PROJECT REPORT ON

Comparative Analysis of IP Multicasting using NS2

Report submitted to the SASTRA Deemed to be University
as the requirement for the course

CSE302: COMPUTER NETWORKS

Submitted by

Santosh Kumar Doodala
124003280, CSE

December 2022

SASTRA
ENGINEERING · MANAGEMENT · LAW · SCIENCES · HUMANITIES · EDUCATION
DEEMED TO BE UNIVERSITY
(U/S 3 of the UGC Act,1956)

THINK MERIT | THINK TRANSPARENCY | THINK SASTRA

SCHOOL OF COMPUTING
THANJAVUR, TAMIL NADU, INDIA – 613 401

SASTRA

ENGINEERING · MANAGEMENT · LAW · SCIENCES · HUMANITIES · EDUCATION

DEEMED TO BE UNIVERSITY

(U/S 3 of the UGC Act,1956)

THINK MERIT | THINK TRANSPARENCY | THINK SASTRA

SCHOOL OF COMPUTING

THANJAVUR, TAMIL NADU, INDIA – 613 401

Bonafide certificate

This is to certify that the report titled **"Comparative Analysis of IP Multicasting using NS2 "** submitted as a requirement for the course, **CSE302: COMPUTER NETWORKS** for B. Tech. is a bonafide record of the work done by **Santosh Kumar Doodala (124003280), BTECH (III YEAR) – (Computer Science Engineering)** during the academic year 2022-2023, in the School of Computing.

Project Based Work *Viva voce* held on _____

Examiner 1 **Examiner 2**

ACKNOWLEDGEMENT

First of all. I would like to express gratitude towards God Almighty for his unlimited favours. I would like to express my sincere gratitude to **Dr S.Vaidyasubramaniam. Vice-Chancellor** for his encouragement during the span of my academic life at SASTRA Deemed University.

I would forever remain grateful and would like to thank **Dr A.Umamakeswri, Dean, School of Computing** and **R. Chandramouli, Registrar** for their overwhelming support provided during my course span in SASTRA Deemed University.

I am extremely grateful to **Dr. Shankar Sriram, Associate Dean, School of Computing** for his constant support, motivation and academic help extended for the past two and half years of my life in School of Computing.

I would specially thank and express my gratitude to **Dr. Kannan Balasubramanian, Professor, School of Computing** for providing me an opportunity to do this project and for his guidance and support to successfully complete the project.

I also thank all the Teaching and Non-teaching faculty, and all other people who have directly or indirectly help me through their support, encouragement and all other assistance extended for completion of my project and for successful completion of all courses during my academic lite at SASTRA Deemed University.

In the end, I thank my parents and ns2 community forum who helped me acquire this interest in project and aided me in completing it within the deadline.

ABSTRACT

Computer networking is the branch of computer science that deals with the integration of multiple computers with communication protocols and routers with data links. The Internet boom has spawned many new network applications, and in the era of the COVID-19 pandemic, usage and dependence on the internet has skyrocketed. Recognizing the importance of leveraging limited network resources, the world leader have shifted their focus from the costly and impractical traditional one-to-one messaging (unicast) implementation to multicast.

Of the three fundamental types of IPv4 addresses: unicast, broadcast, and multicast.

Multicast is a subset of broadcasting in which datagrams must be delivered to a selected number of receivers present in the multicast group.

This project focuses on the IP multicasting technology, and contrast a Multicasting network using PIM- DM (Protocol Independent Multicast - Dense Mode), DVMRP (Distance Vector Multicast Routing Protocol), CTR (Centralized Tree) and BST (Bi-Directional Shared Tree) protocols using NS2 over a suitable network topology.

This project is for simulation done on NS2 and to study the protocol.

TABLE OF CONTENT

LIST OF FIGURES

LIST OF GRAPHS

LIST OF ABBREVIATIONS

1. IGMP - Internet Group Management Protocol
2. tr - trace file
3. DVMRP - Distance Vector Multicast Routing Protocol
4. PIM-DM - Protocol-Independent Multicast - Dense Mode
5. PIM-SM - Protocol-Independent Multicast - Sparse Mode
6. CBT - Core Based Tree
7. BST (BIDIR-PIM) - Bidirectional Shared Tree
8. RPB - Reverse Path Broadcasting
9. RPM (RPF) - Reverse Path Multicasting
10. DF - Designated Forwarder

TOOLS USED

1. Network Simulator (NS2) (version - (2.35+dfsg-3))
2. Network Animator (nam) (version - (1.15-5))
3. AWK Script
4. GNUPLOT (version - (5.2.6+dfsg1-1+deb10u1))
5. UTM for virtual machine (Version 3.2.4 (58))
6. Debian ARM (XFCE) (ARM)

CHAPTER 1 - INTRODUCTION

There are three basic types of IPv4 addresses: unicast, broadcast, and multicast.

Unicast addresses are used to send packets to a single destination. A broadcast address is used to send a datagram to an entire subnetwork.
Multicast addresses, on the other hand, are designed to deliver datagrams to a large number of hosts distributed over various subnetworks that are configured as members of a multicast group.

Multicasting is not connection oriented. Multicast datagram is delivered to destination group members with the same "best effort" reliability as a standard unicast IP datagram. This means that a multicast datagram is not guaranteed to reach all members of the group, or arrive in the same order relative to the transmission of other packets.

MULTICAST GROUPS

Individual hosts join or leave a multicast group when they require.

GROUP MEMBERS

A group membership protocol is employed by routers to learn about the presence of group members on their directly attached subnetworks. When a host joins a multicast group, it transmits a group membership protocol message for the group(s) that it wishes to receive, and sets its IP process and network interface card to receive frames addressed to the multicast group. This receiver-initiated join process has excellent scaling properties since, as the multicast group increases in size, it becomes ever more likely that a new group member will be able to locate a nearby branch of the multicast distribution tree.

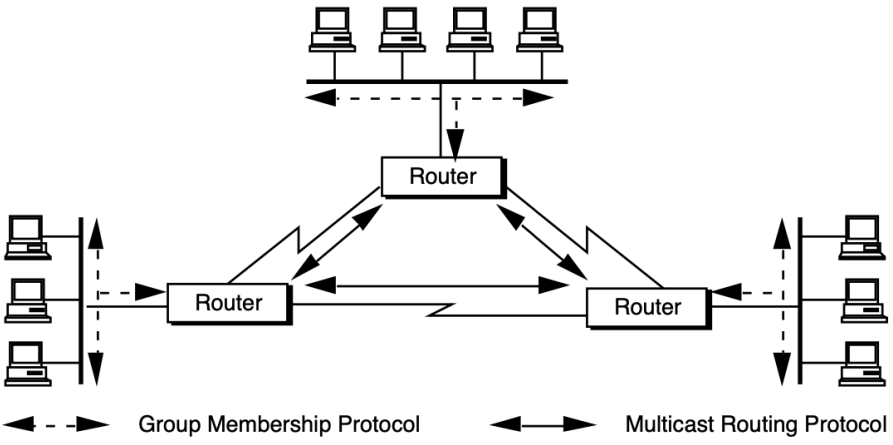


Fig 1.1: Multicast IP Delivery Service

MULTICAST ADDRESSING

The only difference between a multicast IP packet and a unicast IP packet is the presence of a “group address” in the Destination Address field of the IP header. Instead of a Class A, B, or C IP address, multicasting employs a Class D destination address.

CLASS D ADDRESS

An IP multicast group is identified by a Class D address. Class D addresses have their high-order four bits set to “1110” followed by a 28-bit multicast group ID. Multicast group addresses range from 224.0.0.0 to 239.255.255.255 .

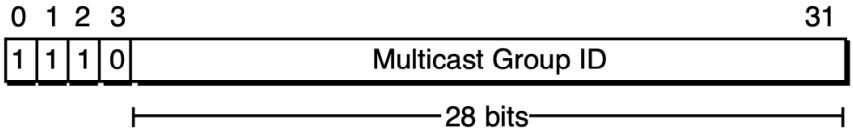


Fig 1.2 :Class D Multicast Address Format

The base address 224.0.0.0 is reserved and the multicast addresses ranging from 224.0.0.1 to 224.0.0.255 is reserved for the use of routing protocols and other low-level topology discovery or maintenance protocols.

The multicast addresses ranging from 224.0.1.0 to 239.255.255.255 are assigned to various multicast applications or remain unassigned and the range, 239.0.0.0 to 239.255.255.255 are to be reserved for site-local “administratively scoped” applications, not Internet-wide applications.

INTERNET GROUP MANAGEMENT PROTOCOL (IGMP)

IGMP runs between a host and its immediate neighbor multicast router (End system to router protocol is IGMP). The mechanisms of the protocol allow a host to notify its local router that it wishes to receive transmissions addressed to a particular multicast group.

Each host keeps track of which mcast groups they are subscribed to. Objective is to keep router up-to-date with group membership of entire LAN

Routers periodically poll the LAN to determine if known group members are still active. If there are multiple routers on a LAN running IP multicast, one of the routers is elected as the "querier" and is responsible for querying the LAN for group members.

The querier periodically sends membership query messages to all system groups. On receipt, hosts start random timers for each multicast group to which they belong. When a host’s timer for group G expires, it sends a Membership Report to group, with TTL. Other

members of G hear the report and stop their timers. Routers hear all reports, and time out non-responding groups.

UNICAST VS MULTICAST

The diagram shows how messages are sent across a topology, this represents how effectively the resource can be utilised.

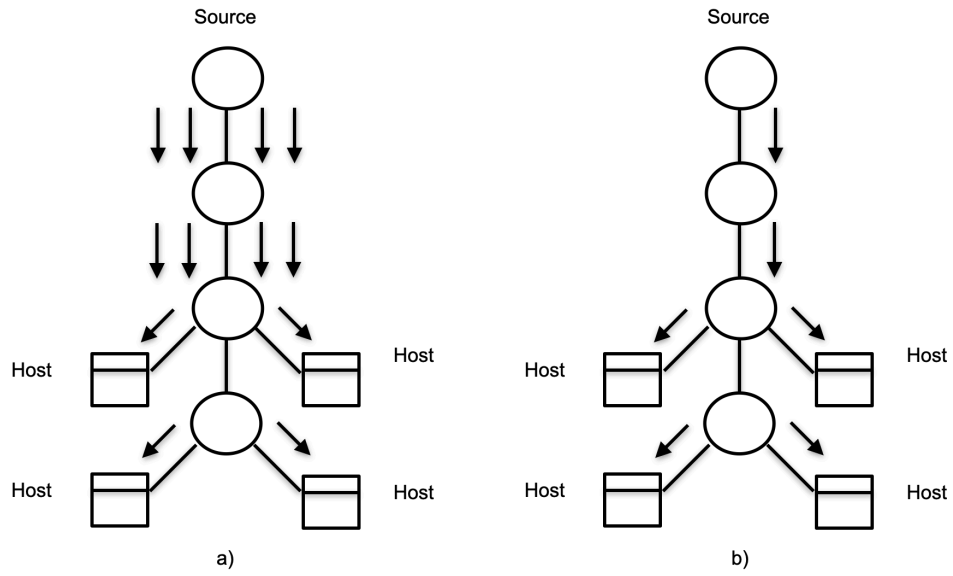


Fig 1.3: a) Unicast vs b) Multicast

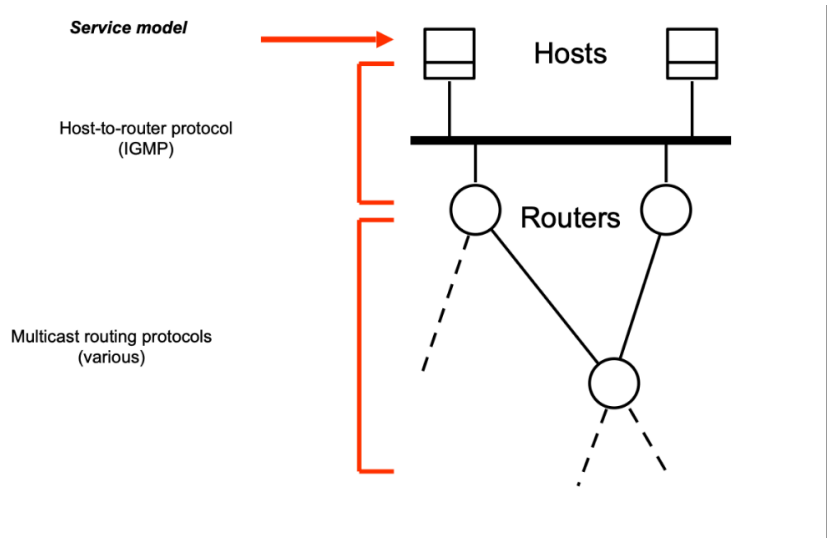


Fig 1.4: IP Multicast Architecture

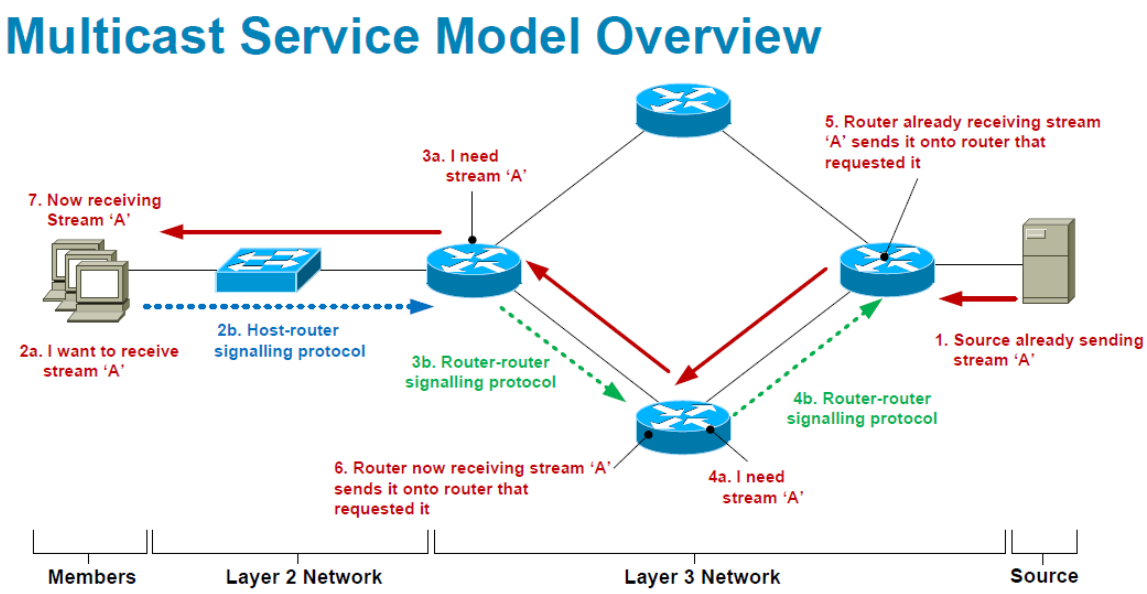


Fig 1.5: Multicast Service Model Overview

MULTICAST FORWARDING ALGORITHM

IGMP provides the final step in a multicast packet delivery service, as it only forwards multicast traffic from the local router to group members on directly connected subnets. IGMP does not handle the distribution of multicast packets between adjacent routers or across an internetwork.

To provide Internet-wide delivery services, a multicast routing protocol must be defined. Multicast routing protocols are responsible for building multicast packet distribution trees and performing multicast packet forwarding.

Two main strategies of Multicasting Protocols are:

Source- Based Tree: In this each router needs to have one shortest path tree for each group. It constructs a separate tree for each source, using the least-cost paths between the source and the members. The shortest path tree for a group defines the next hop for each network that has loyal member(s) for that group. Ex:- DVMRP, PIM-DM

Group-shared tree: In the group shared tree approach, instead of each route having m shortest path trees, only one designated router, called the centre core, a rendezvous router, takes the responsibility of distributing multicast traffic. The core has m shortest path trees in its routing table. The rest of the routers in the domain have none. If a router receives a multicast packet, it encapsulates the packet in a unicast packet and sends it to the core router. The core router removes the multicast packet from its capsule, and consults its routing table to route the packet. Ex:- CBT, BST

Necessary Jargons for Multicasting are:

Flooding: The simplest technique for delivering multicast datagrams to all routers in an internetwork is to implement a flooding algorithm. The flooding process begins when a router receives a packet destined for a multicast group. The router uses a logging mechanism to determine if this is the first time it has seen this particular packet or if it has seen it before. When a packet is received for the first time, it is forwarded on all interfaces except the interface on which it arrived, ensuring that multicast packets reach all routers in the network. If the router has seen the packet before, it simply drops it.

Spanning Tree: It defines a tree structure that connects any two routers on the Internet with only one active path. Forwarding along the branches of the spanning tree ensures that multicast packets do not loop and eventually reach all routers in the network. (Fig 1.6)

Reverse Path Broadcasting (RPB): A more efficient solution than creating a single spanning tree for the entire Internet is to create a group-specific spanning tree for each potential source subnet. These spanning trees are source-routed distribution trees originating from the directly connected subnets of the source stations. Because a group has many potential sources, a different spanning tree is built for each active (source, group) pair. (Fig 1.7)

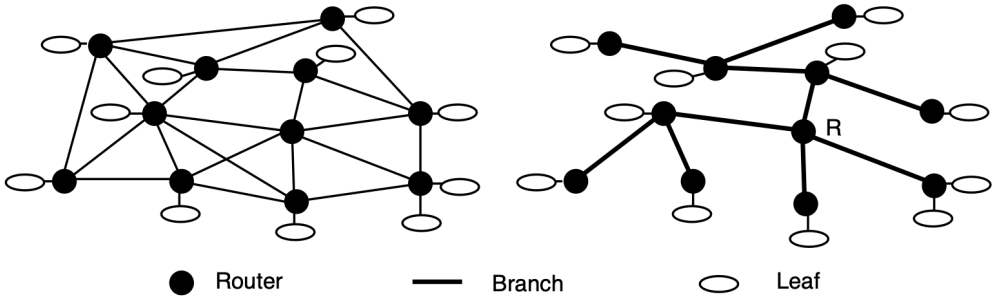


Fig 1.6: Spanning Tree

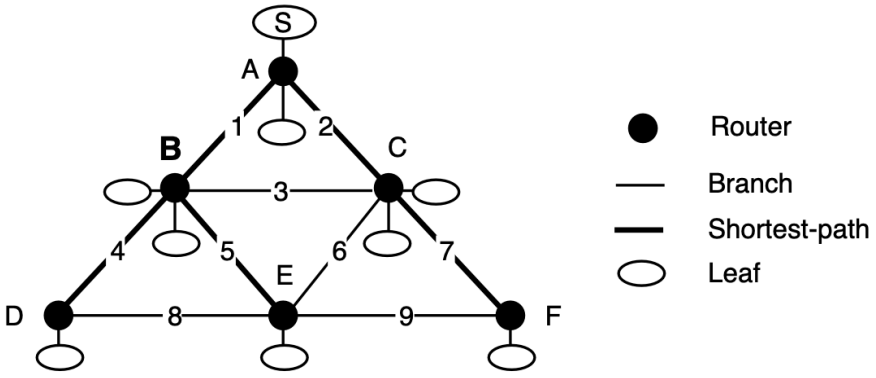


Fig 1.7: RPF

Reverse Path Multicasting (RPM): It is an extension of RPB and TRPB. RPM creates a distribution tree that spans only

- Subnetworks with group members, and
- Routers and subnetworks along the shortest path to subnetworks with group members.

RPM allows the source-rooted spanning tree to be pruned so that datagrams are only forwarded along branches that lead to members of the destination group. (Fig 1.8)

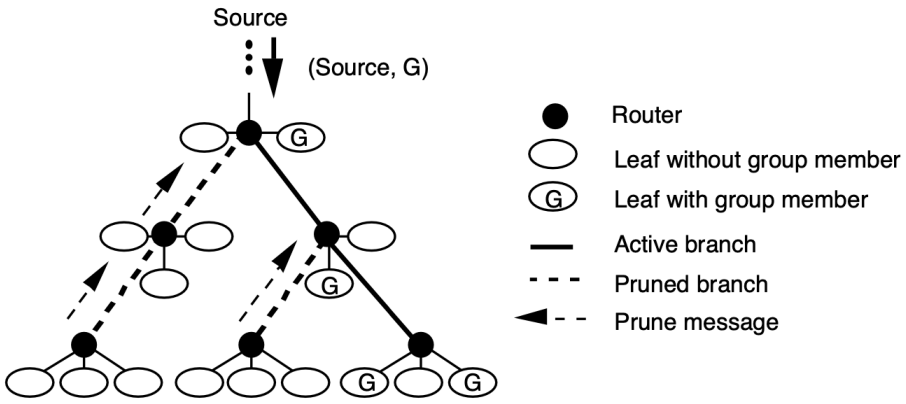


Fig 1.8: RPM

1. ***Distance Vector Multicast Routing Protocol (DVMRP).***

   The distance-vector multicast routing protocol [8] is an implementation of multicast distance-vector routing. A multicast tree is created for each source and destination host group. It implements the Reverse Path Multicasting (RPM) algorithm. Although it is a source-based routing protocol based on RIP, routers do not actually build routing tables, instead they use a unicast routing protocol.

   When a router receives a multicast packet, it forwards it. DVMRP uses a broadcast and pruning mechanism. In other words, the broadcast tree is built from source by exchanging routing information. This broadcast tree is then changed into a multicast tree using a pruning technique. More specifically, multicast datagrams are first delivered to all nodes in the tree. A leaf with no group members sends a prune message to its upstream router and discovers that the group does not exist.

   Upstream routers maintain a pruning state for this group of specific senders. The prune state becomes invalid after a configurable interval, allowing multicast to resume. A pruned branch is reestablished in the multicast tree by sending a graft message to the upstream router. A graft message starts at a leaf node and moves up the tree, sending the message to the first adjacent upstream router.

2. ***PIM-DM***

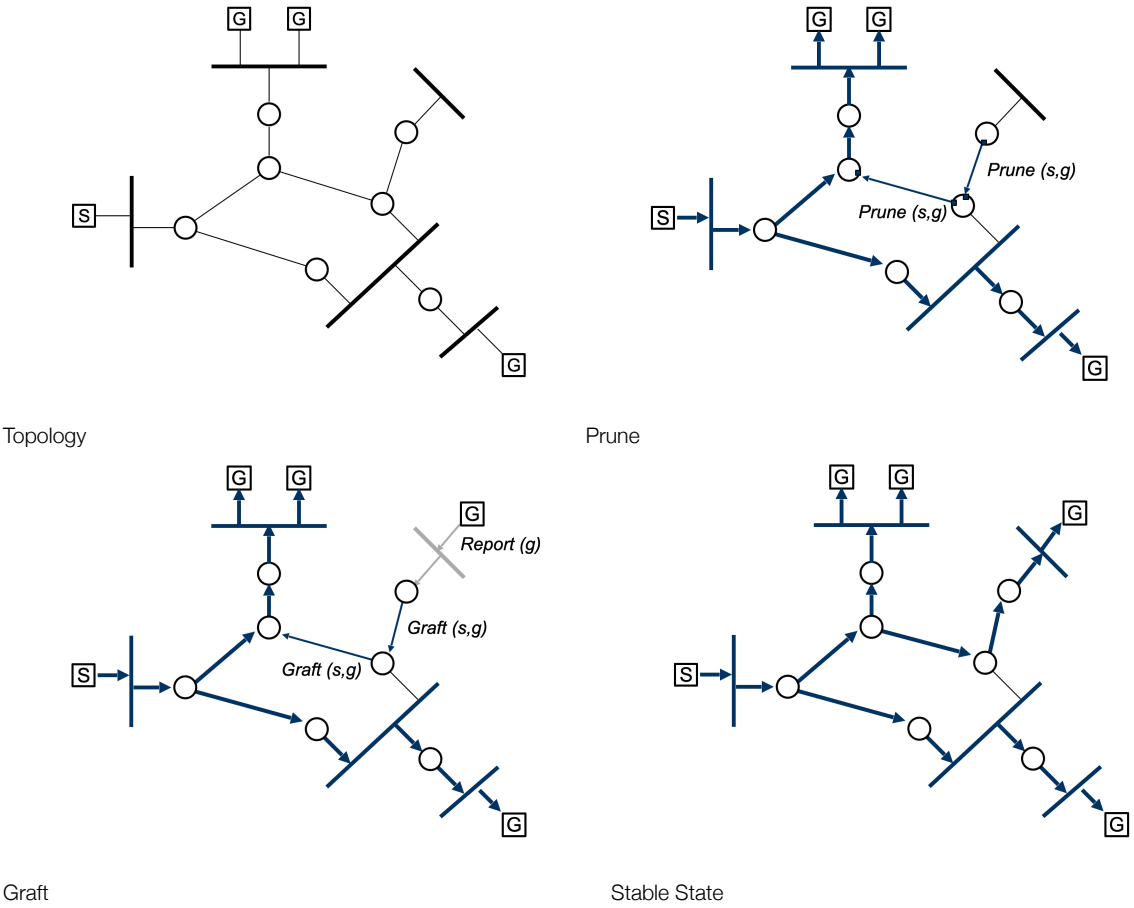


Fig 1.9: DVMRP

Protocol-Independent Multicast (PIM) routing protocol is under development by the IETF's Inter-Domain Multicast Routing (IDMR) working group. The goal of the IDMR working group is to develop a standard multicast routing protocol that can provide scalable multicast routing between domains on the Internet.

Its operation is similar to that of DVMRP. However, unlike DVMRP, it does not rely on any particular unicast protocol. Autonomous systems use a unicast protocol and each router is assumed to have a table that allows it to find the outgoing interface with the best path to the destination.

PIM-DM is intended to be deployed in resource-rich environments such as: Campus LANs where group membership is relatively dense and bandwidth is likely to be readily available.

3. ***Core Based Tree (CBT)***

Similar to PIM-SM, one of the two protocols supported by CTR.

The centralized multicast is a sparse mode implementation of multicast similar to PIM-SM.

Working:

1. Identifying Core Routers: Core selection should be done carefully. A router could become a core when a host on one of its attached subnets is willing to start a group. And in case of a single sender, the router closest to it could be a core.
2. Formation of tree: After choosing a rendezvous point (core), each router is told the unicast address of the chosen router. Each router then sends a unicast join message (similar to a graft message) to indicate that it wants to join the group. This message passes through all the routers that are located between the sender and rendezvous router. Thus a tree is formed in this way.
3. Sending multicast packets: After building the tree, each source can send multicast packets to all members of the group. It just sends the packet to the rendezvous router. This router distributes packets to all members of the group.

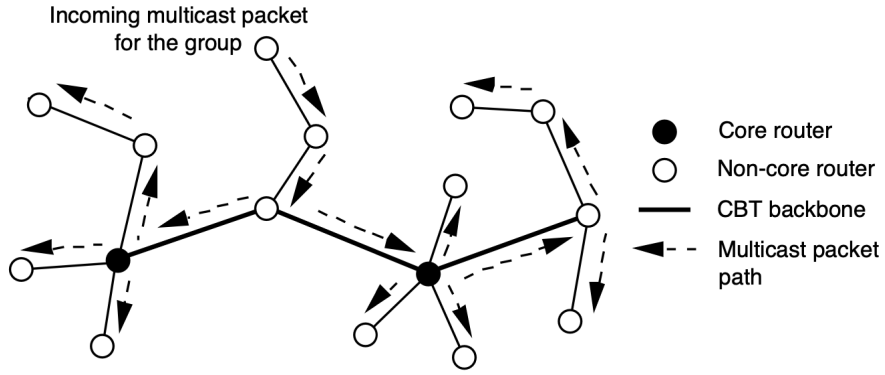


Fig 1.10 : CBT

4. ***Bidirectional Shared Tree (BST) — BIDIR-PIM***

An extension to PIM-SM.

In this, multicast data can travel in either direction of the tree for each receiver. This gives better results than others if the recipients are distributed over the network. Bidirectional trees improve routing optimality by allowing data to be transferred in both directions while maintaining minimal state information. The RPs used in this system are used to maintain routing tables for upstream and downstream receivers. All data is sent to the RP, which forwards it to the receiver using the minimal path.

Loop Prevention Mechanism- one of the router is selected as Designated Forwarder (DF).

CHAPTER 2 - PROBLEM DESCRIPTION

PROBLEM IN MULTICAST

There are two main problems with delivering multicast data streams. The first is to allow receivers to subscribe to the multicast data they want and unsubscribe to data they don't want to receive. The second is to forward these packets from the server to all recipients. So, regardless of how you connect, each packet will be forwarded only once.

We solve these problems using protocols like IGMP which handles the joining and leaving of multicast streams for individual receivers. And protocols like PIM-DM,DVMRP etc that network devices such as routers use to build and manage the multicast delivery tree structures across the network.

COMPARING VARIOUS MULTICAST TECHNOLOGY

We know various multicast protocols but to evaluate which one is better, we have to evaluate them on different scenarios, with different protocols.

Network simulators allows network designers to test the networking protocols without deploying on the actual network, one such is NS2.

Network Simulator (NS2) is an open source application available to simulate a network topology using discrete sets of events developed as a part of the Virtual Internet Testbed (VINT) project

There are several advantages in using NS2 for network simulations. Most important uses include:

- Compare network protocols and evaluate performance
- Simulate new network protocols
- Simulate large network for experimental purposes
- Simulate different Internet protocols.

WHY COMPARISON?

Eventhough there is much data about IP multicasting, there are no definite resource to consult to choose the protocol suitable for topolgy(example:- sparse topology, dense topology).

With this comparison I am able to provide the NS2 code and AWK Script, for future changes in the topology by the readers-for get an gist about it.

CHAPTER 3 - IMPLEMENTATION METHODOLOGY

SELECTING A TOPOLOGY

For comparing the protocols, we have to select a ideal topology.

As in the paper [1] the pyramid networks have very good fault tolerance properties.

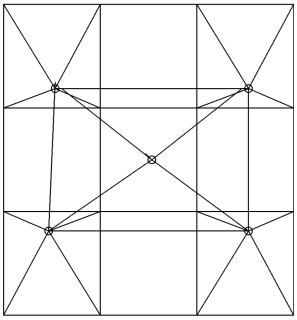


Fig 3.1: Pyramid Topology

Due to the simplicaity purpose and limitation of ns2 for wired networks, I have choosen a flat 2D pyramid.

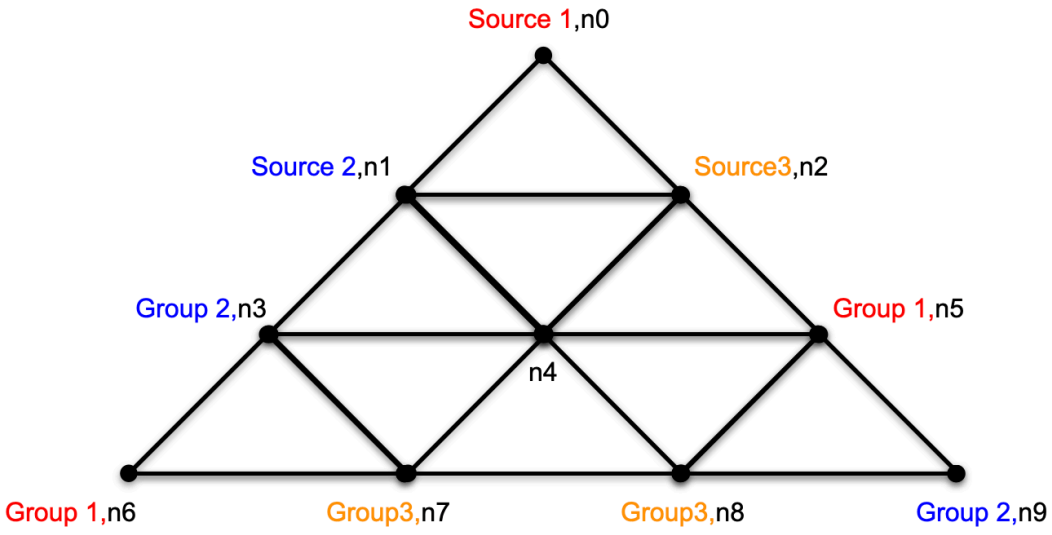


Fig 3.2: 2D Pyramid Topology - 10 nodes

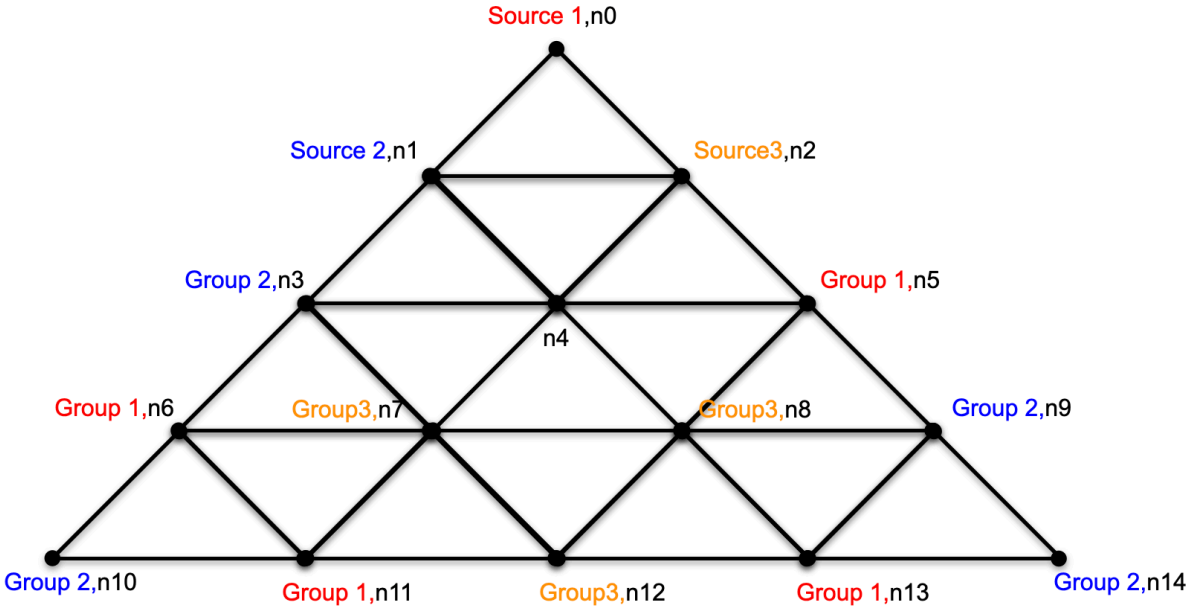


Fig 3.3: 2D Pyramid Topology - 15 nodes

NETWORK SIMULATOR (NS2)

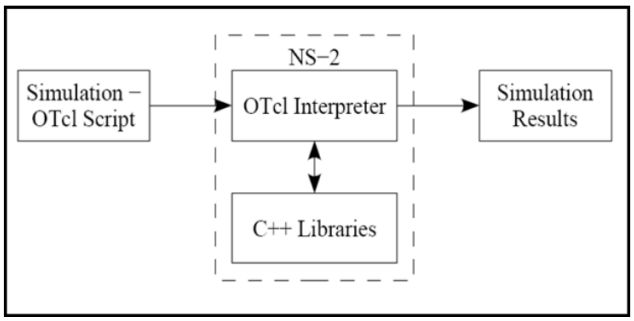


Fig 3.4: Outline of NS2

Basic Architecture

The NS2 installation leaves the user with an executable 'ns' file. When running a simulation, the entered Tcl simulation script is always used as the first input argument. Additional arguments are optional and determined based on the Tcl script entered. When the simulation is called, a simulation trace file is created as output. You can use trace files to view animated topology or draw diagrams based on simulation events. Logically, NS2 has a backend processing frontend with a C++ definition and a set of

simulation events. Objects used by C++ and oTcl are linked by TclCL using handles. Each handle has its own member procedures and variables defined in oTcl domain as instance procedures and instance variables respectively.

The output can be further analyzed by mapping the related behavior of sequential events using an AWK program or a Perl program.

Tcl is a simple scripting language which can be easily integrated to other languages. It allows easier and fast script development with a graphical user interface. It is free and compatible with various platforms. There are individual classes present in C++ as well as in oTcl class hierarchy. TclCL acts as the link between the object oriented Tcl language and C++ objects in the class hierarchy level.

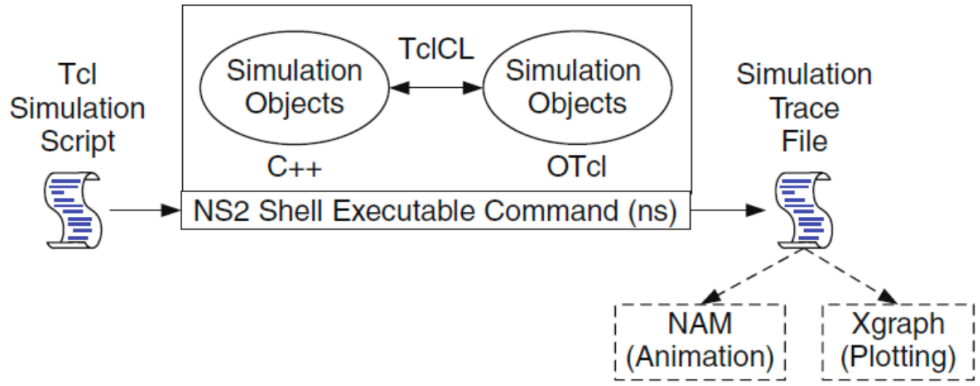


Fig 3.5: Basic Architecture

Understanding TCL Script for IP Multicasting

```
set ns [new Simulator -multicast on]
```

#Setting up DVMRP protocol, which return mrthandle as a handle

```
DM set CacheMissMode dvmrp
set mproto DM
set mrthandle [$ns mrtproto $mproto]
```

#Setting up PIMDM protocol, which return mrthandle as a handle

```
set mproto DM
set mrthandle [$ns mrtproto $mproto]
```

#Setting up CTR protocol, which return mrthandle as a handle

```
set mproto CtrMcast
set mrthandle [$ns mrtproto $mproto]
$mrthandle set_c_rp $n4
$ns at 0.0 "$n4 label \”RP\""
```

#Setting up BST protocol. —>Bidirectional PIM

```
BST set RP_($group1) $n3
BST set RP_($group2) $n4
BST set RP_($group3) $n5
$ns mrtproto BST
```

#Allocating address to a group named group1

```
set group1 [Node allocaddr]
```

#Adding source, connecting to a node, to a group and port, assigning flow_id, and setting up constant-bit-rate traffic

```
set udp1 [new Agent/UDP]
$ns attach-agent $n0 $udp1
$udp1 set dst_addr_ $group1
$udp1 set dst_port_ 0
$udp1 set fid_ 1
set cbr1 [new Application/Traffic/CBR]
$cbr1 attach-agent $udp1
$cbr1 set packetSize_ 3000 #source2
```

Create receiver to accept the packets

```
set rcvr1 [new Agent/LossMonitor]
$ns attach-agent $n3 $rcvr1
$ns at 1.0 "$n3 join-group $rcvr1 $group2"
$ns at 1.0 "$n3 label \”group2\""
```

LossMonitor - a packet sink which checks for losses

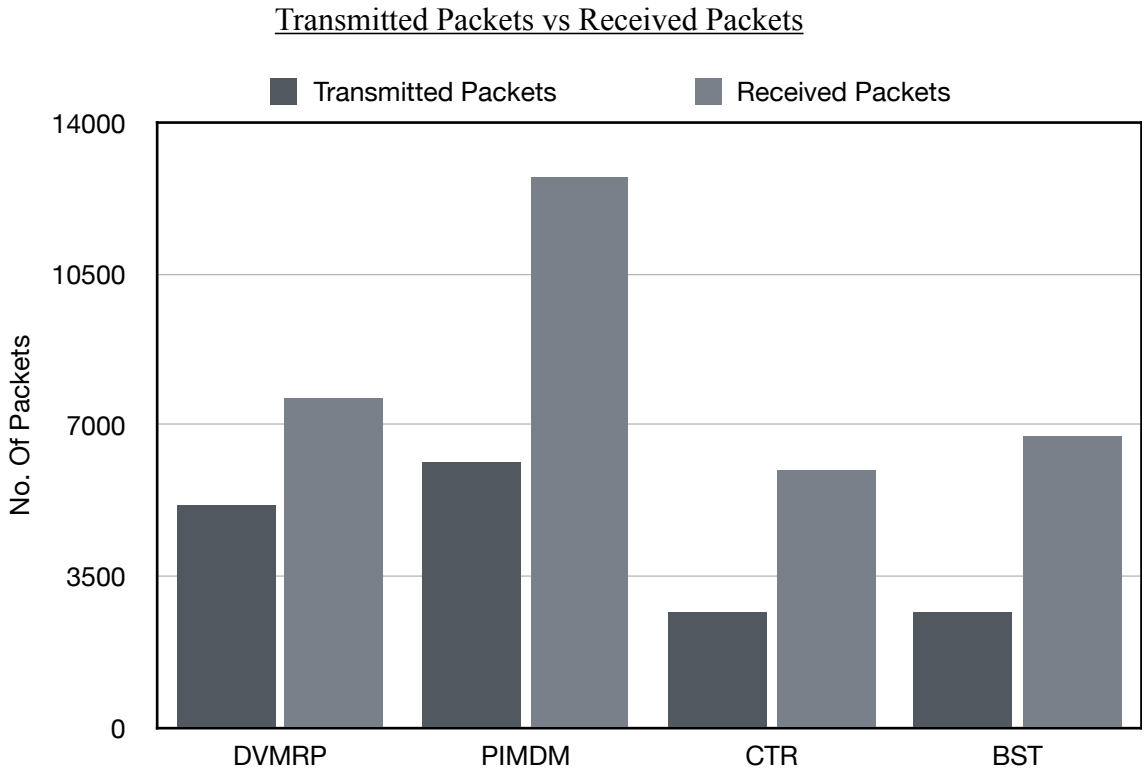
#record procedure

```
proc record {} {
global rcvr1 rcvr2 rcvr3 rcvr4 rcvr5 rcvr6 f0
global cbr1 cbr2 cbr3
set ns [Simulator instance]
set time 0.2
set bw1 [$rcvr1 set bytes_]
set bw2 [$rcvr2 set bytes_]
set bw3 [$rcvr3 set bytes_]
```

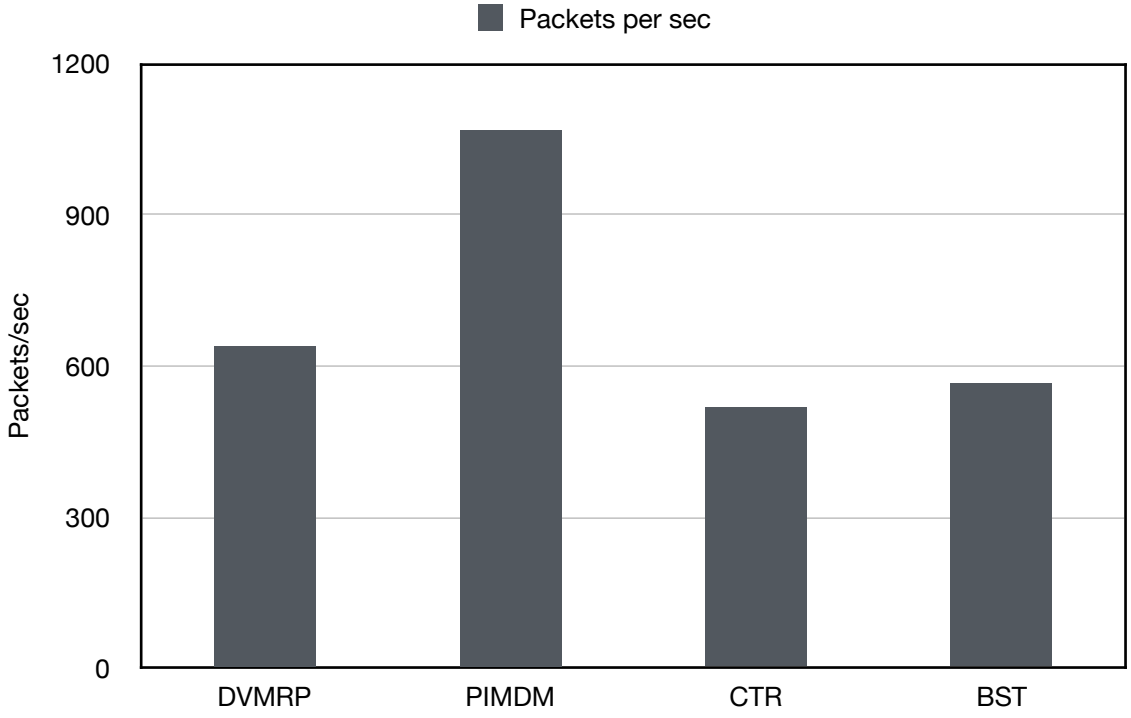
```
set bw4 [$rcvr4 set bytes_]
set bw5 [$rcvr5 set bytes_]
set bw6 [$rcvr6 set bytes_]
set now [$ns now]
puts $f0 "$now [expr ($bw1+$bw2+$bw3+$bw4+$bw5+$bw6)/
$time*8/1000000]” #Mbits/sec
$rcvr1 set bytes_ 0  #resetting the values
$rcvr2 set bytes_ 0
$rcvr3 set bytes_ 0
$rcvr4 set bytes_ 0
$rcvr5 set bytes_ 0
$rcvr6 set bytes_ 0
$ns at [expr $now+$time] "record"
}
$ns at 0.0 "record"
```

CHAPTER 4 - PERFORMANCE EVALUATION

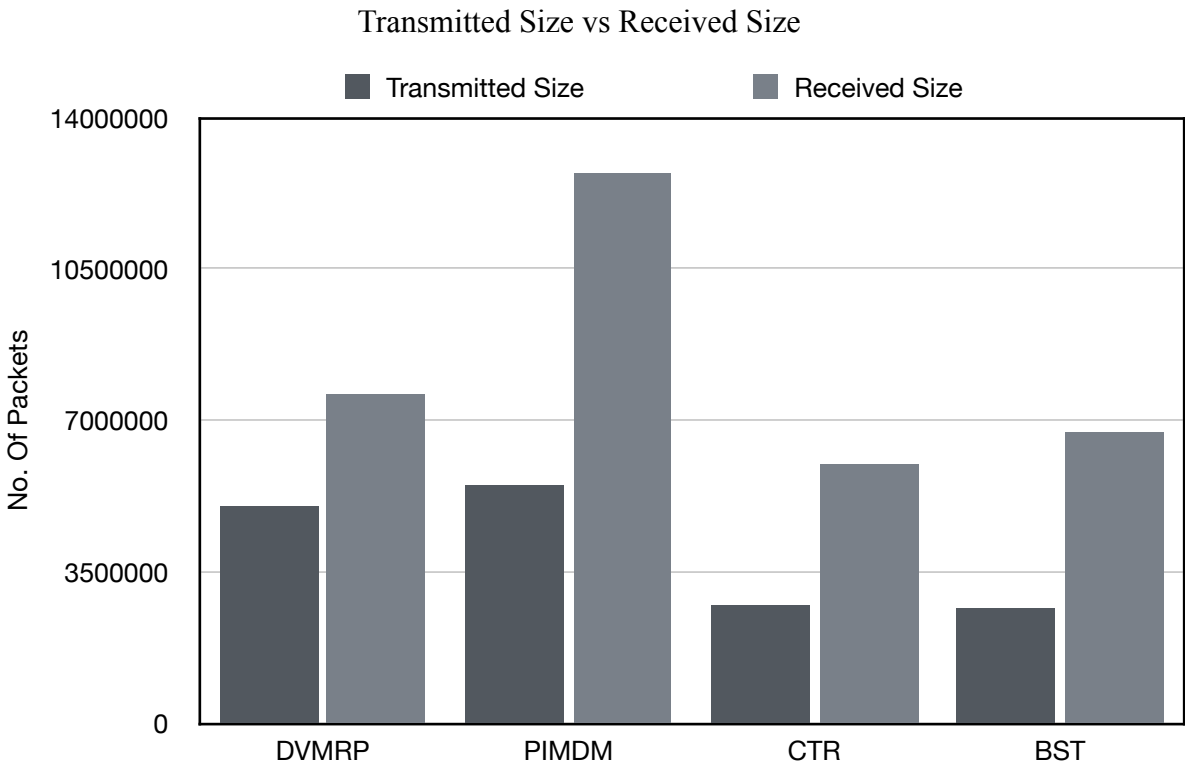
15 NODES



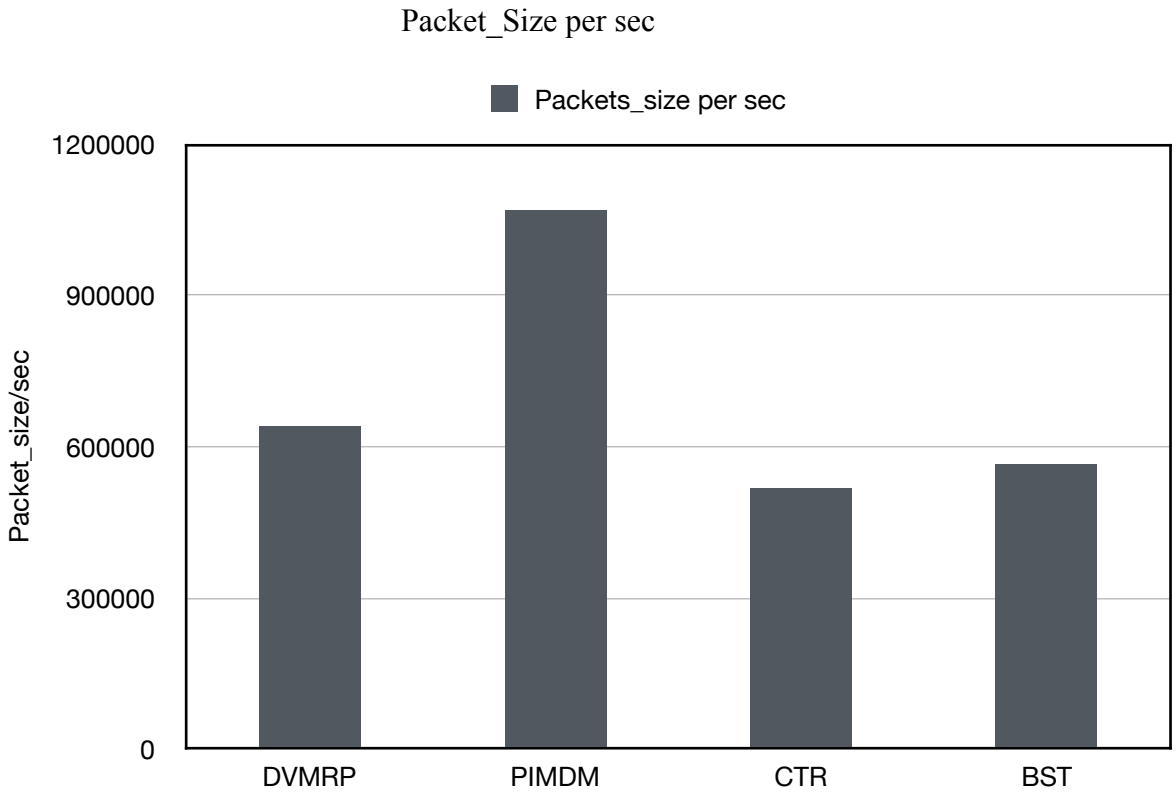
G 4.1: Transmitted Packets vs Received Packets_15 Nodes



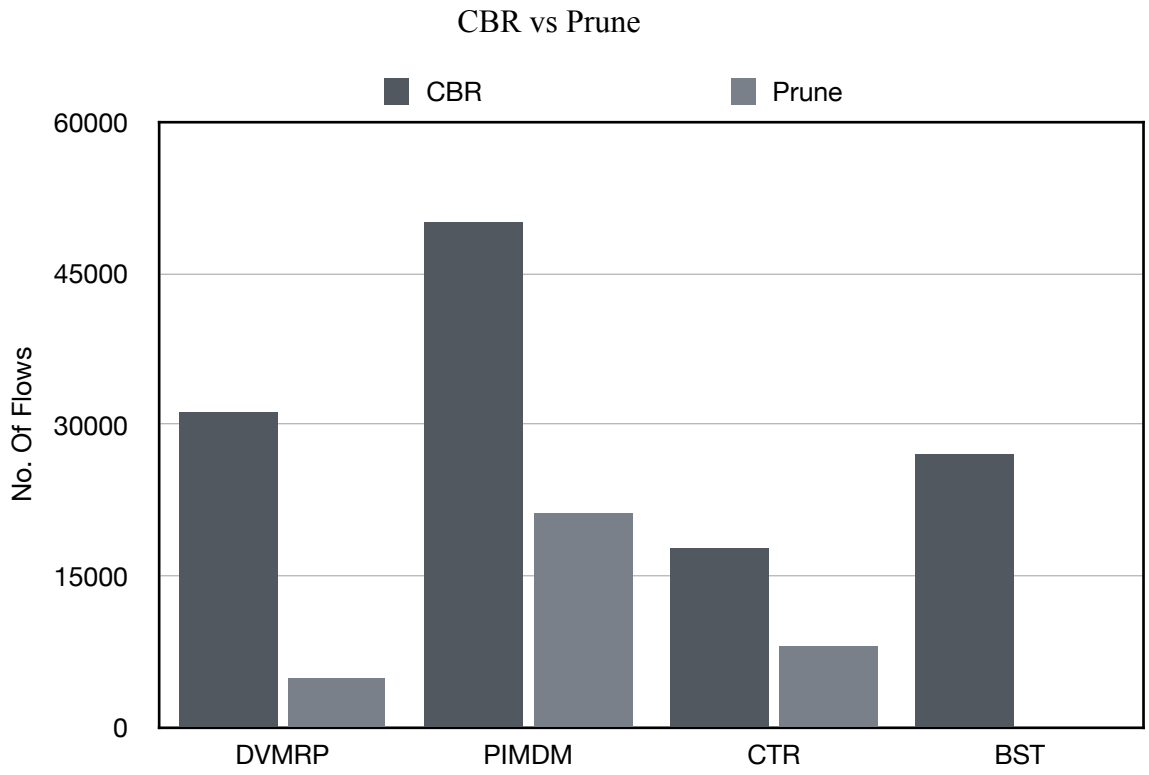
G 4.2: Packets per sec



G 4.3: Transmitted Size vs Received Size_15 Nodes

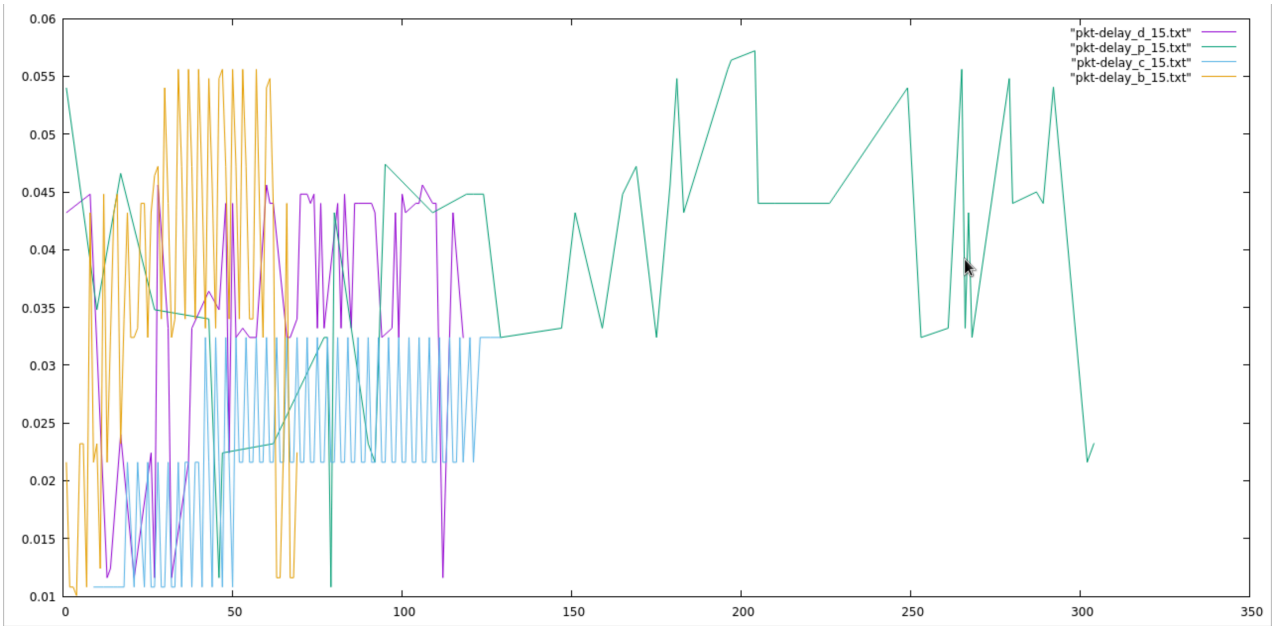


G 4.4: Packets_size per sec



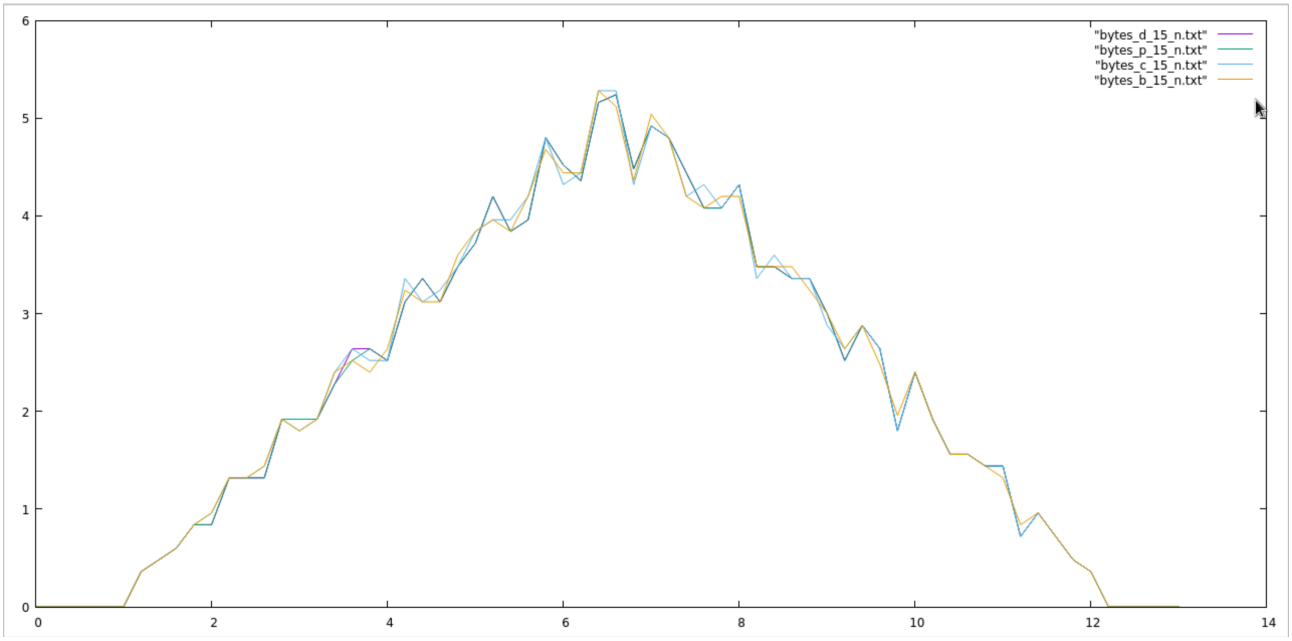
G 4.5: CBR vs Prune_15 Nodes

Packet Delay



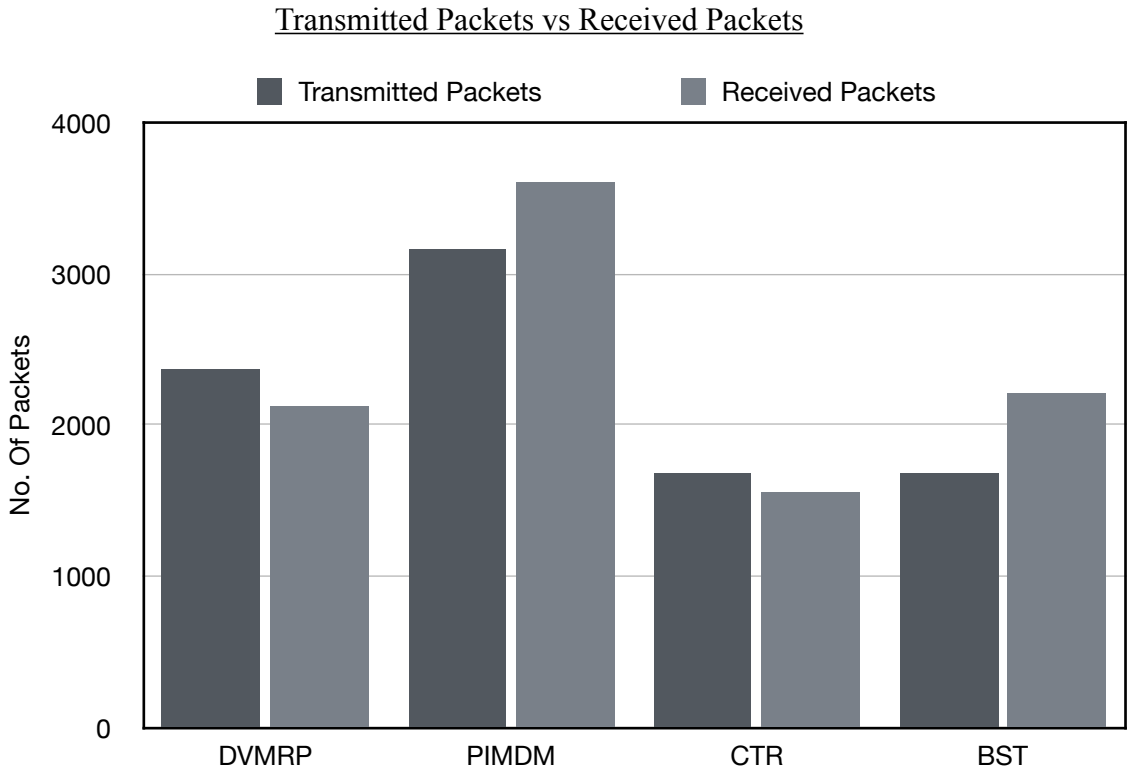
G 4.6: Packet_delay_15N, d=dvmrp, p=pimdm, c=CTR, b=bst. [X-axis = Packets, Y-axis = Time]

Bytes/sec Graph

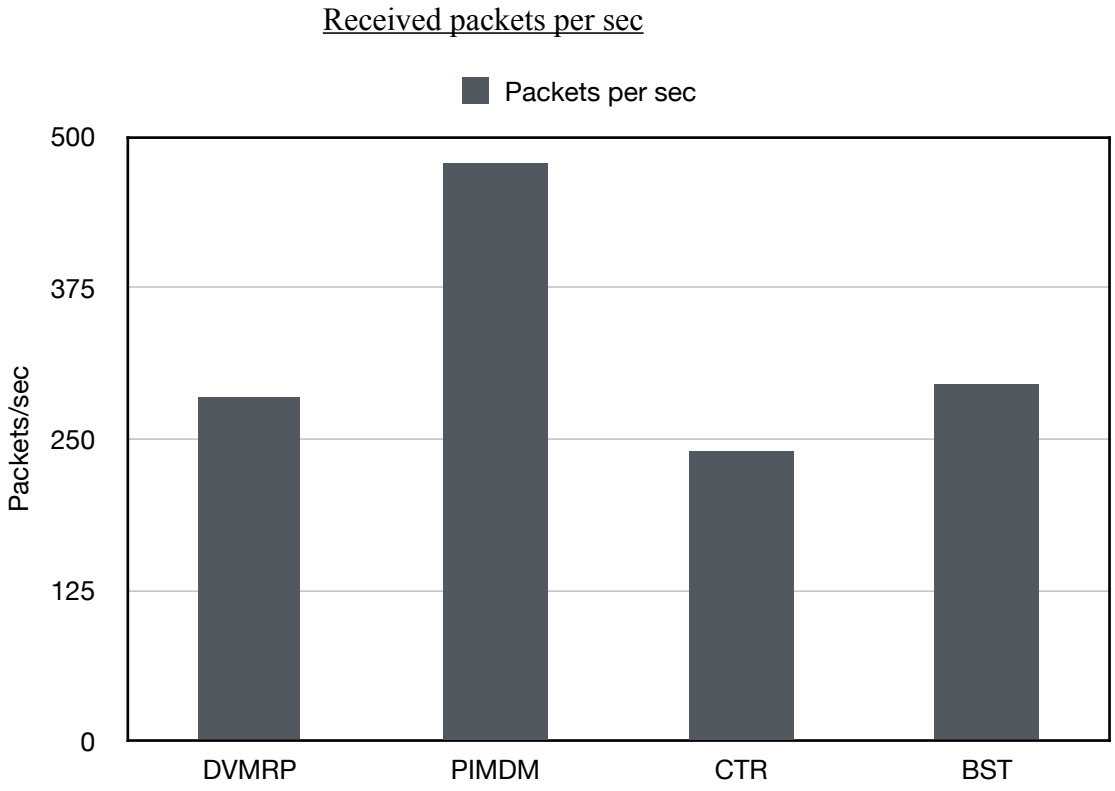


G 4.7: Bytes/sec_15Nodes [X-axis = Bytes, Y-axis = Time]

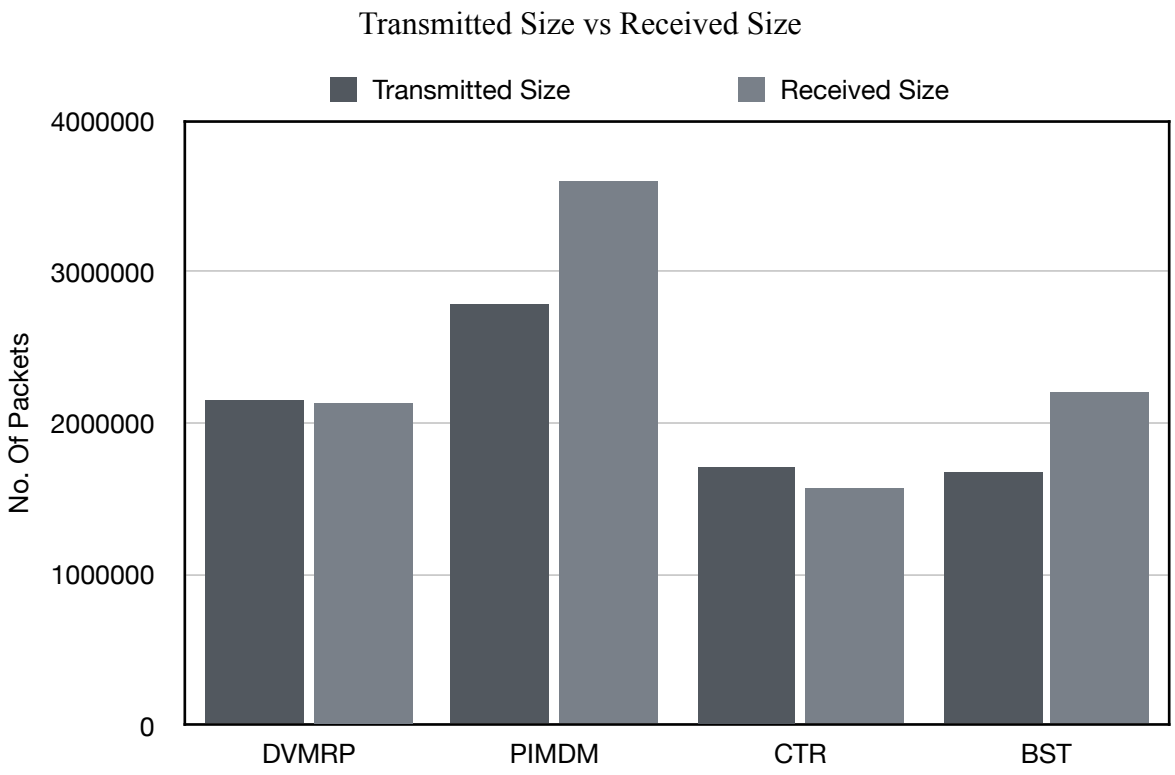
10 NODES



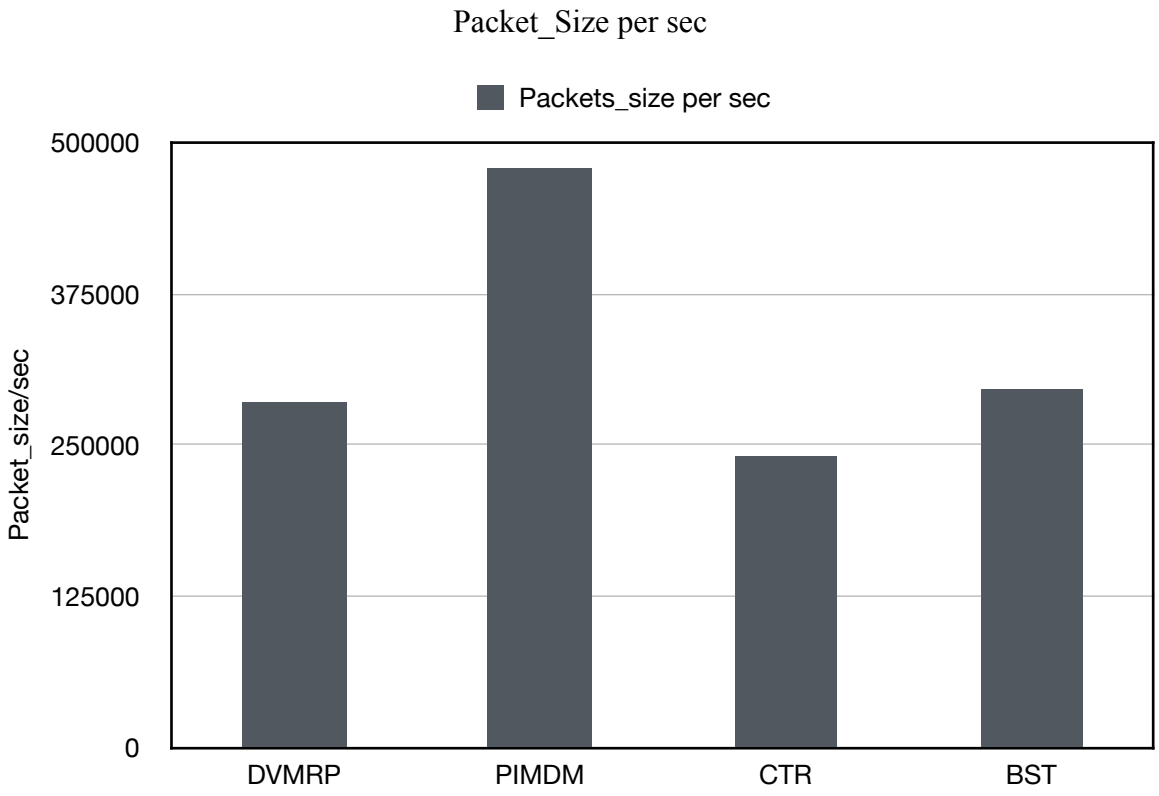
G 4.8: Transmitted Packets vs Received Packets_10 Nodes



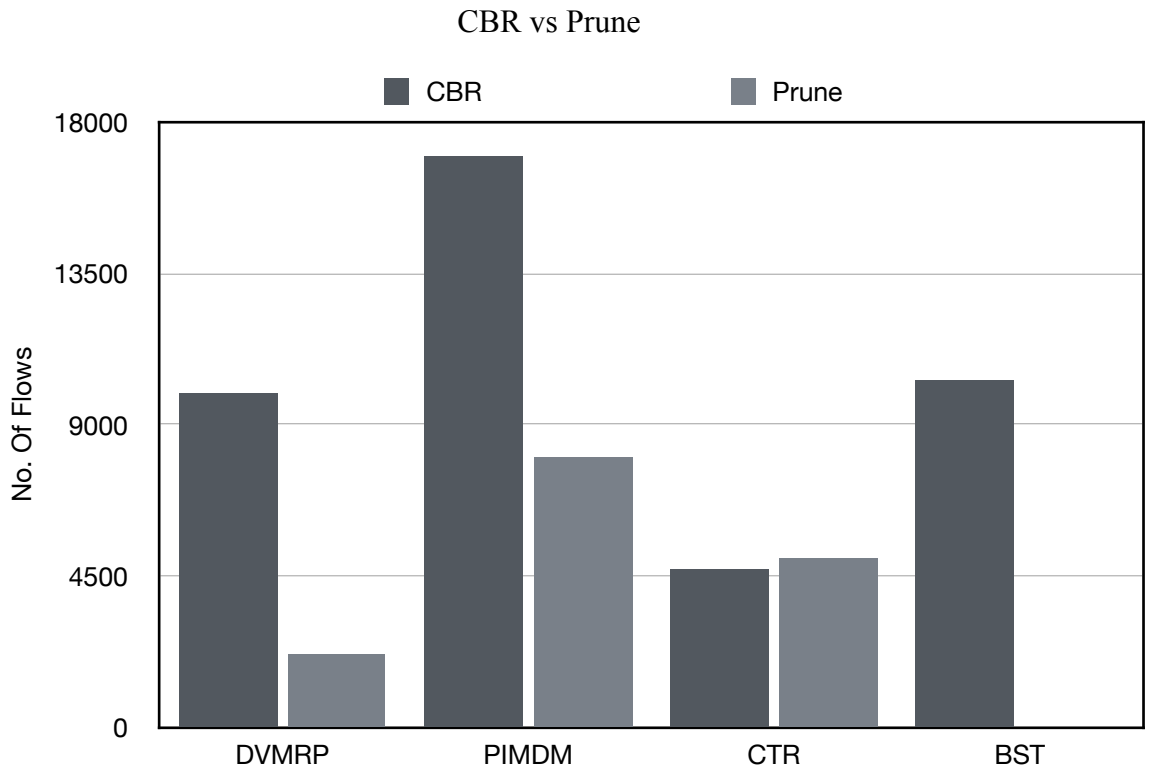
G 4.9: Packets per sec_10 Nodes



G 4.10: Transmitted Size vs Received Size_10 Nodes

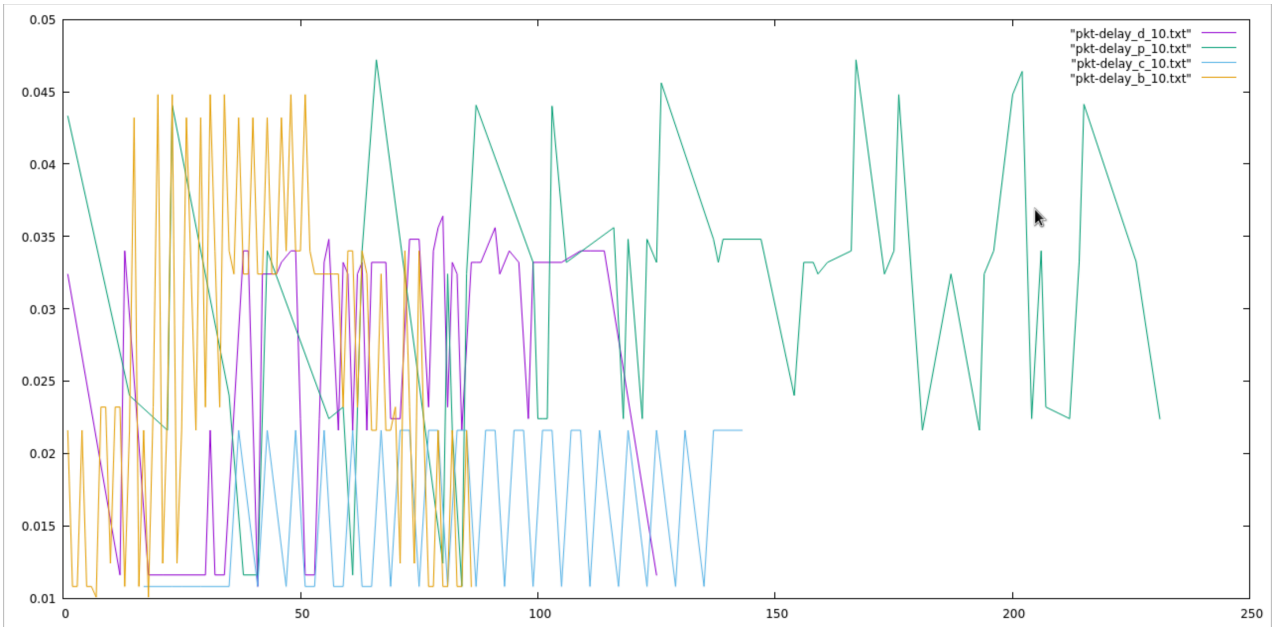


G 4.11: Packets_size per sec_10 Nodes



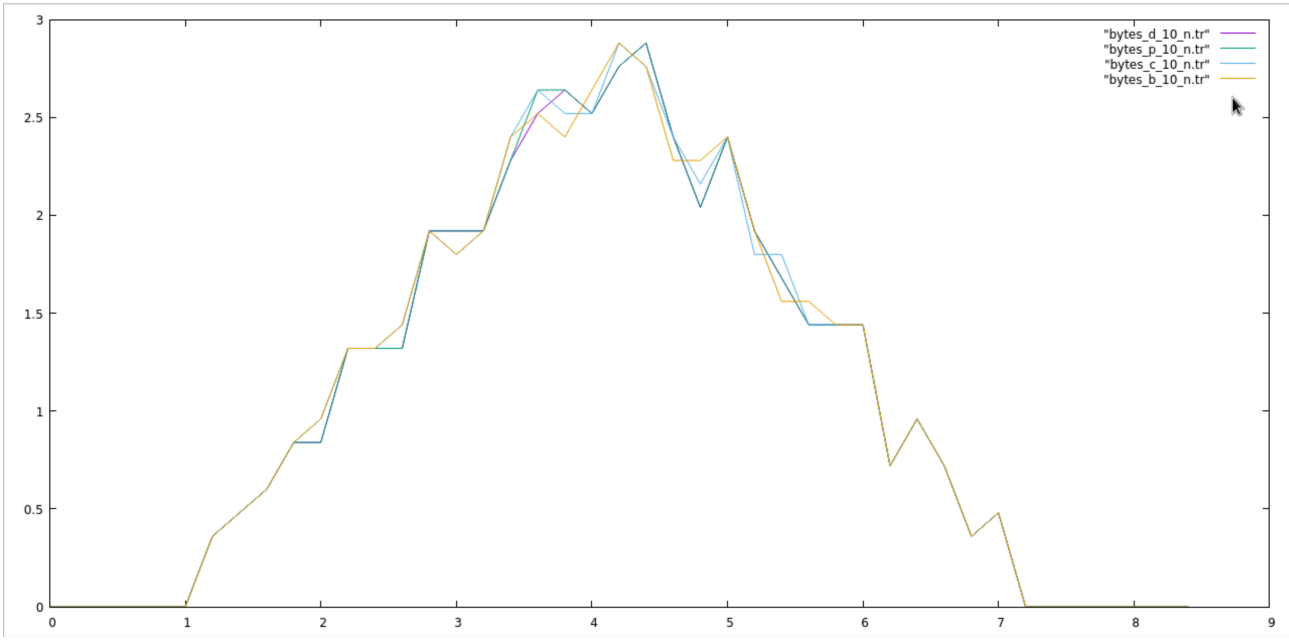
G 4.12: CBR vs Prune_10 Nodes

Packet Delay



G 4.13: Packet_delay_10N, d=dvmrp, p=pimdm, c=CTR, b=bst. [X-axis = Packets, Y-axis = Time]

Bytes/sec Graph



G 4.14: Bytes/sec_10Nodes [X-axis = Bytes, Y-axis = Time]

FORMULAS

1.CBR vs Prune:

It is the formula for comparing flow types CBR and Prune, $5 in the trace file is used to specify flow type.

2. Received Packets:

If the packets are recieved by receiver groups, and the flow id is the desired one, they are counted as received packets.

3.Average Throughput:

It is the rate at which packets can be delivered successfully from one location to another in a given amount of time. It can be represented in terms of ratio of packets sent per unit time or in terms of ratio of bytes sent per unit time. Time is represented in seconds.

4. Average Delay:

It refers to the time taken for a packet to be transmitted across a networkfrom source to destination.

5. Instantaneous Throughput(Mbits per sec) :

It shows the packet transmitted at each second.

CHAPTER 5 - SOURCE CODE

SCENARIO-1 10-NODES (DVMRP)

```
set ns [new Simulator -multicast on]
# caution:- Simulator

#Trace-file
set tf [open trace_dvmrp.tr w]
$ns trace-all $tf

#Animation-file
set nf [open animation_dvmrp.nam w]
$ns namtrace-all $nf
set f0 [open bytes_d_10_n.tr w]

#Colors for traffic
$ns color 1 red
$ns color 2 blue
$ns color 3 yellow
# the nam colors for the prune packets
$ns color 30 purple
# the nam colors for the graft packets
$ns color 31 green

#Intializing nodes
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]
set n5 [$ns node]
set n6 [$ns node]
set n7 [$ns node]
set n8 [$ns node]
set n9 [$ns node]


$n3 color blue
$n7 color yellow
$n9 color blue
$n5 color red
$n6 color red
$n8 color yellow
```

```
$ns at 0.0 "$n0 label \"source1\""
$ns at 0.0 "$n1 label \"source2\""
$ns at 0.0 "$n2 label \"source3\""



#Creating the link with drop-tail queue  # caution:-
DropTail
$ns duplex-link $n0 $n1 10Mb 10ms DropTail
$ns duplex-link $n0 $n2 10Mb 10ms DropTail
$ns duplex-link $n1 $n2 10Mb 10ms DropTail
$ns duplex-link $n1 $n3 10Mb 10ms DropTail
$ns duplex-link $n1 $n4 10Mb 10ms DropTail
$ns duplex-link $n2 $n4 10Mb 10ms DropTail
$ns duplex-link $n2 $n5 10Mb 10ms DropTail
$ns duplex-link $n3 $n4 10Mb 10ms DropTail
$ns duplex-link $n4 $n5 10Mb 10ms DropTail
$ns duplex-link $n3 $n6 10Mb 10ms DropTail
$ns duplex-link $n3 $n7 10Mb 10ms DropTail
$ns duplex-link $n4 $n7 10Mb 10ms DropTail
$ns duplex-link $n4 $n8 10Mb 10ms DropTail
$ns duplex-link $n5 $n8 10Mb 10ms DropTail
$ns duplex-link $n5 $n9 10Mb 10ms DropTail
$ns duplex-link $n6 $n7 10Mb 10ms DropTail
$ns duplex-link $n7 $n8 10Mb 10ms DropTail
$ns duplex-link $n8 $n9 10Mb 10ms DropTail

#Setting the orientation to get the pyramid topology
$ns duplex-link-op $n0 $n1 orient left-down
$ns duplex-link-op $n0 $n2 orient right-down
$ns duplex-link-op $n1 $n2 orient right
$ns duplex-link-op $n1 $n3 orient left-down
$ns duplex-link-op $n1 $n4 orient right-down
$ns duplex-link-op $n2 $n4 orient left-down
$ns duplex-link-op $n2 $n5 orient right-down
$ns duplex-link-op $n3 $n4 orient right
$ns duplex-link-op $n4 $n5 orient right
$ns duplex-link-op $n3 $n6 orient left-down
$ns duplex-link-op $n3 $n7 orient right-down
$ns duplex-link-op $n4 $n7 orient left-down
$ns duplex-link-op $n4 $n8 orient right-down
$ns duplex-link-op $n5 $n8 orient left-down
$ns duplex-link-op $n5 $n9 orient right-down
$ns duplex-link-op $n6 $n7 orient right
$ns duplex-link-op $n7 $n8 orient right
$ns duplex-link-op $n8 $n9 orient right
```

```
#Setting up DVMRP protocol, which return mrthandle as a
handle
DM set CacheMissMode dvmrp
set mproto DM
set mrthandle [$ns mrtproto $mproto]

#Setting up 3 groups namely group1,group2 and group3, and
setting up its address
set group1 [Node allocaddr]
set group2 [Node allocaddr]
set group3 [Node allocaddr]

#Now setting-up source for both groups (udp-cbr)
#source1
set udp1 [new Agent/UDP]
$ns attach-agent $n0 $udp1
$udp1 set dst_addr_ $group1
$udp1 set dst_port_ 0
$udp1 set fid_ 1
set cbr1 [new Application/Traffic/CBR]
$cbr1 attach-agent $udp1
$cbr1 set packetSize_ 3000  # I have set packet size because
i am only able to see thin stripes of packet,thus unable to
see it properly
#source2
set udp2 [new Agent/UDP]
$ns attach-agent $n1 $udp2
$udp2 set dst_addr_ $group2
$udp2 set dst_port_ 0
$udp2 set fid_ 2
set cbr2 [new Application/Traffic/CBR]
$cbr2 attach-agent $udp2
$cbr2 set packetSize_ 3000
#source3
set udp3 [new Agent/UDP]
$ns attach-agent $n2 $udp3
$udp3 set dst_addr_ $group3
$udp3 set dst_port_ 0
$udp3 set fid_ 3
set cbr3 [new Application/Traffic/CBR]
$cbr3 attach-agent $udp3
$cbr3 set packetSize_ 3000

# Create receiver to accept the packets
```

```
set rcvr1 [new Agent/LossMonitor]
$ns attach-agent $n3 $rcvr1
$ns at 1.0 "$n3 join-group $rcvr1 $group2"
$ns at 1.0 "$n3 label \"group2\""

set rcvr2 [new Agent/LossMonitor]
$ns attach-agent $n5 $rcvr2
$ns at 1.5 "$n5 join-group $rcvr2 $group1"
$ns at 1.5 "$n5 label \"group1\""

set rcvr3 [new Agent/LossMonitor]
$ns attach-agent $n6 $rcvr3
$ns at 2.0 "$n6 join-group $rcvr3 $group1"
$ns at 2.0 "$n6 label \"group1\""

set rcvr4 [new Agent/LossMonitor]
$ns attach-agent $n7 $rcvr4
$ns at 2.5 "$n7 join-group $rcvr4 $group3"
$ns at 2.5 "$n7 label \"group3\""

set rcvr5 [new Agent/LossMonitor]
$ns attach-agent $n8 $rcvr5
$ns at 3.0 "$n8 join-group $rcvr5 $group3"
$ns at 3.0 "$n8 label \"group3\""

set rcvr6 [new Agent/LossMonitor]
$ns attach-agent $n9 $rcvr6
$ns at 3.5 "$n9 join-group $rcvr6 $group2"
$ns at 3.5 "$n9 label \"group2\""


#Now the nodes are leaving group
$ns at 4.5 "$n3 leave-group $rcvr1 $group2"
$ns at 4.5 "$n3 label \" \""
$ns at 5.0 "$n5 leave-group $rcvr2 $group1"
$ns at 5.0 "$n5 label \" \""
$ns at 5.5 "$n6 leave-group $rcvr3 $group1"
$ns at 5.5 "$n6 label \" \""
$ns at 6.0 "$n7 leave-group $rcvr4 $group3"
$ns at 6.0 "$n7 label \" \""
$ns at 6.5 "$n8 leave-group $rcvr5 $group3"
$ns at 6.5 "$n8 label \" \""
$ns at 7.0 "$n9 leave-group $rcvr6 $group2"
$ns at 7.0 "$n3 label \" \""
```

```
proc record {} {
global rcvr1 rcvr2 rcvr3 rcvr4 rcvr5 rcvr6 f0
global cbr1 cbr2 cbr3
set ns [Simulator instance]
set time 0.2
set bw1 [$rcvr1 set bytes_]
set bw2 [$rcvr2 set bytes_]
set bw3 [$rcvr3 set bytes_]
set bw4 [$rcvr4 set bytes_]
set bw5 [$rcvr5 set bytes_]
set bw6 [$rcvr6 set bytes_]

set now [$ns now]
puts $f0 "$now [expr ($bw1+$bw2+$bw3+$bw4+$bw5+$bw6)/
$time*8/1000000]"


$rcvr1 set bytes_ 0
$rcvr2 set bytes_ 0
$rcvr3 set bytes_ 0
$rcvr4 set bytes_ 0
$rcvr5 set bytes_ 0
$rcvr6 set bytes_ 0
$ns at [expr $now+$time] "record"
}
$ns at 0.0 "record"

#Starting the traffic at specified tim
$ns at 0.5 "$cbr1 start"
$ns at 8.0 "$cbr1 stop"
$ns at 0.5 "$cbr2 start"
$ns at 8.0 "$cbr2 stop"
$ns at 0.5 "$cbr3 start"
$ns at 8.0 "$cbr3 stop"

#Termination, with file closing and calling a procedure
$ns at 8.5 "finish"
proc finish {} {
global ns tf nf
$ns flush-trace
close $tf
close $nf
exec nam animation_dvmrp.nam &
exit 0
}
$ns run
```

AWK SCRIPTS

1.Prune vs CBR

```
BEGIN{
count_cbr=0;
count_prune=0;
}
{
if($5=="cbr"){
count_cbr++;
}
if($5=="prune"){
count_prune++;
}
}
END{
printf("prune Message = %d\n",count_prune);
printf("CBR Message = %d\n",count_cbr);
}
```

2.Number_of_packets

```
BEGIN {
    recvdPkts = 0
    transPkts = 0
    startTime = 400
    stopTime = 0
    tput=0
}

{
    event = $1
    time = $2
    send_id = $3
    rec_id = $4
    pkt_size = $6
    flow_id = $8

    # Store start time
    if (send_id == "0"||send_id == "1"||send_id == "2" ) {
        if (time < startTime) {
            startTime = time
        }

        if (event == "+") {
```

```
            # Store transmitted packet's size
            transPkts++
        }
    }

    # Update total received packets' size and store packets
arrival time
    if ((event == "r") && (rec_id == "10" || rec_id == "12"
||  rec_id == "13" ||  rec_id == "14" || rec_id == "3" ||
rec_id == "11" || rec_id == "5" || rec_id == "6" || rec_id
== "7" || rec_id == "8" || rec_id == "9")) {
        if (time > stopTime) {
            stopTime = time
        }
        # Store received packet's size
        if (flow_id == "1"||flow_id == "2"||flow_id == "3")
{
            recvdPkts++
        }
    }
}


END {
    #printf("%i %i\n", transPkts, recvdPkts )
    tput=recvdPkts/(stopTime-startTime)
    printf("Received packets per sec: %d\n",tput)
}
```

3.Throughput

```
BEGIN {
    recvdSize = 0
    transSize = 0
    startTime = 400
    stopTime = 0
    tput=0
}

{
    event = $1
    time = $2
    send_id = $3
    rec_id = $4
    pkt_size = $6
    flow_id = $8
```

```
    # Store start time
    if (send_id == "0"||send_id == "1"||send_id == "2") {
        if (time < startTime) {
            startTime = time
        }

        if (event == "+") {
            # Store transmitted packet's size
            transSize += pkt_size
        }
    }

    # Update total received packets' size and store packets
arrival time
    if ((event == "r") && (rec_id == "10" || rec_id == "12"
||  rec_id == "13" ||  rec_id == "14" || rec_id == "3" ||
rec_id == "11" || rec_id == "5" || rec_id == "6" || rec_id
== "7" || rec_id == "8" || rec_id == "9")) {
        if (time > stopTime) {
            stopTime = time
        }
        # Store received packet's size
        if (flow_id == "1"||flow_id == "2"||flow_id == "3")
{
            recvdSize += pkt_size
        }
    }
}

END {
    printf("%i %i\n", transSize, recvdSize)
    #tput=recvdSize/(stopTime-startTime)
    #printf("%d\n",tput)
}
```

4.Packet_delay

```
#This program is used to calculate the end-to-end delay for
CBR
BEGIN {
highest_packet_id = 0;
}



{
action = $1;
time = $2;
```

```
from = $3;
to = $4;
type = $5;
pktsize = $6;
flow_id = $8;
src = $9;
dst = $10;
seq_no = $11;
packet_id = $12;
packet_num=0;

if ( packet_id > highest_packet_id ){
highest_packet_id = packet_id;
}

if ( start_time[packet_id] == 0 ){
start_time[packet_id] = time;
}

if ((flow_id == 2 || flow_id == 1 || flow_id == 3) && action
!= "d" ) {
if ( action == "r" ) {
end_time[packet_id] = time;
}
else {
end_time[packet_id] = -1;
}
}
}

END {
for ( packet_id = 0; packet_id <= highest_packet_id;
packet_id=packet_id+15 ) {
packet_num++;
start = start_time[packet_id];
end = end_time[packet_id];
packet_duration = end - start;
if ( start < end ) printf("%f %f\n", packet_num,
packet_duration);
}
}
```

TOPOLOGY

10 Nodes

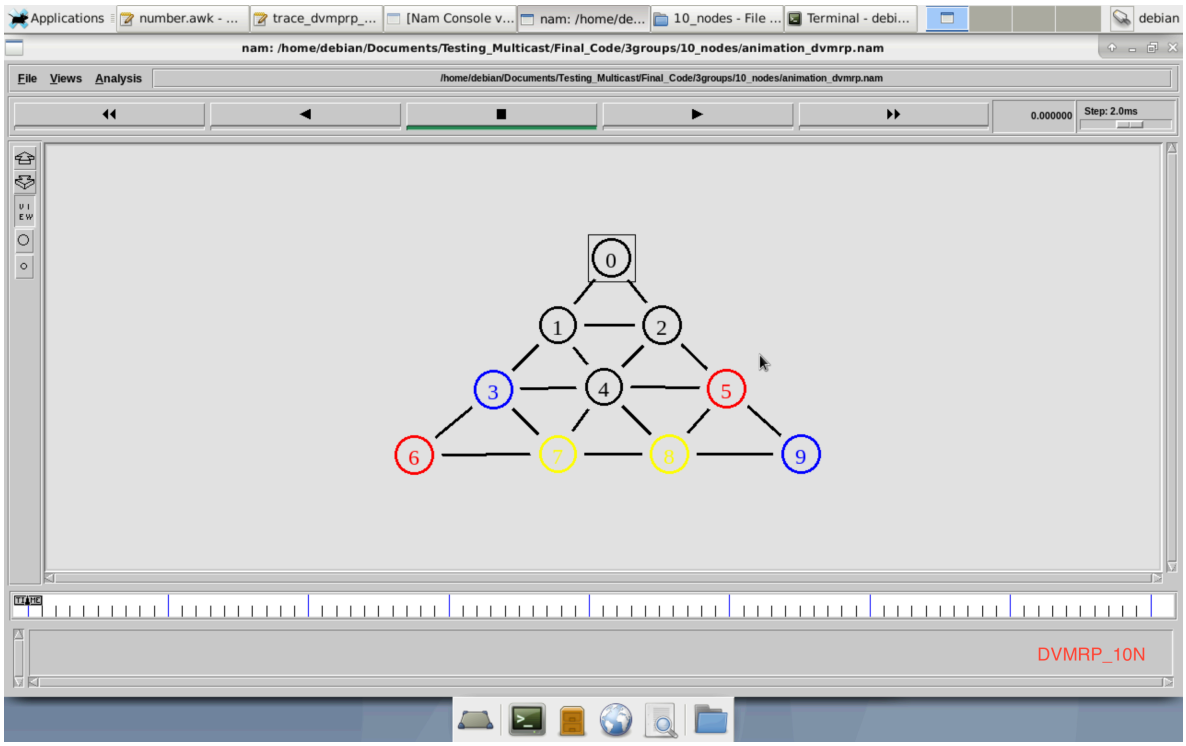


Fig 5.1: DVMRP_10N_SS

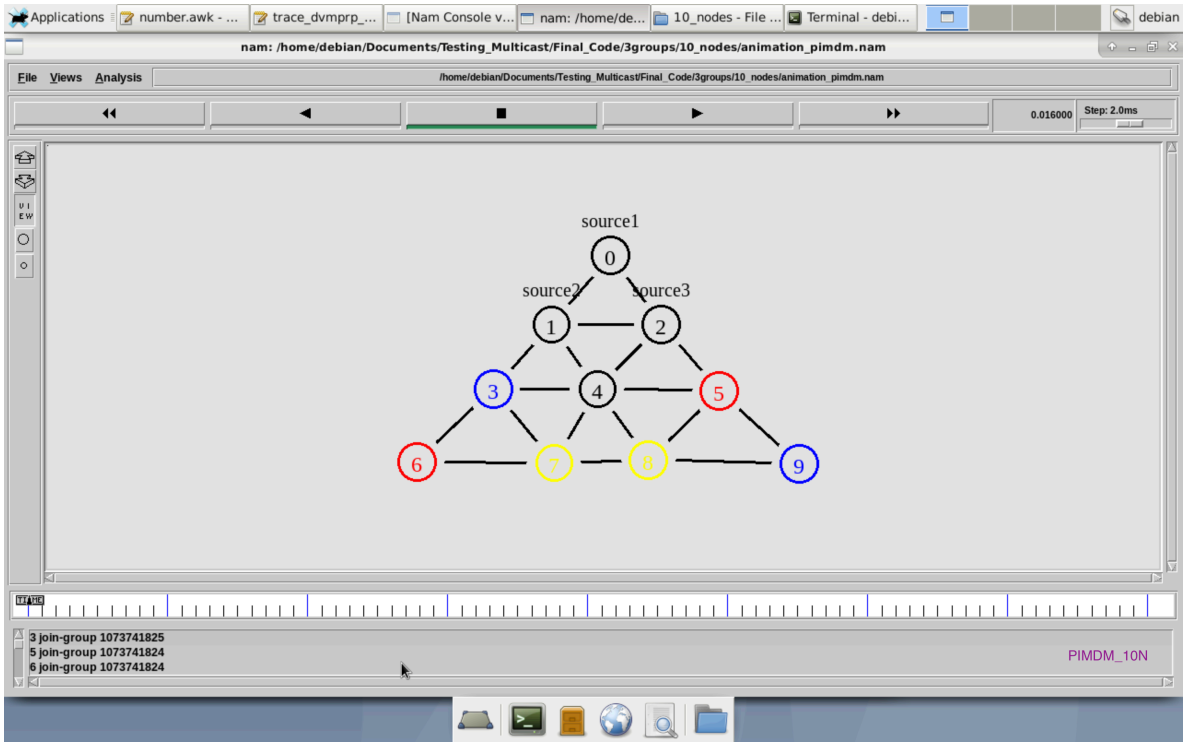


Fig 5.2: PIMDM_10N_SS

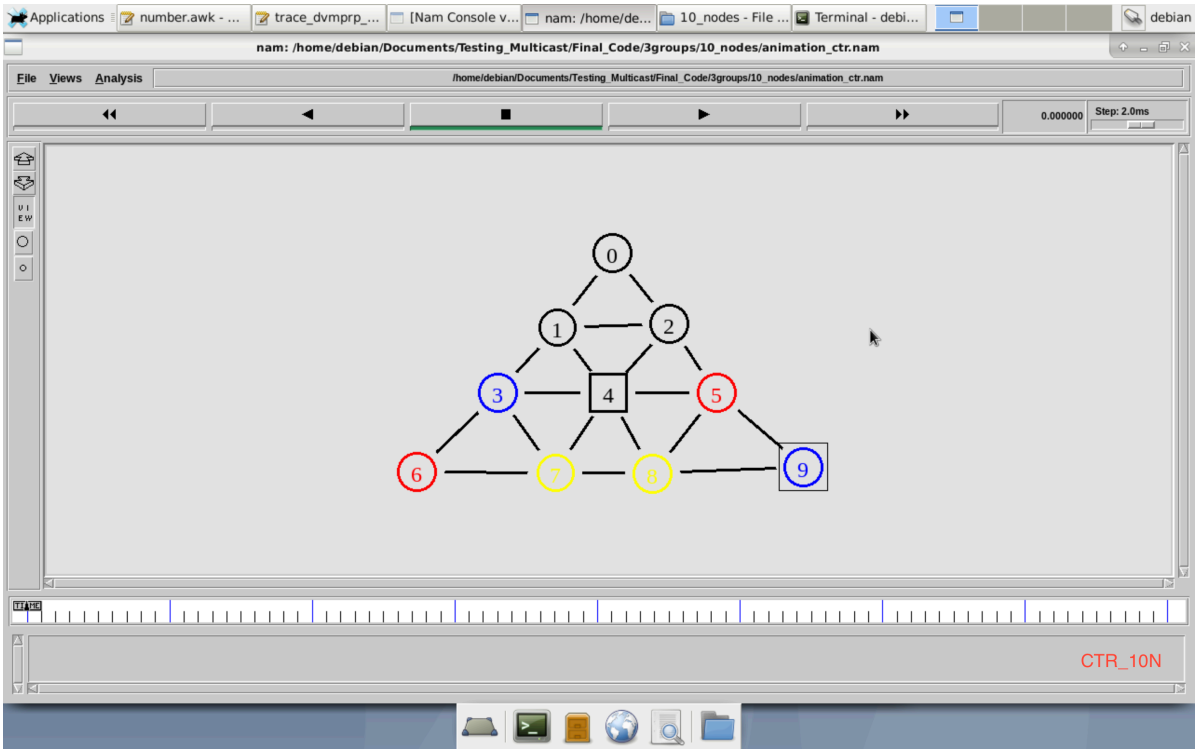


Fig 5.3: CTR_10N_SS

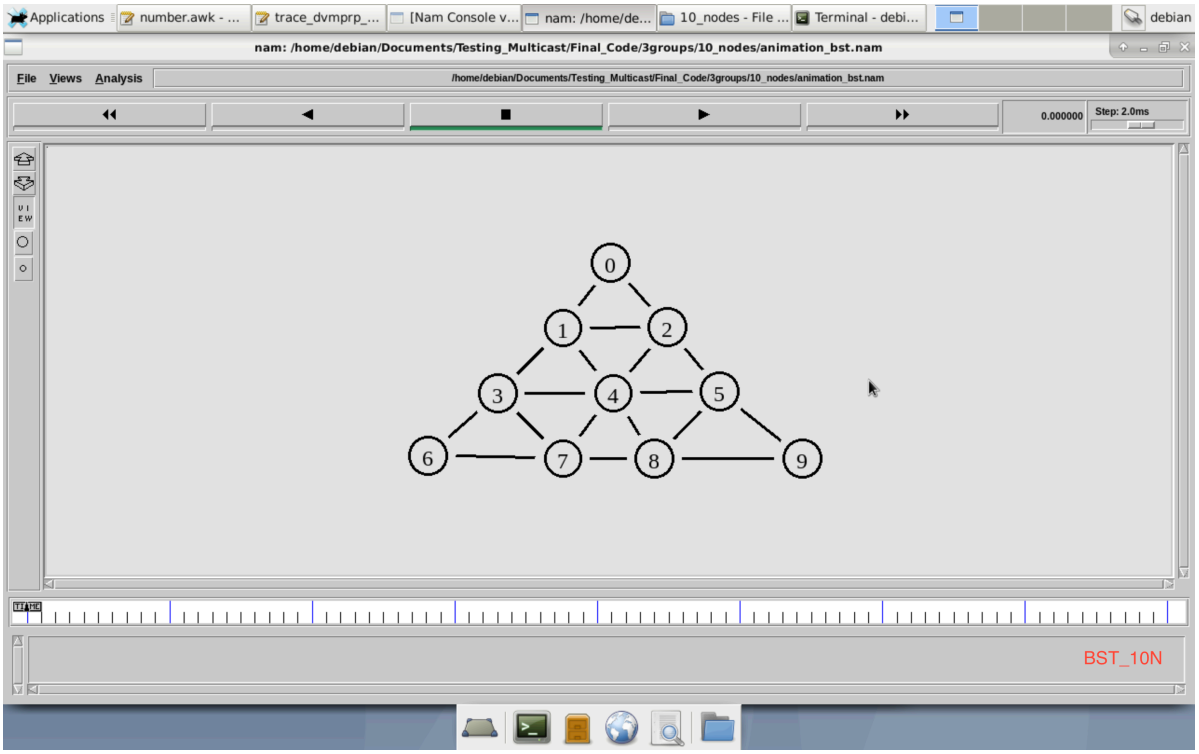


Fig 5.4: BST_10N _SS

15 Nodes

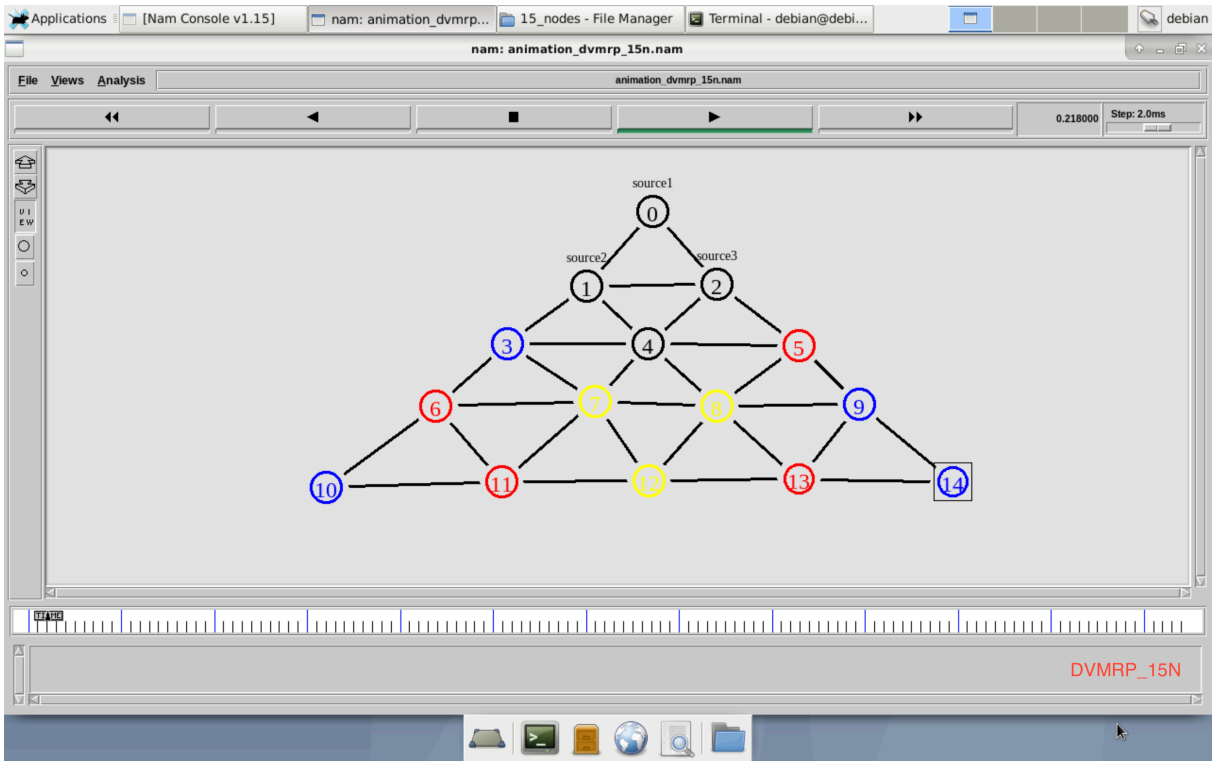


Fig 5.5: DVMRP_15N_SS

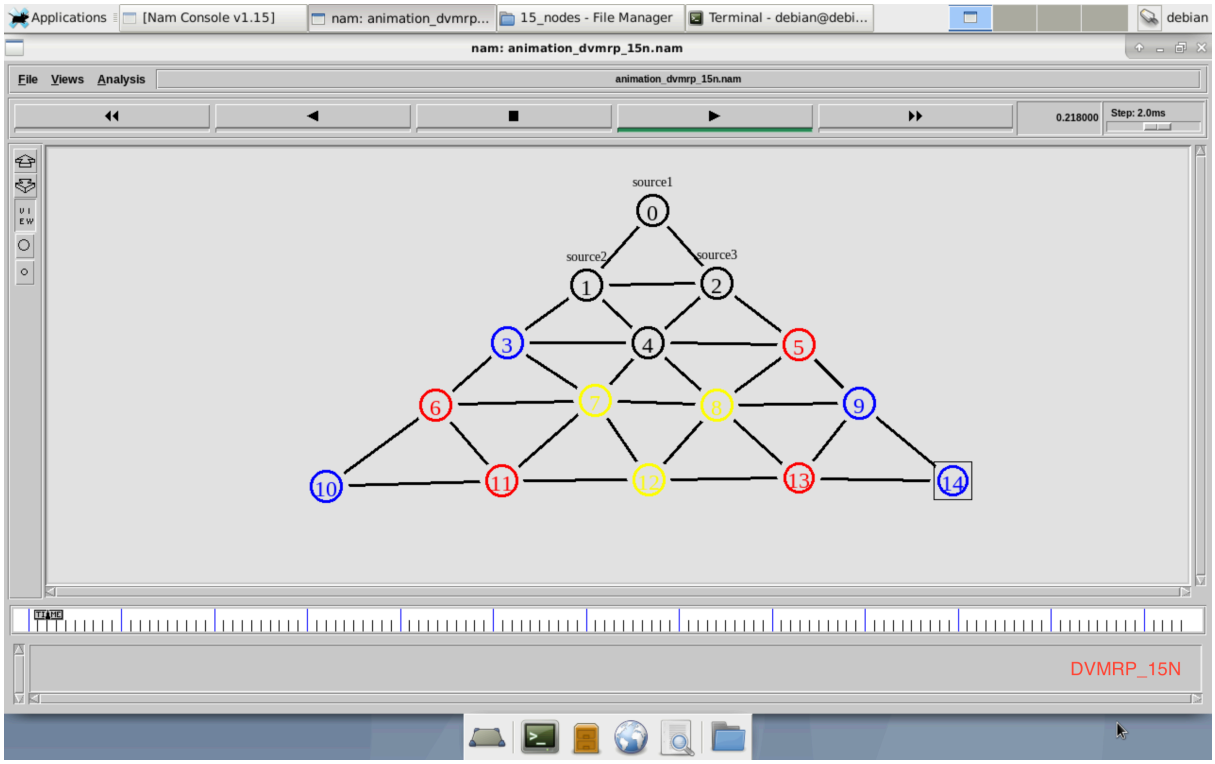


Fig 5.6: PIMDM_15N_SS

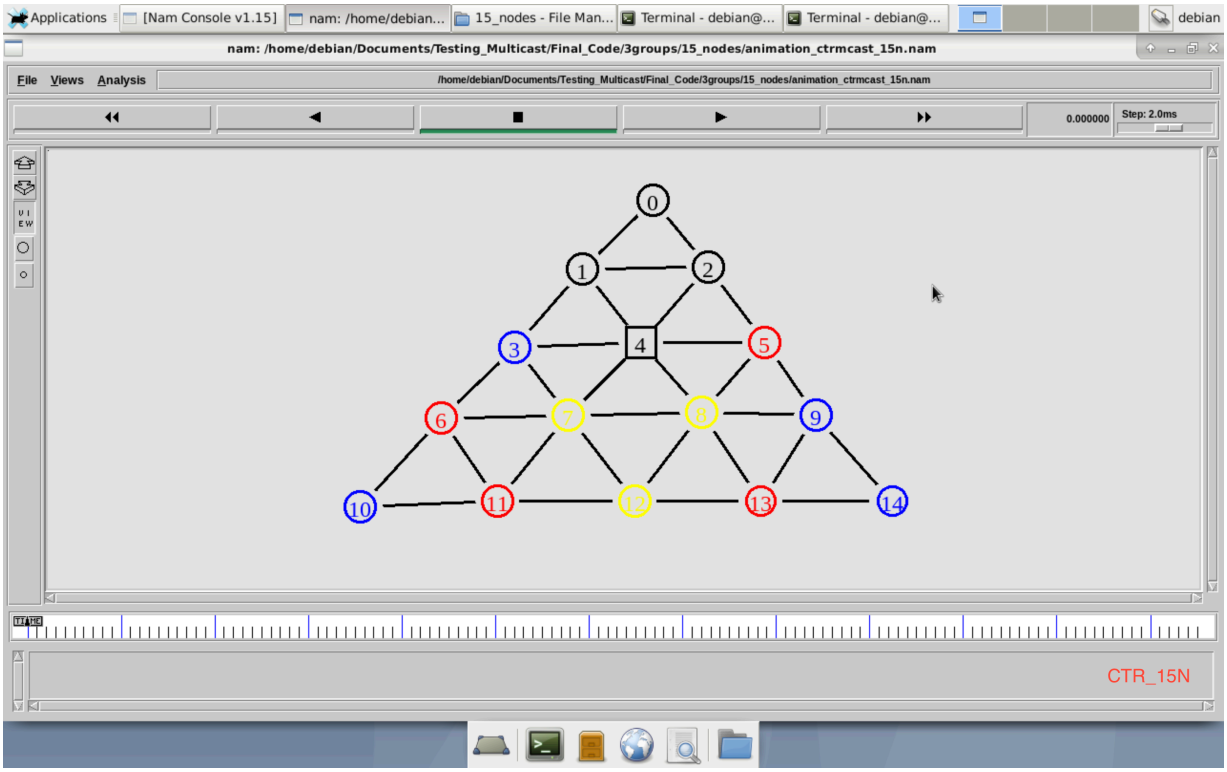


Fig 5.7: CTR_ 15N_SS

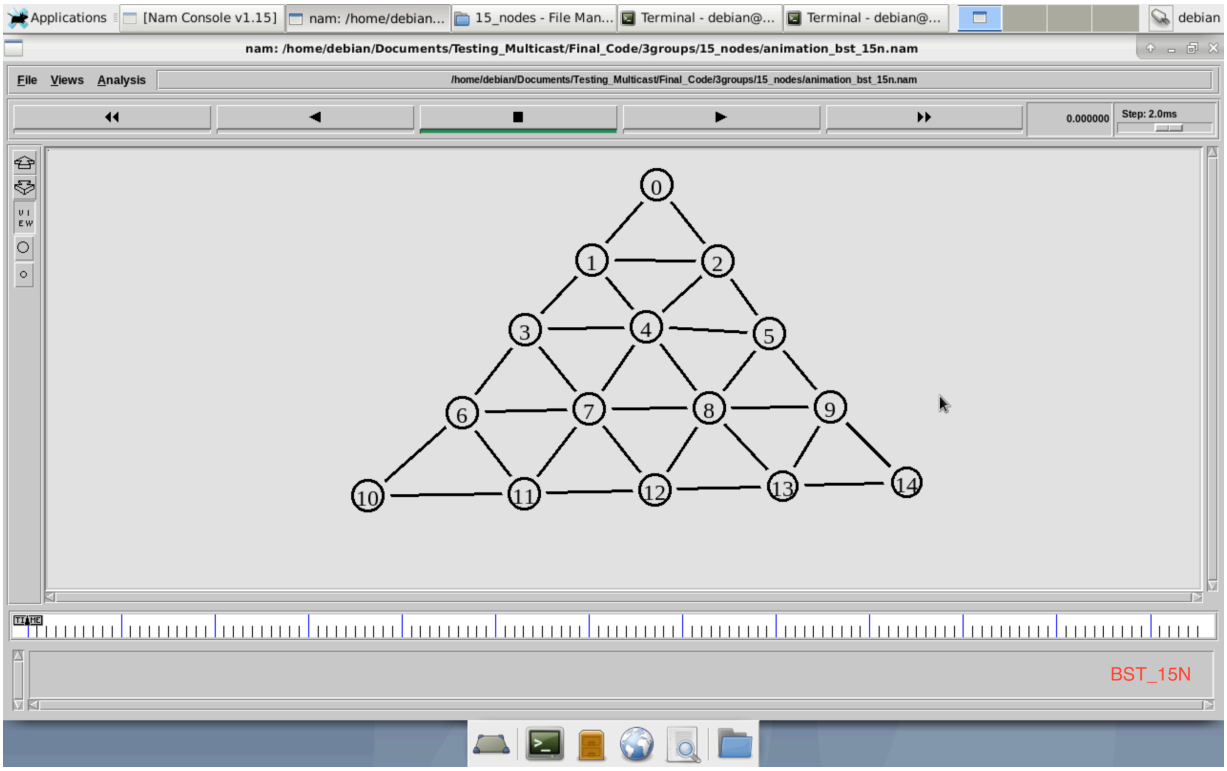


Fig 5.8: BST 15N_SS

All Scenario Codes

Link_to_Code

Simulation

Link_to_Simulations

Whole Data

Link_to_all

CHAPTER 6 - CONCLUSION

1. CBR vs Prune

DVMRP- For both the topolgy the cbr vs prune ratio is consistent

PIMDM - For both the topolgy the cbr vs prune ratio is consistent

CTR - For 10 nodes topology Prune flow exceeds CBR flow, therfore CTR is not good for low number of nodes topology

BST - For both the topologies, CBR flow exceed the prune flow by a huge margin.

2. Number of Packets

For 10 nodes - Transmitted packets and received packets are very close . And for CTR transmitted packets exceed the received packets, which is justifies by number of prune messages sent.

For 15 nodes - Transmitted packets are less than received packets.

This shows that the if the topology has more nodes, the need to send the packets is reduced by a huge margin, judging from the graph PIM-DM receives more packets.
DVMRP is not suitable for low number of nodes topology.

3.Packets Size

For 10 nodes - Only PIM-DM has the significant margin, whereas DVMRP has more Transmitted Size than Received Size.

For 15 nodes - BST has huge margin.

PIMDM is the best in both topology in terms of effective packets transfer. BST(15 nodes) is best in terms of less wastage of resource.

4.Packet Delay

1. CBT one of the two protocol supported by CTR has less packet delay compared to others and the packet delay is consistent owing to the fact that they have a RP.
2. .DVMRP has consistent delay.
3. BST has less delay in the starting and in the end, but delay is higher than DVMRP in the middle of the transmission time, which means the more the traffic the more the delay.
4. PIMDM has more delay than others

5.Bytes per sec

For 10 nodes - The graph is nearly consistent, except at the peak traffic where CTR and BST has more transmission rate.

PIMDM experiences significant drop in transmission rate

For 15 nodes - Same as 10 node but, CTR experiences significant drop in transmission rate.

This paper compares the multicasting protocols using NS2 simulation tool on the basis of different performance parameters as throughput, average end to end delay, number of packets send and cbr vs prune flow.

We draw the conclusion for that different protocols are suited for different toplogy one is suited for Dense topology while other is suited to more Sparse topology, and the each protocol has its own advantage with respect to the toplogy.

CHAPTER 7 - REFERNCES

1. Efficient Video Streaming Technique For Online Classes During Covid Pandemic
   Https://Www.Researchgate.Net/Publication/351423100_Efficient_Video_Streaming_Technique_For_Online_Classes_During_Covid_Pandemic#Fulltextfilecontent

2. The NS2 manual (formerly NS2 notes)
   https://www.isi.edu/nsnam/ns/doc/node338.html

3. Fault Tolerance Properties of Pyramid Networks
   https://ieeexplore.ieee.org/document/743415

4. Research on Simulation of multicast Protocol using OPNET Moduler
   https://ieeexplore.ieee.org/document/4722792

5. Configuring IP Multicast Routing
   https://www.cisco.com/en/US/docs/switches/lan/catalyst3850/software/release/3se/consolidated_guide/b_consolidated_3850_3se_cg_chapter_0101000.html#topic_046CCC6211D149DBAD2E9D7F22C7306D

6. Stanford Book on Multicasting Concepts
   https://web.stanford.edu/class/ee384a/files/Introduction_to_IP_Multicast_Routing.pdf

7. Content on Multicast Protocols
   https://www.cse.wustl.edu/~jain/cis788-97/ftp/ip_multicast/index.html

8. Prototypical Assessment of Multicasting Modes
https://www.researchgate.net/publication/263891343_Prototypical_Assessment_of_PIM-DM_DVMRP_CTR_and_BST_Multicasting_Modes

9. Network Lessons
https://networklessons.com/multicast/introduction-to-multicast

10. Multicast Routing
http://www2.ic.uff.br/~michael/kr1999/4-network/4_08-mcast.htm

11. Comparative Study of Multicasting Protocols Based on Average End-to-End Delay
https://ieeexplore.ieee.org/document/7914940

12. A comparison of the Internet multicast routing protocols
https://www.sciencedirect.com/science/article/pii/S0140366498002369