-37-68-...
eth2	<input type="checkbox"/>		<input checked="" type="checkbox"/>		B2-F9-B8-08-B8-...
eth3	<input type="checkbox"/>		<input checked="" type="checkbox"/>	10.0.1.1/24	60-73-81-38-83-BF

At the bottom of the window are three buttons: 'Add interface', 'Save', and 'Cancel'.

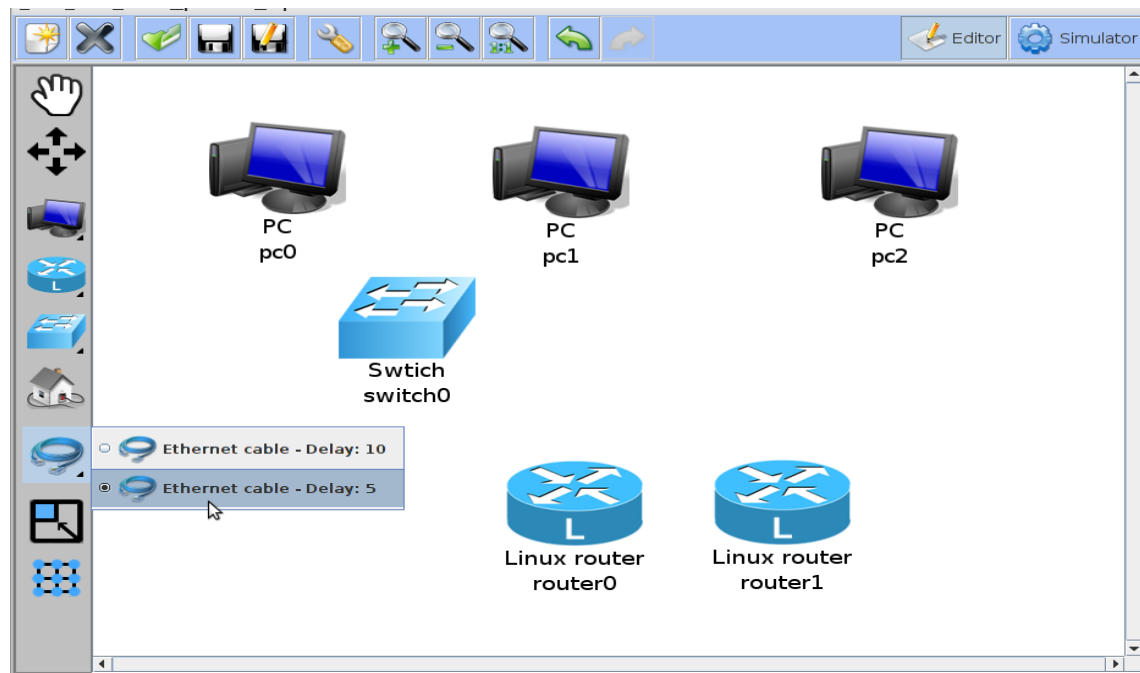
### Interfacing the IP addresses

In this example, the IP addresses for each node in the network should be configured as follows:

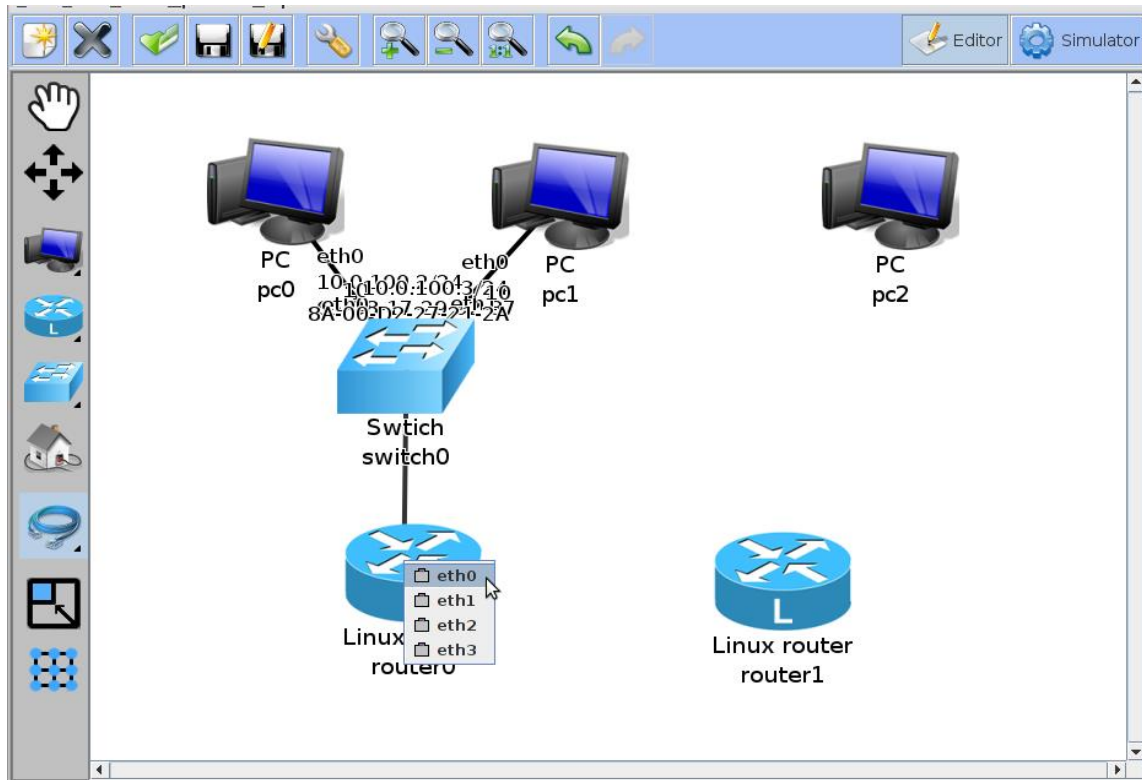
Node	Interface	IP Address
PC 0	eth0	10.0.100.2/24
PC 1	eth0	10.0.100.3/24
Router 0	eth0	10.0.100.1/24
	eth3	10.0.1.1/24
Router 1	eth0	10.0.200.1/24
	eth3	10.0.1.2/24
PC 2	eth0	10.0.200.2/24

## Connecting the Network Elements

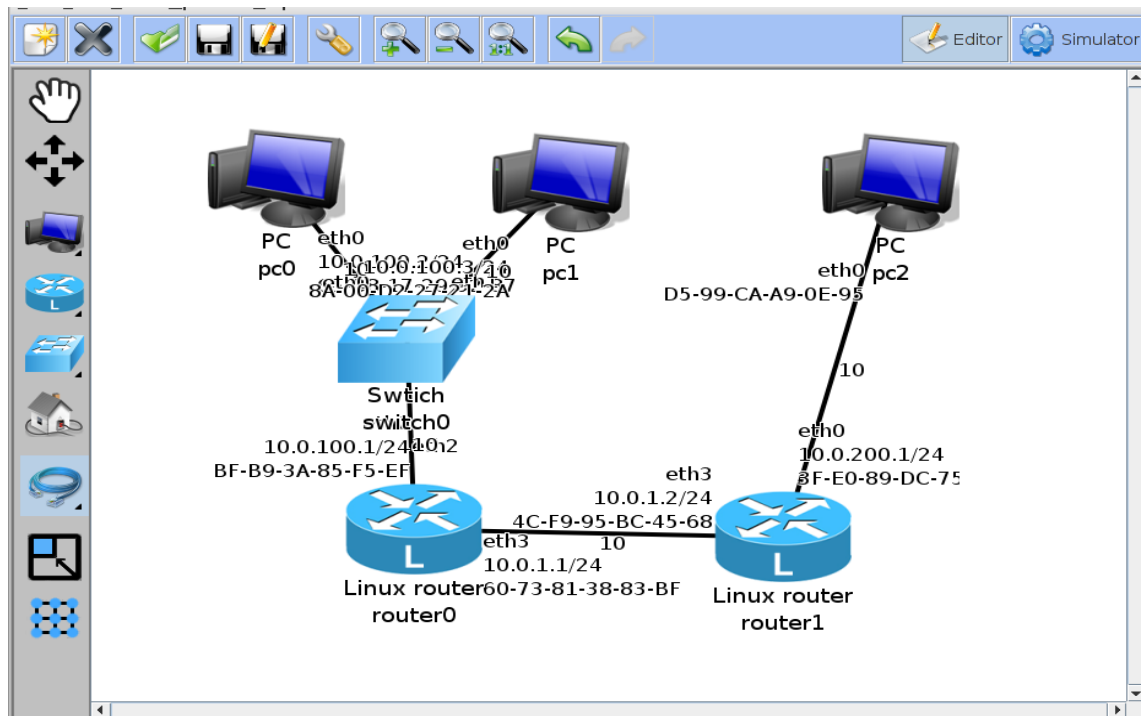
To establish a network connection between two nodes, click on the **Link** tool on the toolbar and then click on both the nodes that will form an endpoint of that connection.



We also need to mention which network interface to use on node devices with multiple interfaces like routers. To do this, right click on that node and then click the interface option on the sub-menu.



## Properties of Editing Screen Window



As we establish connection between nodes, more details about the network interfaces is displayed on the editing screen. Finally, the screen gets overfilled with the details.

We can also make changes with the JAVA simulator preferences so that lesser details are displayed. We can do this by clicking on the **Preferences** menu item. In that case, we can chose to show only the name of interfaces and IP addresses on each network interface.

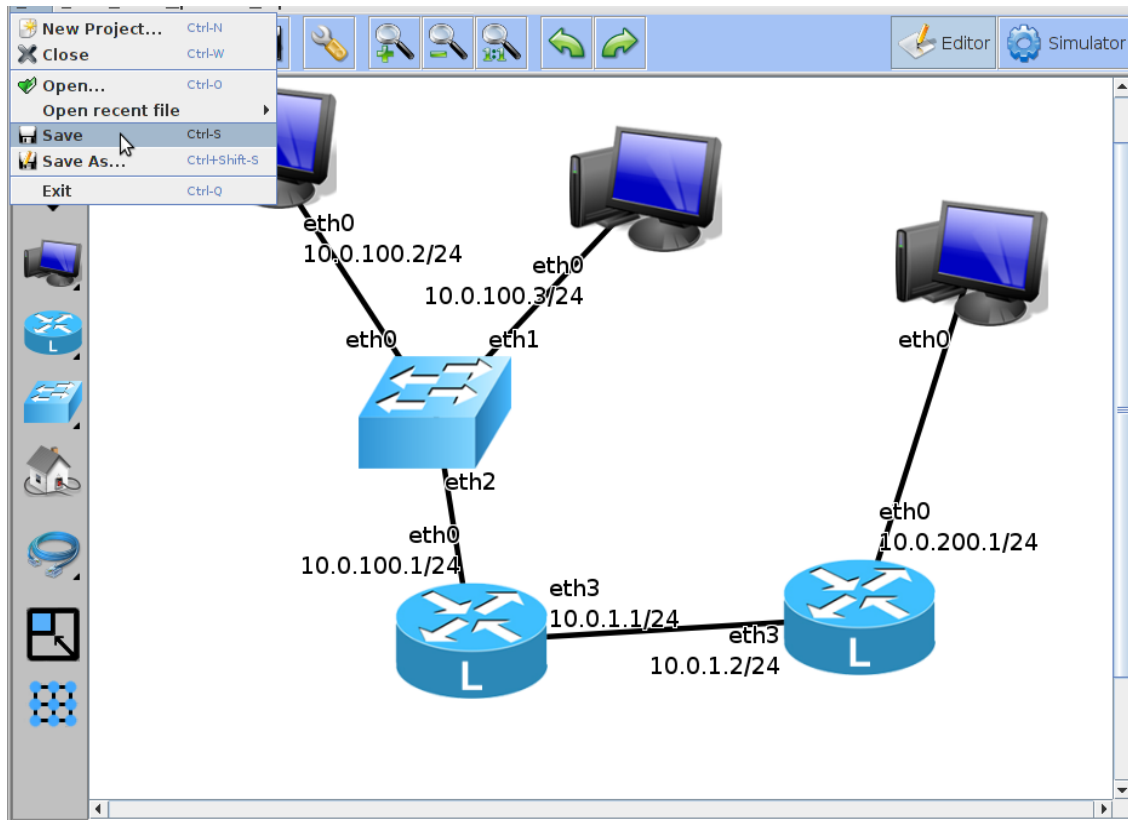


Now the editing screen is convenient to read.



## Steps to save the network topology XML file

To save the network topology created via topology editing tool, Click on the menu **File** then **Save**. In this case, we saved the network topology as **test-topology.xml**.



## 4.3 Starting the back-end Simulator

To start the back-end network simulator server in order to analyze the network topology we developed previously and saved as **test-topology.xml**. Enter the command:

```
$java -jar simulator_backend.jar test-topology.xml
```

The back-end network simulation server tells the port numbers of each network node. We will use this port number later on to connect to that network node via telnet. Remember the port

number which the back-end server is listening to. Also it must be the same for the front-end simulator.

```
Starting simulator, build 2014-01-29

[IMPORTANT] TELNET: TELNET LISTENING PORT: : Device:
pc0      listening port: 11000 (pc0)

[IMPORTANT] TELNET: TELNET LISTENING PORT: : Device:
router1 listening port: 11001 (router1)

[IMPORTANT] TELNET: TELNET LISTENING PORT: : Device:
pc1      listening port: 11002 (pc1)

[IMPORTANT] TELNET: TELNET LISTENING PORT: : Device:
pc2      listening port: 11003 (pc2)

[IMPORTANT] TELNET: TELNET LISTENING PORT: : Device:
router0 listening port: 11004 (router0)

[IMPORTANT] NETWORK_MODEL_LOAD_SAVE:
config.configTransformer.Loader: Configuration
succesfully loaded from: test-topology.xml

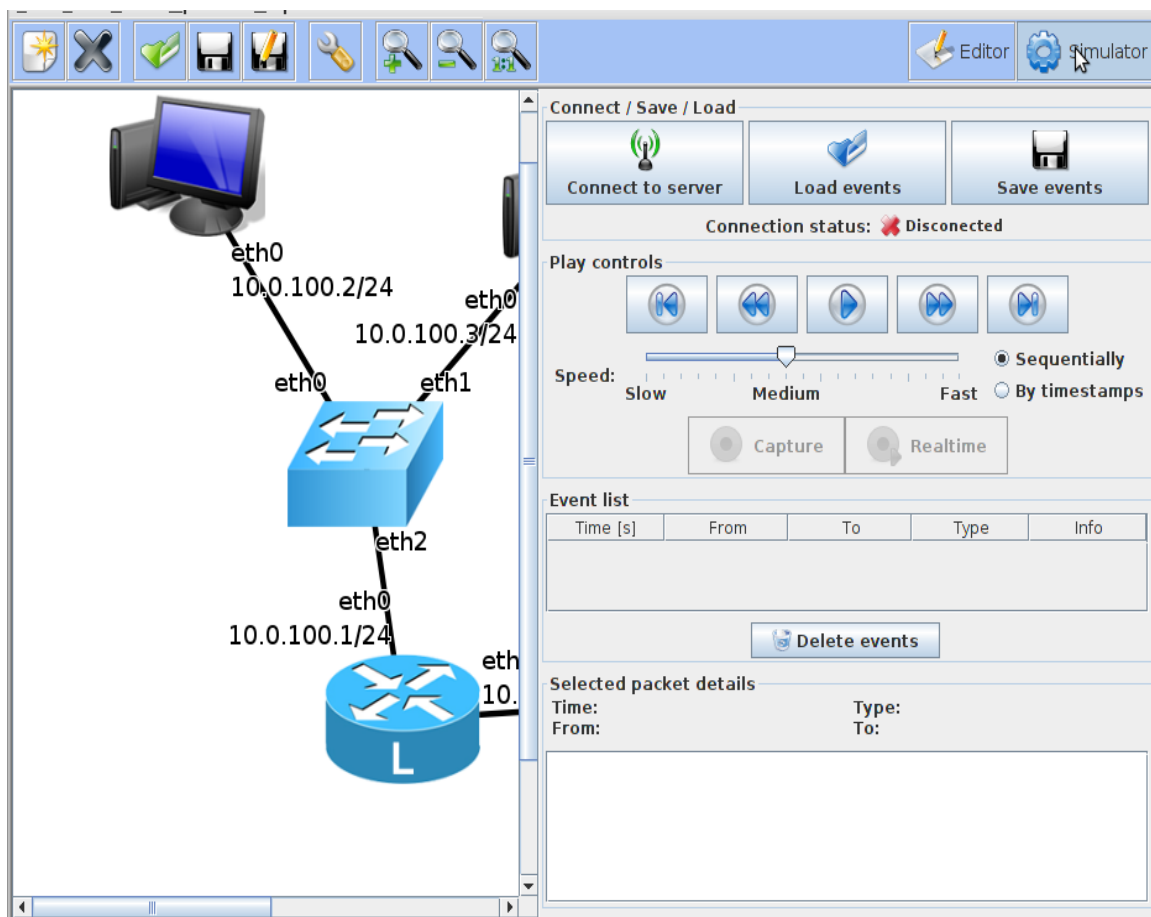
[IMPORTANT] EVENTS_SERVER: PACKET FLOW SERVER: : Server
successfully started, listening on port: 12000
```

This JAVA network simulator will not give us an error if we are executing different XML files in the front-end GUI network simulator and the back-end simulation server. So, we must take

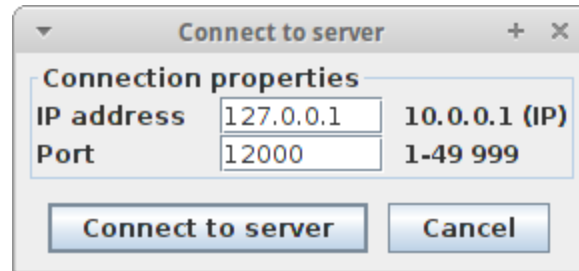
care of running the same network topology XML file in both of the front-end GUI simulator and the back-end simulation server.

#### 4.4 Connection between front-end simulator and back-end server

Click on the **Simulator** button in the top-right corner of the front-end simulator. We see the simulator controls appear on the right side of the front-end simulator. Now, we click on the **Connect to Server** button.



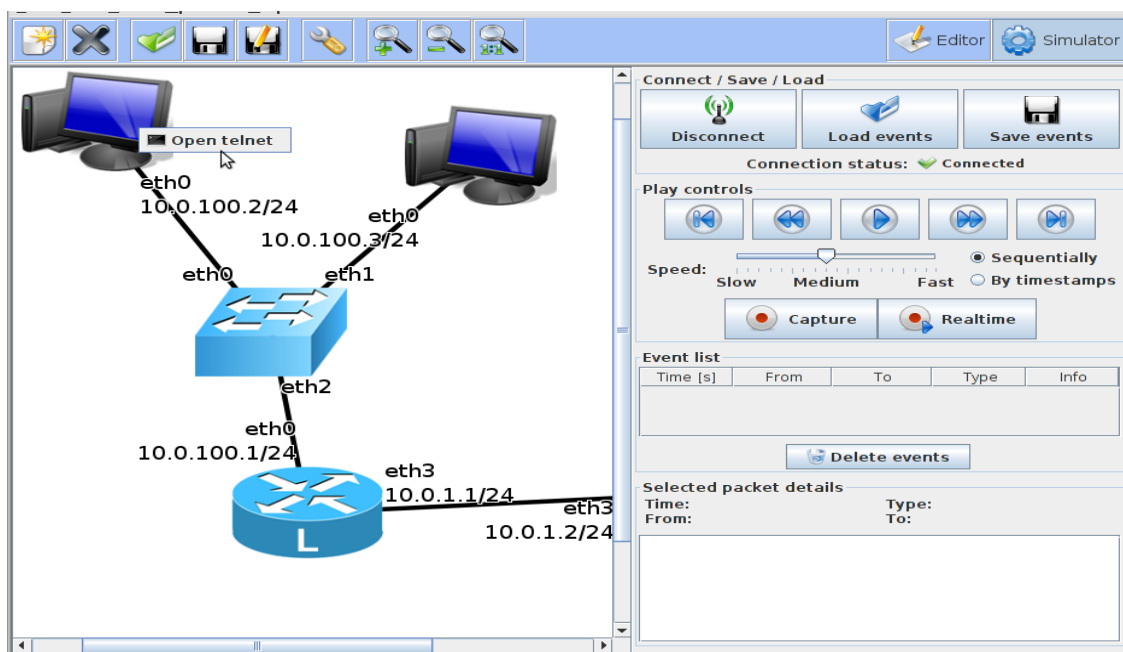
After clicking, the **Connect to server** dialogue box will appear. Input the IP Address and the port number of the computer running the back-end simulation server. After it, click the **Connect to server** button.



Now, the connection must have been established between the front-end simulator and the back-end simulation server.

#### 4.5 Opening a terminal for each element node in the network

To connect to a particular node in the network, right click on the element node and select Open Telnet from the sub-menu. This selection will open a Java terminal window connected to that node.



While using Cisco routers in the network topology, it's better to use an Xterm Emulator rather than Telnet due to its unreliability in the case of Cisco routers.

For example, to connect to node 1, enter this command in the Xterm terminal:

```
$ telnet localhost 11000
```

#### 4.6 Configuring the network element nodes

To configure the network nodes so that messages can be transmitted from any node to another node, use the steps:

1. Enable *IP Forwarding* on the two routers
2. Create static routes on each router
3. Create a default route on each PC

##### Enable IP Forwarding on routers

Change the value of the file `/proc/sys/net/ipv4/ip_forward` to 1 to enable the IP Forwarding on the routers.

In order to modify this file, use the following commands on each of the routers.

```
# editor /proc/sys/net/ipv4/ip_forward
```

In this file, replace the value from 0 with the value 1 and then save the file.

##### Creating a static route on each router

On **Router 0**, create a static route to the network:

```
# ip route add 10.0.200.0/24 via 10.0.1.2
```

On **Router 1**, create a static route to the network:

```
# ip route add 10.0.100.0/24 via 10.0.1.1
```

### Creating a default route on each host PC node

On **PC 0**, create a default route pointing to **Router 0**:

```
# ip route add default via 10.0.100.1
```

On **PC1**, create a default route pointing to **Router 0**:

```
# ip route add default via 10.0.100.1
```

On **PC2**, create a default route pointing to **Router 1**:

```
# ip route add default via 10.0.200.1
```

### Verifying the network configuration

To verify the configuration, check the IP addresses on each node are correct and valid. Execute the following command in the Xterm window to check and verify all the IP addresses related to all nodes are correct.

```
# ip addr show
```

To verify the routing table of the **Router 0**, enter the following command:

```
# ip route show
```

```
10.0.200.0/24 via 10.0.1.2 dev eth3
```

```
10.0.100.0/24 dev eth0
```

```
10.0.1.0/24 dev eth3
```

```
#
```

To verify the routing table of the **Router 1**, enter the following command:

```
# ip route show
```

```
10.0.100.0/24 via 10.0.1.2 dev eth3
```

```
10.0.200.0/24 dev eth0
```

```
10.0.1.0/24 dev eth3
```

```
#
```

## 4.7 Transmitting and capturing the data packets

After the verification of the network configuration is done, we can run simulation in the network topology and capture and handle some data packets.

### Ping Command

To capture the data packets, click on the **Capture** button on the **Simulator** tool panel on the left side of the front-end simulator. Since this simulator only supports the **Ping** and **Traceroute** so we will use the **Ping** command.

On the **PC0** Xterm terminal window, execute the command:

```
# ping 10.0.100.3
```

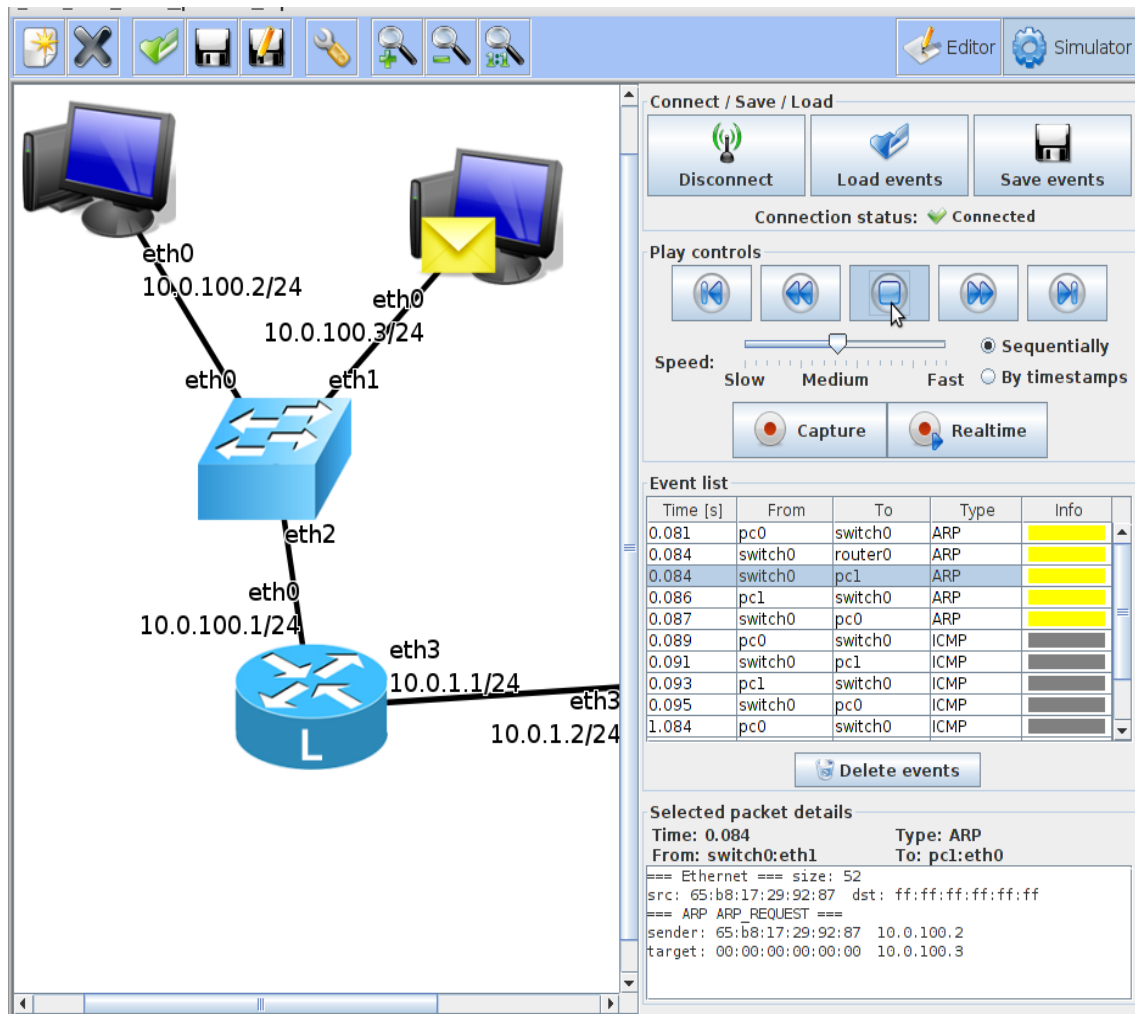
While executing this command, **PC 0** will start sending Internet Control Message Protocol echo data packets to **PC 1** and we can see these transmitted data packets being received in the JAVA network simulator's data packet window. After a little while, we can terminate the data traffic with **Ctrl+C**.

We can look for the packets being transmitted in the Event List section of the simulator. We can also click on any data packet to look out for details.

The JAVA Simulator will play the events and will animate the message passing through the simulated network by moving the abstract icons through the network. The icons are colorful in order to match the same color of every event type in the event section of the simulator.

We can change the speed of transmission of packets through the network by changing the speed. We can also run the animation sequence wise or with accordance of every event's time by selecting that button on the simulator.





## Traceroute Command

Firstly, start capturing the data messages by clicking on the **Capture** button.

Then, execute the **Traceroute** command and look for the network traffic generated.

For example: on **PC 1**, trace the routing traffic to **PC 2** by executing the following command:

```
# traceroute 10.0.200.2
```

On the **PC 1** Xterm terminal, the result of the command can be seen.

```
# pc1:/#traceroute 10.0.200.2

# traceroute 10.0.200.2

traceroute to 10.0.200.2 (10.0.200.2), 30 hops max, 60 byte packets

 1  10.0.100.1 (10.0.100.1)  25.064 ms  15.272 ms  11.69 ms

 2  10.0.1.2 (10.0.1.2)  19.507 ms  28.932 ms  25.112 ms

 3  10.0.200.2 (10.0.200.2)  27.459 ms  28.578 ms  30.476 ms

#
```

After the simulation results, stop packet capturing and playback the network events using the **Playcontrols** in the event section.

#### 4.8 Saving the network configurations

The altered configurations can be saved using the **Save** command. To save the configuration, execute the following command.

```
# save
```

## **5. CONCLUSION**

This JAVA Network Simulator provides a good network simulation experience suitable for basic IP networking analysis. It can be used to present the analysis of the effects of executing basic configuration commands on Linux PCs, Linux and Cisco routers in a simulated data network.

This simulator only supports a sub-set of networking commands that would be available on a Linux system or a Cisco router. It supports only static routing. The only commands available to generate network traffic are **ping** and **traceroute** so the only packet types a person gets to inspect are Address Resolution Protocol echo data messages and Internet Control Message Protocol echo data messages.

One of the most interesting advantages in this simulator is the animation feature that shows visual transmission of the data packets in the network topology which should be very helpful while learning the basic networking concepts. In my opinion, this animation feature and Platform Independency Property of Java programming language is the main reason one might consider using this JAVA simulator.

## **6. REFERENCES**

1. Desbrandes, F., Bertolotti, S., and Dunand, L. OPNET 2.4: an environment for communication network modeling and simulation. In Proceedings of the European Simulation Symposium (Delft, Netherlands, Oct. 1993), Society for Computer Simulation, pp.
2. Dupuy, A., Schwartz, J., Yemini, Y., and Bacon, D. NEST: A network simulation and prototyping testbed. *Communications of the ACM* 33, 10 (Oct. 1990).
3. Keshav, S. REAL: a network simulator. Tech. Rep. 88/472, University of California, Berkeley, Dec. 1988.
4. L. Breslau, D. Estrin, K. Fall, S. Floyd, J. Heidemann, A. Kelmy, P. Huang, S. McCanne, K. Varadhan, Y. Xu, and H. Yu, “Advances in network simulation,” *IEEE Computer*, pp. 59–67, May 2000.
5. V. Paxson, “End-to-End internet packet dynamics” *IEEE/ACM Trans. Networking*, vol. 7, pp. 277–292, June 1999.
6. V. Paxson, “Automated packet trace analysis of TCP implementations” in *Proc. SIGCOMM '97*, Sept. 1997.
7. Elliotte Rusty Harold, O'Reilly and Associates, *Java Network Programming*, 1997, ISBN 1-56592-227-1.
8. Behrouz A. Forouzan, *Data Communications and Networking Fourth Edition* (2007).
9. Craig Hunt, O'Reilly and Associates *TCP/IP Network Administration, Second Edition*, 1997, ISBN 1-56592-322-7.