University of Alberta

Department of Computer Science
Faculty of Science

Impact of Network Errors on User Perception of Streaming Video

A Final Project/MINT 709 Submitted by

Johan Johan

In partial fulfilment of the requirement of the degree of Master of Science in Internetworking

Supervisor: Professor Mike MacGregor

December 2008

ACKNOWLEDGEMENT

I sincerely would like to thank my supervisor Professor Mike MacGregor for his advice during this project and throughout the study.

I would like to thank Associate Professor Paul Lu for his advice and help during MINT 709 project.

I am thankful to Jason Hidegh, Tseganesh Mehari, Mr Shanawaz Mir, and Parisa Zadeh for their advice and encouragements during this project.

I am thankful to Mr Mike Darling, my team lead at IBM, for the given flexible schedule during this project.

Finally, I would like to thank my family for the support and advice during my study at University of Alberta.

TABLE OF CONTENT

ABSTRACT

Streaming video has become popular to home users nowadays. Many households have an internet TV which is based on streaming video. In order to provide an excellent quality of streaming video to customers, the bandwidth and the efficiency/effectivity of a company's infrastructure needs to be performed. Before implementing it, the quality of streaming video needs to be tested. One way to measure it is through a user's perception on streaming video's quality. This is because there is not an industrialized-standard method to measure streaming video's quality. Or, even there is one; people's perception method is the one gets to decide.

CHAPTER ONE

1.1. Problem

With the growing development of internet, video distributed over internet has increased in demand. In order to provide an excellent quality of this service, testing the quality of the streaming video has become a company's top priorities. This certainly has brought attentions to many researchers focusing on internet multimedia applications.

One way of testing it is through a user's perception because viewers will have the sense whether or not the quality of delivered video streaming has met their expectation. In this experiment, a user is to evaluate the quality of video streaming through a multicast network against network errors such as packet loss, packet corruption, packet delay and packet duplication. Of the result of this experiment, a company can analyze the acceptable quality of video streaming and make a decision whether or not to improve their network infrastructure.

1.2. Project Goals

To gather data the user's perception of the acceptable range of errors on video streaming.

1.3. Limitation of scope

The following is the scope of the project.

- Multicast Network with PIM Sparse Mode
- Using VLC to stream video packets to the multicast network
- Using Netem to simulate network errors such as packet loss, packet corruption, packet duplication, and packet delay
- The collection of experimental data on actual user perceptions is outside the scope of

this project. So, the user perception on video streaming in this project is conducted by the author/writer of this project.

1.4. Research Methodology

- Researching associated theories such as multicast network along with its necessary protocols, network emulator, and VLC
- Laboratorium experiments including:
    - Design and build a multicast network with PIM Sparse Mode.
    - install, configure, and simulate network emulator
    - Configure and run VLC to send and received video packets across the multicast network.

1.5. Writing Systematization

Chapter 1 Introduction

This chapter explains problem, project's goal, limitation of scope, research methodology, and writing systematization.

Chapter 2 Supporting Theory

This chapter explains the associated theories relating to this project.

Chapter 3 Design and Implementation

This chapter explains the design of the multicast network and implementation of multicast network and network emulation.

Chapter 4 Data Analysis and Interpretation

This chapter discusses the data collection of user's perception on video streaming.

Chapter 5 Conclusion

This chapter concludes the result of data collection done in chapter 4.

CHAPTER TWO

2. 1. Multicast Network with PIM-SM [1]

IP Multicast is a technology to deliver a single stream of information to multiple/ thousands of users without adding burdens on the source and the receivers and only need the least network bandwidth.

IP Multicast addresses, which IP Class D, start from 224.0.0.0 to 239.255.255.255 and IP addresses from 224.0.0.0 to 224.0.0.255 are reserved. Globally scoped addresses are from 224.0.1.0 to 238.255.255.255.

The following is IP multicast to Ethernet/FDDI MAC Address mapping:

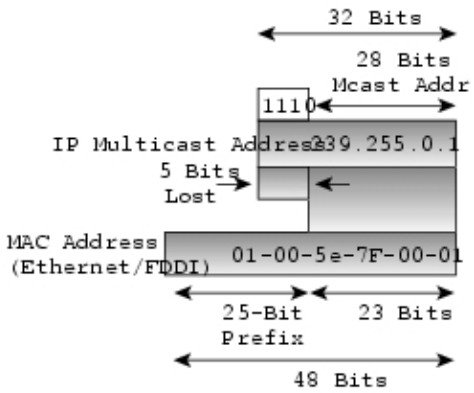

Figure 2.1. IP Multicast to Ethernet/FDDI MAC Address Mapping [1]

2.2. IGMP/Internet Group Management Protocol

In order a host to join a multicast group on a particular LAN, a host needs to send IGMP messages to its local multicast router. IGMP version 2 has four types of messages:

- Membership Query
- Version 1 membership report

- Version 2 membership report
- Leave group

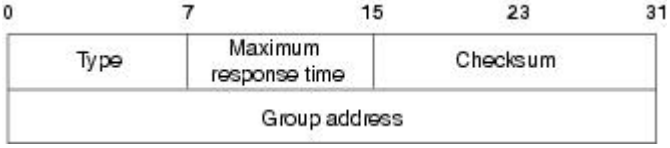

Figure 2.2. IGMPv2 Message Format[1]

If a host wants to join a multicast group, it needs to send an IGMP membership report to the specific multicast group. A router sends IGMP membership query to find out if there is a host still interested in the multicast group. If the router does not receive a reply after three times sending IGMP membership queries, the router times out and stops forwarding traffic.

2.3. PIM/Protocol Independent Multicast

PIM as per its name is a routing protocol-independent which uses unicast routing table produced by a unicast routing protocol to perform the multicast forwarding function. PIM uses the unicast routing table to perform the reverse path forwarding (RPF).

2.4. PIM Sparse-Mode

This type of PIM is used for this project. PIM Sparse-Mode is designed to be deployed for a large multicast network, of which some of the hosts, spread across a large network, interested in joining the multicast network. This is the opposite idea of PIM Dense-Mode, of which sends packets to hosts that might not be interested in joining the group.

In order to route packets from sources to receivers, there are three phases as per RFC 4601 [3]:

1. Phase one: RP(Rendezvous Point) Tree

   A host which is interested in joining a multicast group sends IGMP membership messages to the local router and forwarded to a designated router (DR). This happens after a local router

is elected as a designated router. DR sends PIM join message to RP for the multicast group. This join message is named as a (*, G) join, which means join group G for all sources to the group. The (*,G) join travels from one router to another one until reaching RP and each routers the messages passes through, multicast tree state for group G is created.

(*, G) join message either reaches RP or a router which has had (*, G) join state for the group. With other routers which have had (*, G) join state, this has established a distribution tree for group G rooted at RP/RP tree (known as a shared tree). Join message is periodically sent to RP to maintain the status of join. This can be seen on the following figure.

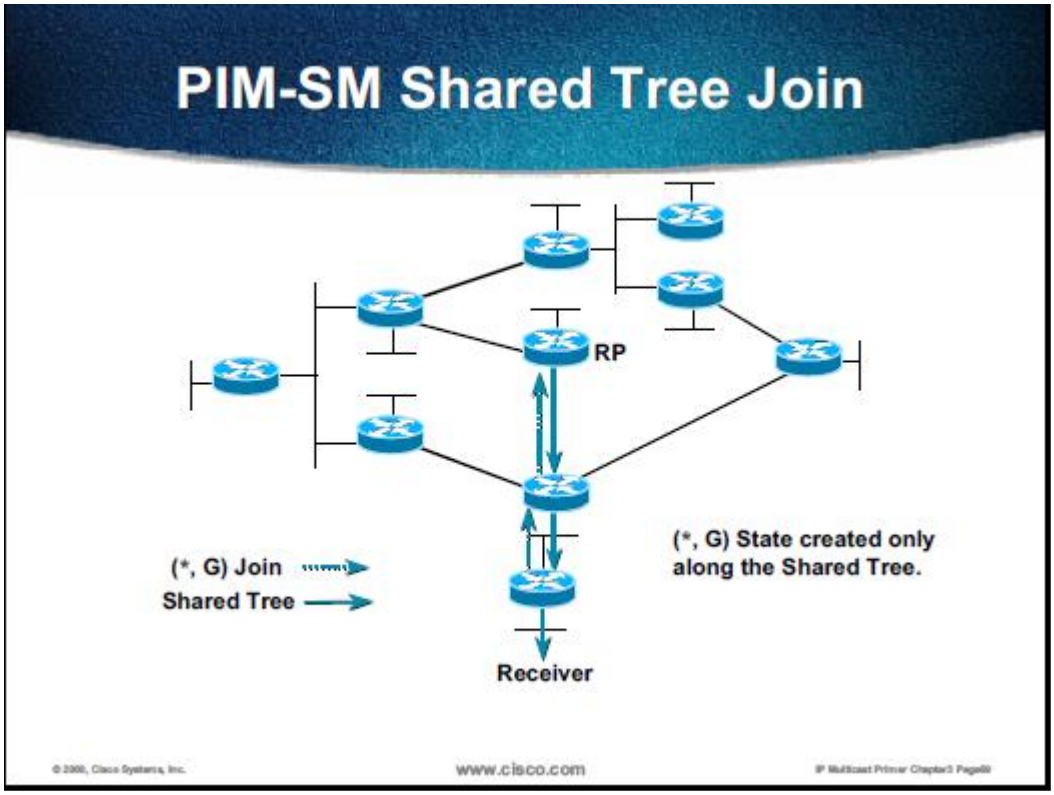

Figure 2.3. PIM-SM Shared Tree Join [2]

When a source sends packets to a multicast group, these packets are taken by a DR (sender's DR) and then encapsulated and then resent to RP. This encapsulation process is called registering and the encapsulation packets are called PIM Register packets. RP forwards these packets down the distribution tree/shared tree by following (*,G) multicast tree state(in

routers). These packets are replicated at RP tree branches and finally arrived at multicast receivers. This process can be illustrated as follows:

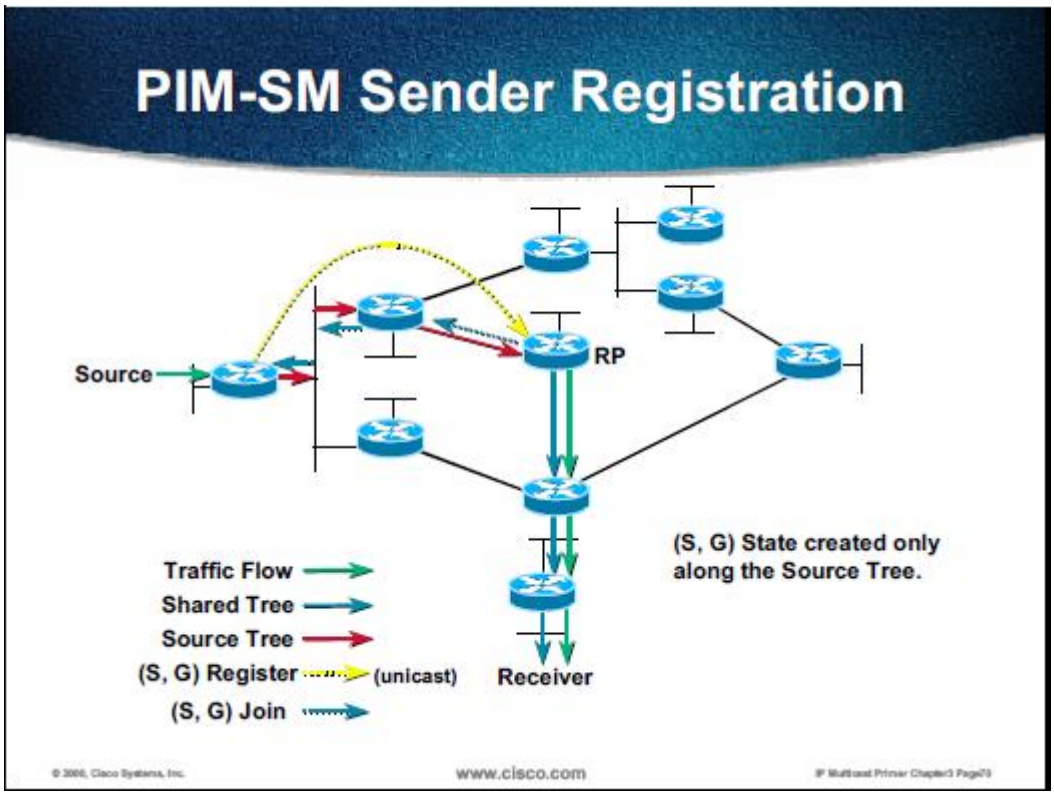

Figure 2.4. PIM-SM Sender Registration [2]

If receivers on a leaf –network want to leave the multicast network, this can be done by sending PIM(*,G) prune message to RP. If the PIM(*,G) prune message has not been received by RP, join state will eventually time out.

2. Phase Two: Register-Stop

Since register-encapsulation is likely to go on indefinitely and this is seemed inefficient, usually the RP will switch to native forwarding. To implement this method, at a time RP receives register-encapsulated packets, RP will initiate (S, G) state which is Source-Specific Join towards Source (S). This can be seen on the above picture. This join message propagates hop-by-hop towards S and instantiate (S, G) multicast state in the routers along the way to RP. The purpose of (S, G) multicast tree state is to forward packets to group G

from Source S.

If join message reaches source S's subnet or routers which have had (S, G) multicast tree state, data packets from source S starts to flow to RP by following multicast tree state (S, G) towards RP. If packets arrive at routers having (*, G) state along the way to RP, they can make a shortcut to the RP at this point.

When RP is n the process of joining source-specific tree for source S, packets are still encapsulated before arriving at RP. So, when packets begin to flow natively to the RP, the RP will receive two copies of the same packets. Therefore, the RP discards the encapsulated packets and sends a register-stop message back to source S's designated router. This process can be illustrated in the following figure.

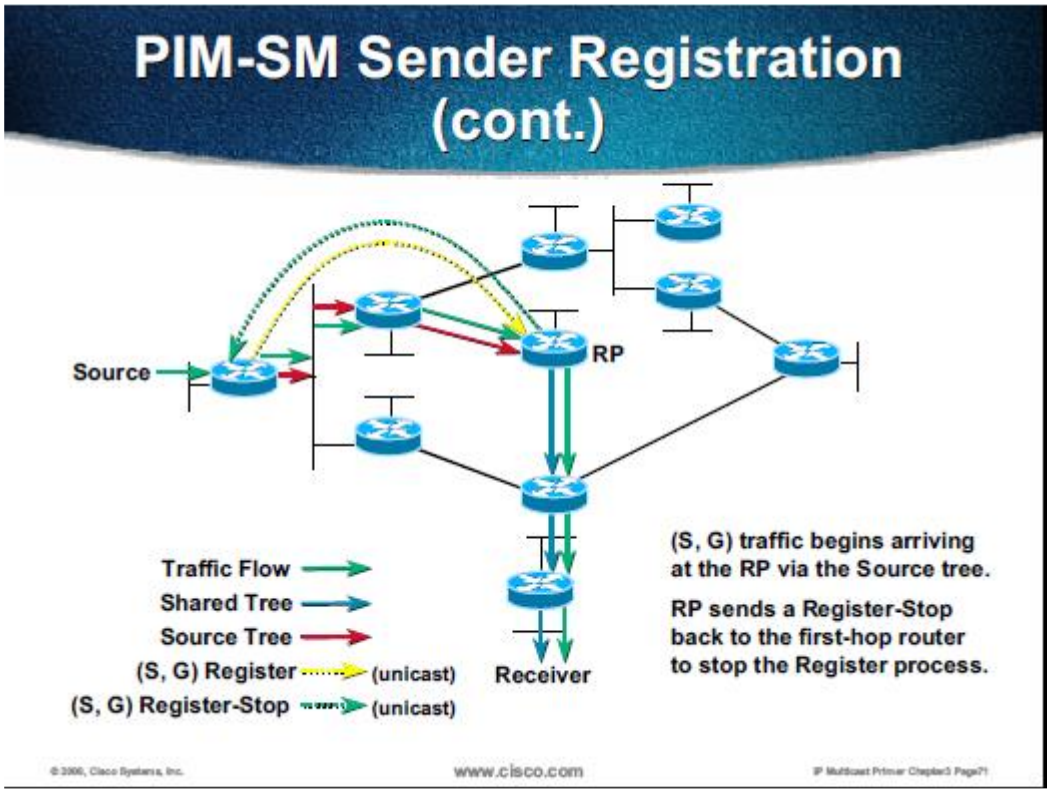

Figure 2.5. PIM-SM Sender Registration 2 [2]

So, now the traffic flows from source S to the RP and then go down the shared tree to the

receivers as illustrated in the following figure.

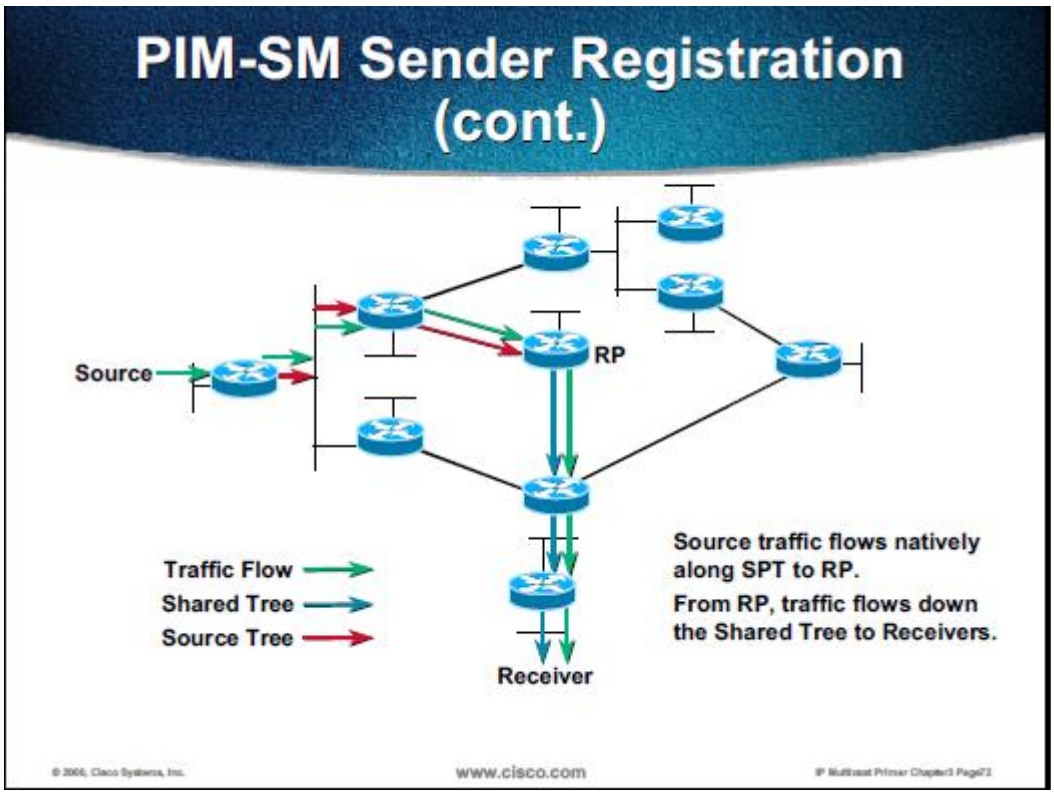

Figure 2.6. PIM-SM Sender Registration 3 [2]

3. Phase Three: Shortest-Path Tree

Even though RP has joined the source and data packets flows from the source to the RP and then to receivers, this may not be efficient or optimal compared to the shortest path from the source to the receiver. A router on receiver's LAN or DR initiates a direct transfer from the source shortest-path tree (SPT). In detail, DR on the receiver sends (S, G) join message towards the source and this state is created along the way to the source. Eventually, this join message either reaches the source or a router that has had (S, G) state. This process is illustrated as follows:

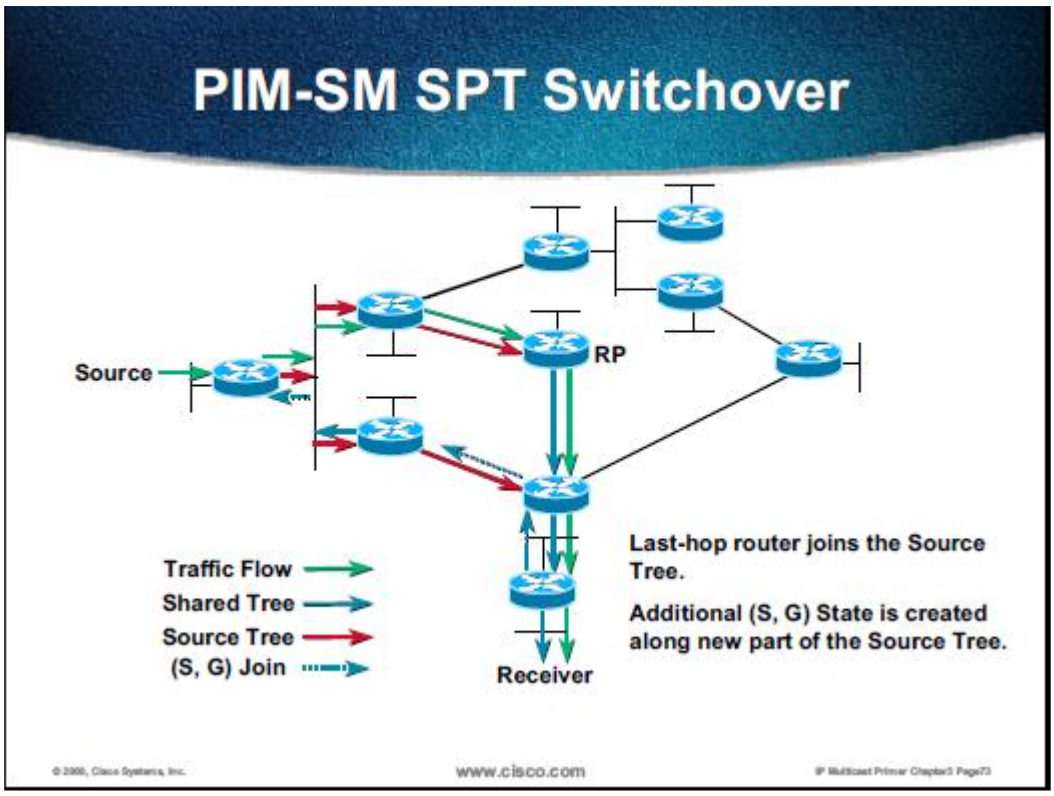

Figure 2.7. PIM-SM SPT Switchover [2]

Then, data packets flow from source S to receivers by following (S, G) state. At this time, the receiver will receive data packets from both SPT and RP tree. The receiver's DR then drops packets from RP tree and send an (S, G) prune message known as (S, G, rpt) prune to the RP. When this message travels towards the RP, state (S, G) is instantiated on any routers/hop in the way by signifying that data packets from RP tree shouldn't be forwarded to them anymore. This process is illustrated in the following figure.

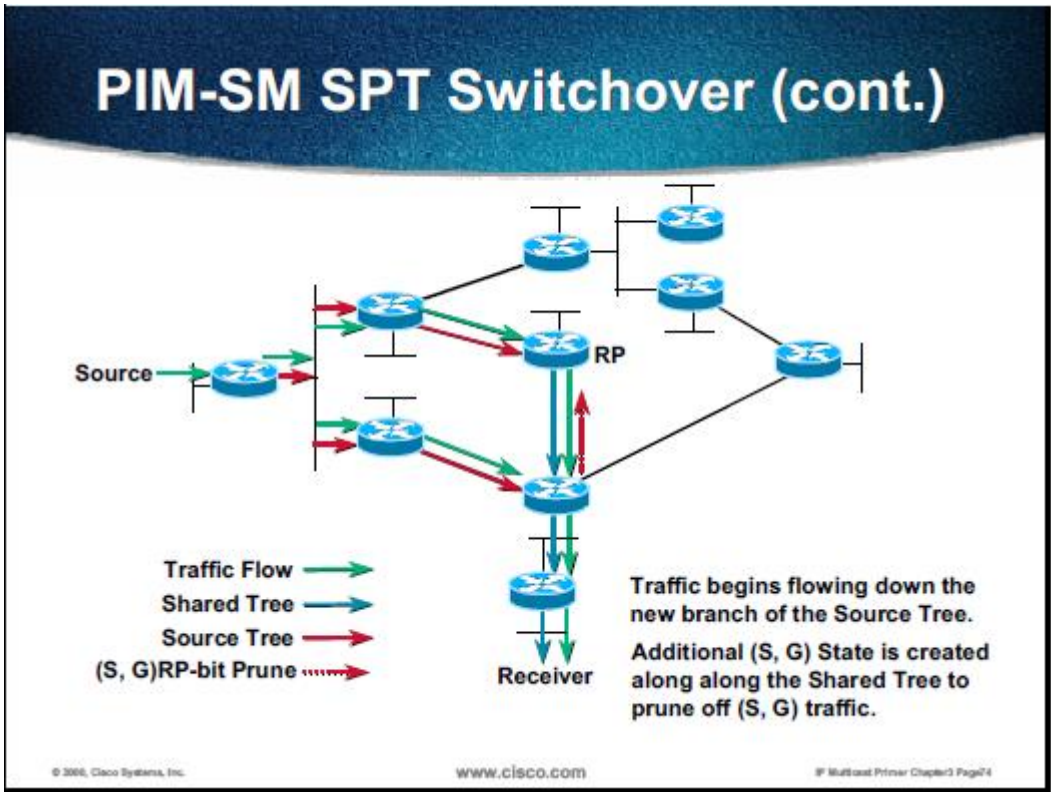

Figure 2.8. PIM-SM SPT Switchover 2 [2]

After (S, G) prune message reaches the RP through all the branches of the shared tree, the receiver only receives data packets from source S through shortest-path tree (SPT), but the source still sends data packets to RP. But, these data packets do not get forwarded down the shared tree.

2.5. MPEG

MPEG stands for Moving Picture Experts Group, which is an organization in charge of development of video and audio encoding standards. MPEG-2 is a standard released by the group which was implemented as an industrial encoding standard for DVD.

2.6. MPEG2- Transport [4]

This is a communication protocol for video, audio, and data as described in MPEG-2 Systems. This protocol is known as TS, TP, MPEG-TS, or M2T. Transport stream protocol has features

to multiplex digital video and audio and to synchronize the output. Transport stream can also carry a number of multiplexed individual programs.

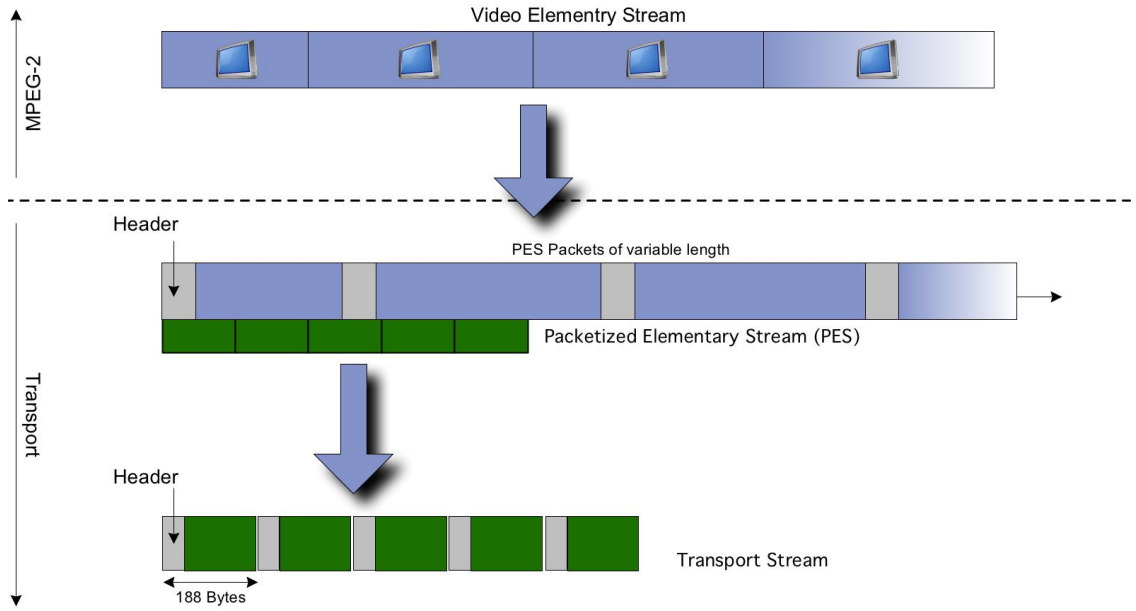

Figure 2.9. MPEG-2 Transport Standard [5]

As seen in the above picture, packetized elementary stream (PES) is made of video elementary stream. Video elementary stream is made of encoded video that follows MPEG-2 specification. And in order to stream video to a network, PES is broken down in many transport streams, and each one of which has 188 bytes in length.

Transport Packet Header (4 bytes)

1. Synchonization Byte

   TS packet starts with byte 0x47 and this code is used by a decoder to begin reading TS packets.

2. Transport Error Bit

   This bit is used to determine whether or not there is an error in the transport stream.

3. Payload Unit Start Indicatior (PUSI)

   This bit is to tell the decoder if there is PES somewhere in the payload.

4. Packet Identifier (PID)

   A decoder uses this ID to determine which component stream the packet belongs to.

5. Countinuity Counter (4 bits)

   This counter shows the position of the packet within the stream. The decoder uses this counter to arrange packets in their intended order.

Transport packet has 4 byte header which starts with a synchronization byte 0x47. The second byte consists of a Transport Error bit, a payload start bit, a transport priority bit, and 13-bit packet identifier.

2.7.1. ErrorMan [5]

ErrorMan is a tool developed by Jason Hidegh, a University of Alberta undergraduate student, to inject errors into mpeg2-ts packets. ErrorMan written in C++ uses pcap library, which is packet capture library. This library basically captures mpeg2-ts packets and inject errors into them.

Command: ErrorMan -d[device] -p[number] -f[UDP/RTP] -e[rate] -s[size] -r -l[logfile]

This command needs a root privileges to run or “permission denied” will be generated.

-d[device]	device used such as eth0, eth1, or en1 depending on the platform's device name.
-p[number]	Number represents how many TS packets to filter before stopping the capture. If number is 0, ErrorMan will capture packets until a “stop” command is issued.
-f[UDP/RTP]	Filter mpeg2-ts packets either encapsulated in UDP or RTP packets.
-e[rate]	Error rate ranges from 0 to 1
-l[logfile]	A file to log ErrorMan's activity
-r	If set, it enables packet forwarding/routing. ErrorMan will forward

	captured packets to the receiving hosts. If not set, ErrorMan is in test mode and will not transmit the stream.

Table 2.1. ErrorMan Commands Description

There are some dependencies that need to be met when installing ErrorMan. They are as follows:

1. Pcap library (ErrorMan was developed using version 0.9.5).
2. GNU C++ compiler and linker (version 4.0.1 or higher)
3. ncurses library
4. pthread library

2.7.2. Installation ErrorMan on Ubuntu

There are several problems at the time of ErrorMan installation on Ubuntu:

1. Since ErrorMan was developed and compiled with g++ version 4.0.1, compiling the program with the g++ 4.2.4, it will generate errors such as “SendPacket.h:39: error: extra qualification 'SendPacket::' on member 'display_segment'”. Therefore by removing the extra qualification: “SendPacket::”, compiling the program works.
2. After successful compilation, ErrorMan was tested for the first time by typing the command: ErrorMan -deth0 -p0 -frtp -r -e0.5 -s25 and then command: start. The program throws an error message “Segmentation Fault”. It was thought that the problem would be the incompatibility of the required ncurses library; however, when a new program from ubuntu distribution that uses ncurses library is installed, the new program behaves as expected. This proves that the installed ncurses library works. But, “Segmentation fault” problem is still not solved.
3. At the time of installing ErrorMan, pcap library version 0.9.8 was installed. ErrorMan still generated “Segmentation fault”. As per the developer's recommendation, pcap library version 0.9.8 was rolled back to version 0.9.5 (which was the one at the time of Errorman

development) and version 0.9.3. The result is that ErrorMan still generated “Segmentation fault” at the line of “device = pcap_lookupdev(errbuf);” of file: PacketSniffer.cpp. Variable device is supposed to be “eth0” or whichever the active interface a host has (configured in ErrorMan). “pcap_lookupdev()” is a library function of pcap library.

4. To prove that libpcap 0.9.5 or 0.9.8 works, the following code is place in the main class/function of ErrorMan program (ErrorMan.cpp):

```
char* active_device;
char errbuf[PCAP_ERRBUF_SIZE];
active_device = pcap_lookupdev(errbuf);
fprintf(stderr,"in ErrorMan active_device=%s\n",active_device);
fprintf(stderr,"in ErrorMan errbuf=%s\n",errbuf);
```

The result is that variable “active_device” is “eth0”, which is the active interface of the host(computer used in the development of this project) in rack1. This result proves that libpcap both version 0.9.8 and 0.9.5 behaves on Ubuntu.

5. As per the developer's suggestion, g++ version 4.2.4 was rolled back to version 4.0.1. The result suggests that g++ version 4.0.1 gives the same result as g++ version 4.2.4 does. But, g++ version 4.0.1 accepts extra qualification as mentioned in item 1.

The following is the error message acquired under Ubuntu with g++ 4.2.4(or g++4.0.1), pcap version 0.9.8(or pcap version 0.9.5):

```
[New Thread 0xb7ce56c0 (LWP 25476)]
Program received signal SIGSEGV, Segmentation fault.
[Switching to Thread 0xb7ce56c0 (LWP 25476)]
0x01d58c20 in ?? ()
(gdb) backtrace
#0      0x01d58c20 in ?? ()
#1      0xb7de23a7 in getifaddrs() from /lib/tls/i686/cmov/libc.so.6
#2      0x0806830b in pcap_findalldevs()
#3      0x0805a0dc in pcap_lookupdev()
#4      0x0804cab7 in PacketSniffer(this=0x80c2bd8, device=0x0, debug=false) at
```

```
PacketSniffer.cpp: 23
#5      0x0804fb44 in mpeg2_network(this=0x80c2bd8, c_info=0xbffcb600) at
mpeg2_network.cpp:77
#6      0x0804ba7f in consoleUI() at ErrorMan.cpp:295
#7      0x0804c86c in main(argc-6, argv=0xbffcb8b4) at ErrorMan.cpp:134
(gdb)
```

Figure 2.10. Debugging ErrorMan with GDB

Since there might be an interaction problem between errorman and pcap library and all debugging options have been exhausted, as per Associate Prof. Paul Lu's suggestion, a new operating system: Slackware was installed on a computer/host in MINT lab with the following detail:

- Slackware 12.1
- pcap version 0.9.8
- g++ version 4.2.4

The result is that ErrorMan can run without “segmentation fault”. After all the above errors were solved, ErrorMan were able to inject errors but, this does not make any changes on the screen of workstations/receivers. So, ErrorMan is not used later in the experiment.

2.8. Netem/Network Emulator [6]

Netem is an extention of TC, a trafic control tool in linux, which is in iproute2 package. Netem is used to emulate network properties by simulating packets dropping, packets corrupting, packets delaying, and packets duplication.

1. Packet Loss [7]

   command: tc qdisc add dev eth0 root netem loss 0.1%

   tc is traffic control.

qdisc is short for queuing discipline

eth0 is name of an interface at which packets leave.

0.1% is an example of loss' percentage.

To change the percentage of loss:

tc qdisc change dev eth0 root netem loss 1%.

And to remove it: tc qdisc del eth0 netem loss 1%.

2. Packet Corruption [7]

   Single bit of error is injected randomly into packet.

   command: tc qdisc add dev eth0 root netem corrupt 1%

3. Packet Delay [7]

   command: tc qdisc add dev eth0 root netem delay 100ms

   Since delay is not uniform in the real network, netem offers random delay and a variety of delay like a distribution normal.

   Example of the random delay:

   tc qdisc change dev eth0 root netem delay 100ms 50ms (the delay will be 100ms ± 50ms)

   tc qdisc change dev eth0 root netem delay 100ms 50ms 10% (the delay is 100ms ± 50ms and the next delay will depend on 10% of the last delay)

   Example of variation of delay in the form of a normal distribution:

   tc qdisc change dev eth0 root netem delay 100ms 30ms distribution normal.

4. Packet Duplication [7]

   command: tc qdisc add dev eth0 root netem duplicate 10%

CHAPTER THREE/ Design and Implementation

3.1. Network Topology

The implementation of multicast network is conducted in MINT lab. Routers used in this project are Cisco 2600 series, Cisco 2800 series, and Cisco 3600 series(Router A,B,C,D, and E). The switches are Cisco Catalyst 3750 series and Cisco Catalyst 3500 series switch. The video server is a PC running VLC, a software/program to stream video packets to the multicast network. The topology of the network is illustrated in Figure 3.1.

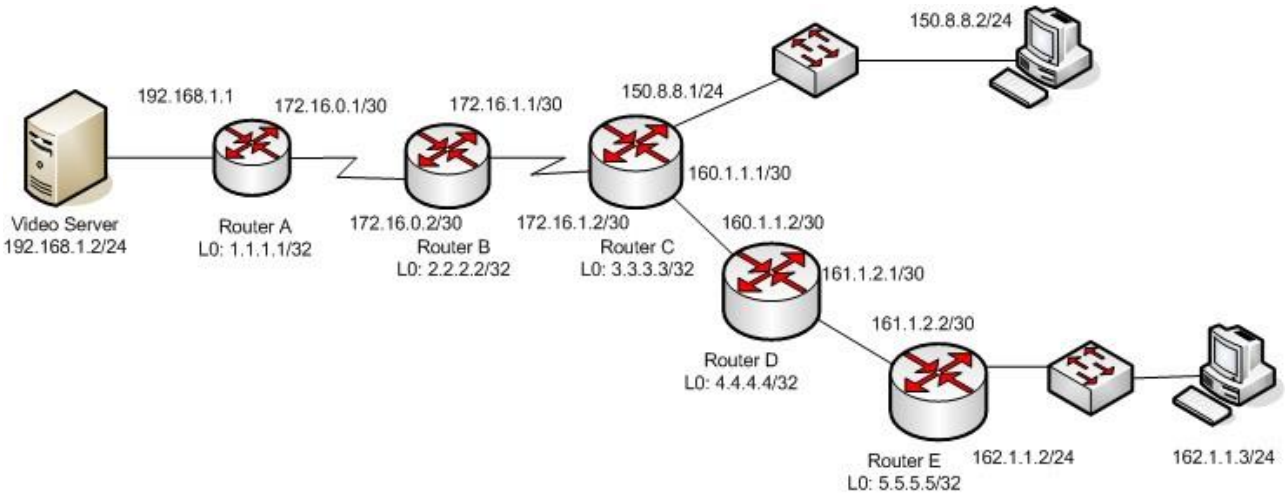

Figure 3.1. Network Topology

The two hosts are two PCs in MINT lab. These two hosts are to receive the video packets and run VLC to display them.

3.2. Routing Information

3.2.1. Routing Information for Senders or Receivers

Each source or destination of a multicast network has to have route: 224.0.0.0 netmask 240.0.0.0 in their routing table as seen as follows:

Kernel IP routing table						
Destination	*Gateway*	*Genmask*	*Flags*	*MSS*	*Window*	*irtt* *Iface*
150.8.8.0	*0.0.0.0*	*255.255.255.0*	*U*	*0 0*		*0 eth1*
224.0.0.0	***0.0.0.0***	***240.0.0.0***	***U***	***0 0***		***0 eth1***

Fig 3.2. Routing Table

This route enables a host to send or receive multicast packets and video packets distributed over a multicast network.

3.2.2. Routing information for Routers

Routing protocol used in this project is OSPF (Open Shortest Path First). OSPF is a link-state routing protocol in Interior Gateway Protocol. By flooding LSAs to all other routers within the same area, a designated router maintains its link-state database (LSDB). Once their LSDB is fully updated, a short-path tree for each route is calculated based on Dijkstra's algorithm.

A hello packet selects a Designated Router (DR) and a Backup Designated Router. A designated router allows a reduction in network traffic and in the size of network's topology. Multicast address used is 224.0.0.5, which is known as AllSPFRouters and 224.0.0.6, which is all Designated Router/AllDRouters[8].

Since this project focuses on the user perception of video streaming over a multicast network, OSPF is not detailed here. The following is the routing table of a host in the multicast network as seen in figure 3.1.

```
E#sh ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route

Gateway of last resort is not set

     1.0.0.0/32 is subnetted, 1 subnets
O E2    1.1.1.1 [110/20] via 161.1.2.1, 00:17:08, Ethernet1/0
     2.0.0.0/32 is subnetted, 1 subnets
O E2    2.2.2.2 [110/20] via 161.1.2.1, 00:17:08, Ethernet1/0
     3.0.0.0/32 is subnetted, 1 subnets
O       3.3.3.3 [110/12] via 161.1.2.1, 00:17:08, Ethernet1/0
     4.0.0.0/32 is subnetted, 1 subnets
O E2    4.4.4.4 [110/20] via 161.1.2.1, 00:17:08, Ethernet1/0
     5.0.0.0/32 is subnetted, 1 subnets
C       5.5.5.5 is directly connected, Loopback0
     172.16.0.0/30 is subnetted, 2 subnets
O IA    172.16.0.0 [110/139] via 161.1.2.1, 00:17:09, Ethernet1/0
O       172.16.1.0 [110/75] via 161.1.2.1, 00:17:09, Ethernet1/0
     162.1.0.0/24 is subnetted, 1 subnets
C       162.1.1.0 is directly connected, Ethernet3/0
     160.1.0.0/30 is subnetted, 1 subnets
O       160.1.1.0 [110/11] via 161.1.2.1, 00:17:10, Ethernet1/0
     161.1.0.0/30 is subnetted, 1 subnets
C       161.1.2.0 is directly connected, Ethernet1/0
O IA 192.168.1.0/24 [110/140] via 161.1.2.1, 00:11:40, Ethernet1/0
     150.8.0.0/24 is subnetted, 1 subnets
O       150.8.8.0 [110/12] via 161.1.2.1, 00:17:10, Ethernet1/0
```

Figure 3.3. Routing information on host E

```
C#sh ip route
Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, ia - IS-IS inter area
       * - candidate default, U - per-user static route, o - ODR
       P - periodic downloaded static route

Gateway of last resort is not set

     1.0.0.0/32 is subnetted, 1 subnets
O E2    1.1.1.1 [110/20] via 172.16.1.1, 00:16:18, Serial0/0
     2.0.0.0/32 is subnetted, 1 subnets
```

```
O E2    2.2.2.2 [110/20] via 172.16.1.1, 00:16:18, Serial0/0
     3.0.0.0/32 is subnetted, 1 subnets
C       3.3.3.3 is directly connected, Loopback0
     4.0.0.0/32 is subnetted, 1 subnets
O E2    4.4.4.4 [110/20] via 160.1.1.2, 00:16:18, FastEthernet0/0
     5.0.0.0/32 is subnetted, 1 subnets
O E2    5.5.5.5 [110/20] via 160.1.1.2, 00:16:18, FastEthernet0/0
     172.16.0.0/30 is subnetted, 2 subnets
O IA    172.16.0.0 [110/128] via 172.16.1.1, 00:16:19, Serial0/0
C       172.16.1.0 is directly connected, Serial0/0
     162.1.0.0/24 is subnetted, 1 subnets
O       162.1.1.0 [110/21] via 160.1.1.2, 00:16:20, FastEthernet0/0
     160.1.0.0/30 is subnetted, 1 subnets
C       160.1.1.0 is directly connected, FastEthernet0/0
     161.1.0.0/30 is subnetted, 1 subnets
O       161.1.2.0 [110/11] via 160.1.1.2, 00:16:20, FastEthernet0/0
O IA 192.168.1.0/24 [110/129] via 172.16.1.1, 00:10:50, Serial0/0
     150.8.0.0/24 is subnetted, 1 subnets
C       150.8.8.0 is directly connected, FastEthernet0/1
```

Figure 3.4. Routing information on host C

To see if each host can communicate, firstly all network addresses are seen in the routing table illustrated in Figure 3.3 and Figure 3.4. Secondly, All hosts ping one another. In this experiment, we try to ping host E to host A and vice versa (in Figure 3.5).

----------Ping from host E to host A----------	----------Ping from host E to host A----------
E#ping Protocol [ip]: Target IP address: 1.1.1.1 Repeat count [5]: Datagram size [100]: Timeout in seconds [2]: Extended commands [n]: Sweep range of sizes [n]: Type escape sequence to abort. Sending 5, 100-byte ICMP Echos to 1.1.1.1, timeout is 2 seconds: !!!!! Success rate is 100 percent (5/5), round-trip min/avg/max = 1/2/4 ms E#	A#ping Protocol [ip]: Target IP address: 5.5.5.5 Repeat count [5]: Datagram size [100]: Timeout in seconds [2]: Extended commands [n]: Sweep range of sizes [n]: Type escape sequence to abort. Sending 5, 100-byte ICMP Echos to 5.5.5.5, timeout is 2 seconds: !!!!! Success rate is 100 percent (5/5), round-trip min/avg/max = 1/2/4 ms A#

Fig 3.5. Hosts Ping-ing

3.3. Multicast Route

3.3.1. Reverse Path Forwarding/RPF [9]

RPF is used to make sure that multicast packets arrive at the correct incoming interface and to make sure of loop-free forwarding of multicast packets. When multicast packets arrive at an interface of a router, the router only accepts them if that is the interface that the router would send unicast datagrams to the source of multicast network. If it is not correct, RPF fails. The following is the "show rpf "of the experiment starting from Router C.

```
------------------------sh ip rpf --------------------------
C#sh ip rpf 192.168.1.1
RPF information for ? (192.168.1.1)
  RPF interface: Serial0/0
  RPF neighbor: ? (172.16.1.1)
  RPF route/mask: 192.168.1.0/24
  RPF type: unicast (ospf 100)
  RPF recursion count: 0
  Doing distance-preferred lookups across tables
C#

B#sh ip rpf 192.168.1.2
RPF information for ? (192.168.1.2)
  RPF interface: Serial0/0
  RPF neighbor: ? (172.16.0.1)
  RPF route/mask: 192.168.1.0/24
  RPF type: unicast (ospf 100)
  RPF recursion count: 0
  Doing distance-preferred lookups across tables
B#
A#sh ip rpf 192.168.1.2
RPF information for ? (192.168.1.2)
  RPF interface: FastEthernet0/0
  RPF neighbor: ? (0.0.0.0) - directly connected
  RPF route/mask: 192.168.1.0/24
  RPF type: unicast (connected)
  RPF recursion count: 0
  Doing distance-preferred lookups across tables
A#
```

Figure 3.6. RPF

Figure. 3.6 shows that the incoming interface of Router C for the multicast network is Serial0/0, which is directly connected to interface Serial0/1 of Router B with IP address 172.16.1.1. Then,

"sh rpf" on Router B shows that the interface for the incoming multicast packets is Serial0/0 which is directly connected to Serial0/0 of Router A with IP address 172.16.0.1. At Router A, the incoming interface is FastEthernet0/0 and it is directly connected to the source of multicast network with IP Address 192.168.1.2.

3.3.2. Multicast Routing Information

The following is the "show ip mroute summary" on Router C.

```
C#sh ip mroute summary
IP Multicast Routing Table
Flags: D - Dense, S - Sparse, B - Bidir Group, s - SSM Group, C - Connected,
       L - Local, P - Pruned, R - RP-bit set, F - Register flag,
       T - SPT-bit set, J - Join SPT, M - MSDP created entry,
       X - Proxy Join Timer Running, A - Candidate for MSDP Advertisement,
       U - URD, I - Received Source Specific Host Report, Z - Multicast Tunnel
       Y - Joined MDT-data group, y - Sending to MDT-data group
Outgoing interface flags: H - Hardware switched
 Timers: Uptime/Expires
 Interface state: Interface, Next-Hop or VCD, State/Mode

(*, 239.255.0.1), 00:33:33/stopped, RP 3.3.3.3, OIF count: 4, flags: SJCL
  (192.168.1.2, 239.255.0.1), 00:23:00/00:03:29, OIF count: 3, flags: LT

(*, 224.0.1.40), 00:33:33/00:03:10, RP 3.3.3.3, OIF count: 3, flags: SJCL

(*, 224.2.127.254), 00:26:00/00:03:13, RP 3.3.3.3, OIF count: 2, flags: SJC

(*, 239.255.255.255), 00:26:00/stopped, RP 3.3.3.3, OIF count: 2, flags: SJC
  (192.168.1.2, 239.255.255.255), 00:22:58/00:03:25, OIF count: 2, flags: T

(*, 239.195.255.255), 00:26:01/00:03:11, RP 3.3.3.3, OIF count: 2, flags: SJC

C#
```

Figure 3.7. Multicast Route Information

In shared trees, "*" means any source since the source is not specifically noted. And the root of shared trees is Rendevouz Point (RP) [11].

239.195.255.255, 239.255.255.255, and 224.2.127.254 are SAP IP Addresses. SAP stands for Session Announcement Protocol which is used to distribute multicast sessions and information to prospective multicast clients. 224.2.127.254 is a global-scope SAP IP address [12]. 239.195.255.255 and 239.255.255.255 are SAP IP addresses for organization scope and local scope respectively [13]. Therefore, (192.168.1.2, 239.255.255.255) means that the source of the local scope SAP announcement is 192.168.1.2, which is VLC server/the source.

In PIM-SM static rp configuration, 224.0.1.40 is a group service in the shared tree in order to communicate with the RP [14].

(192.168.1.2, 239.255.0.1) means that the receiver's PIM-SM Designated Router (DR) has the information of the active source. And since the RP is in this router, it shows that RP has the information of the active source as well.

3.4. VLC media player

VLC media player is a free cross-platform multimedia player released under the GNU General Public License. VLC is used in this project to stream video packets to a multicast network.

VLC's command [10] to stream a movie to a multicast network:

vlc /home/movies/movie.mpg --ttl 16 --sout

'#duplicate{dst=std{access=rtp,mux=ts,dst=239.255.0.1:1234,sap,name="testing"},dst=display}'

/home/movies/movie.mpg is a sample movie.

access=rtp means that RTP is used to stream packets.

mux=ts means that streams' s encapsulation method, which is MPEG2-TS muxer.

dst is the destination.

CHAPTER FOUR / Data Analysis and Interpretation

4. Data Analysis and Interpretation

In this project, user's perception on video streaming is conducted by the author (me). I rate the quality of video streaming from number 5, which is good quality (normal) to number 1, which is the worst.

4.1. Packet Loss

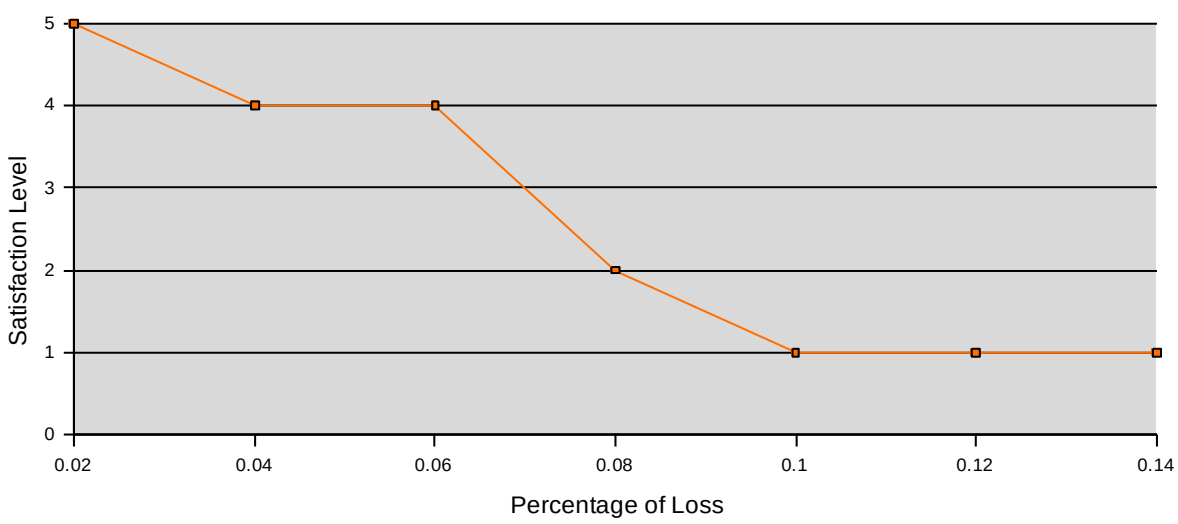

Figure 4.1. User's Perception on Video Streaming Based on Packet Loss

The errors become apparent when the percentage of packet loss is greater than 0.06%. When the percentage of packet loss is 0.1% or more, the video becomes unacceptable to viewers.

4.2. Packet Corruption

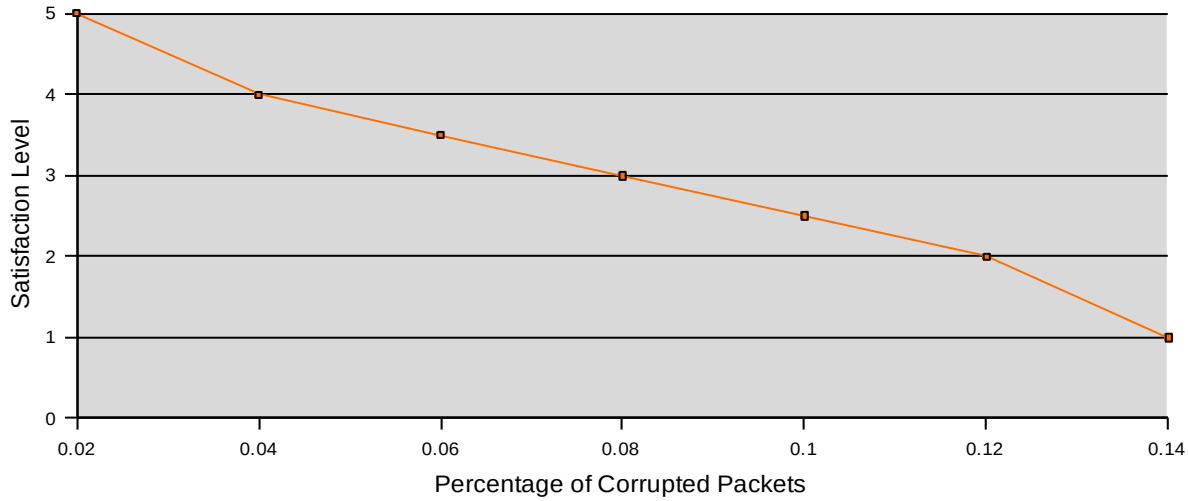

Figure 4.2. User's Perception on Video Streaming Based on Corrupted Packets

In this scheme, errors become annoying when the percentage of corrupted packets increases from 0.1%.

4.3. Packet Duplication

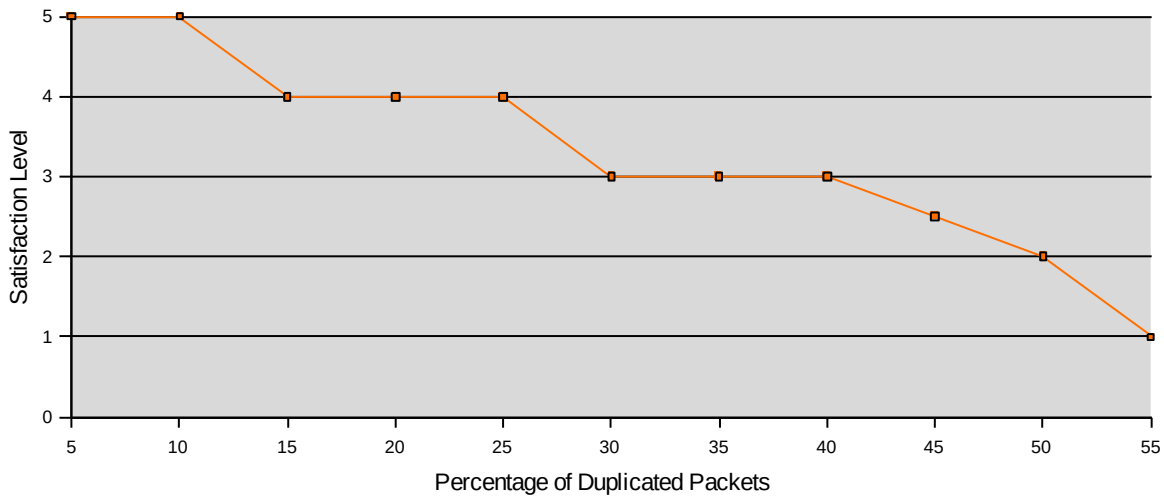

Figure 4.3. User's Perception on Video Streaming Based on Duplicated Packets with one movie

The effect of duplicated packets on video streaming grows bad when the percentage of which is increased over 45%;however, the impact of duplicated packets on video streaming becomes

much worse when there are more than one movie being streamed to a network. Of the experiment, two movies are streamed to the same multicast network: 239.255.0.1. The result is that the quality of video streaming becomes very bad if the percentage of duplicated packets increases over 5% as illustrated in the chart below.

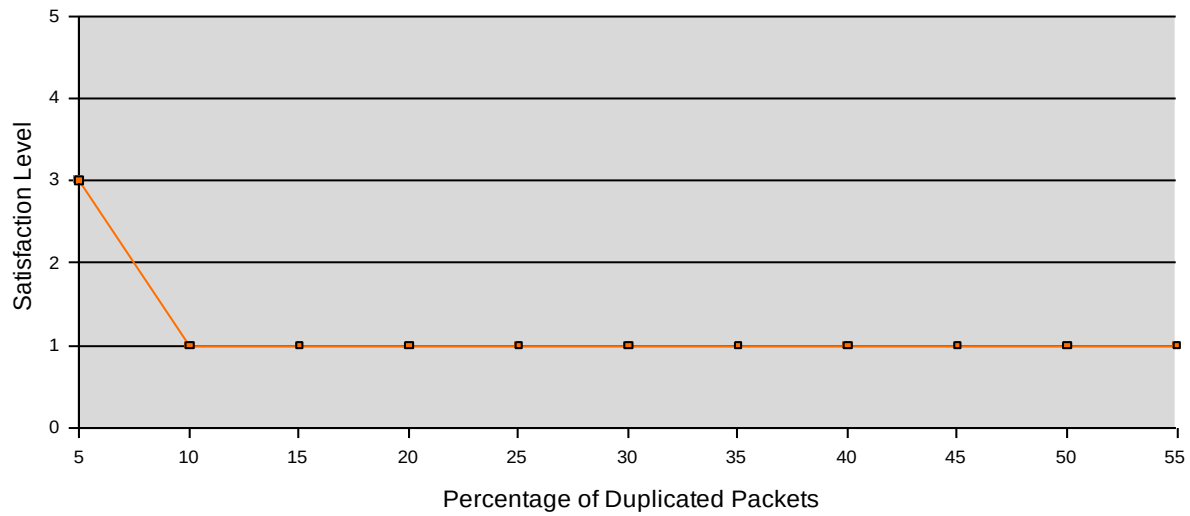

Figure 4.4. User's Perception on Video Streaming Based on Duplicated Packets with two movies

4.4. Packet Delay

In this experiment, the delay is started from 10 ms all the way up to 5 seconds. Delay gives impact to the quality of video streaming if the delay is over 2 seconds. This is because VLC has cache/buffer that is good enough to accommodate delayed packets (default cache is 300 ms). At 3 second delay, the video starts to blur.

CHAPTER FIVE / Conclusion

By analyzing the result of user's perception of network errors on video streaming as collected in chapter four, I can take the conclusion as follows:

- In packet duplication experiment, the number of videos streamed to the network affects user's perception of the quality of video streaming.
- The result of packet corruption experiment is slightly better than the one of packet loss experiment. This might be caused by that the small number of the random injected errors in packet corruption scheme might not cause the quality of the video streaming worse while the number of loss packets will not be recovered and this impacts the viewers' perception.
- VLC's cache may play an important role in user's perception of the impact of delay on video streaming.

REFERENCES

1. Cisco Systems, Inc.. “Internet Protocol (IP) Multicast”, <http://www.cisco.com/en/US/docs/internetworking/technology/handbook/IP-Multi.html>

2. Cisco System, Inc.. *Implementing Cisco Multicast (MCAST) v1.0*. 2003

3. Fenner, B., Handley, M., Holbrook, H., and I. Kouvelas, “Protocol Independent Multicast – Sparse Mode (PIM-SM): Protocol Specification (Revised)”, RFC 4601, August 2006.

4. “MPEG Transport Stream.” *Wikipedia*.1 May 2008.<http://en.wikipedia.org/wiki/MPEG_transport_stream>.

5. Hidegh, Jason. *Media Transport Technical Summary*, CMPUT 496, 2007

6. Keller, Ariane. *Manual TC Packet Filtering and Netem*, ETH Zurich July 20,2006

7. “Netem.” LinuxFoundation.com:Netem”. 1 June 2008<http://www.linuxfoundation.org/en/Net:Netem.>

8. “Open Shortest Path First.” *Wikipedia*.10 July 2008.<http://en.wikipedia.org/wiki/Open_Shortest_Path_First>.

9. “Reverse Path Forwarding.” Juniper.net: Reverse Path Forwarding. 10 August 2008 <http://www.juniper.net/techpubs/software/erx/erx50x/swconfig-routing-vol1/html/ip-multicast-config7.html.>

10. “Transcoding and multiple streaming.” Videolan.org. 10 June 2008. <http://www.videolan.org/doc/streaming-howto/en/ch04.html#id311538>

11. Solie, Karl.,Lynch, Leah. “CCIE Practical Studies, Volume II”. Cisco Press. 7 November 2003. Pearson Education. 8 January 2009. <http://www.informit.com/library/content.aspx?b=CCIE_Practical_Studies_II&seqNum=28>.

12. Rahman, Syed Mahbubur, "Multimedia Networking: Technology, Management, and Applications". Idea Group Inc (IGI), 2002.

13. “Scope Relative Multicast Addresses”.7 January 2009. <http://www.multicast.org.uk/address-tools/scope-relative.html>.

14. “CCIE, the beginning:Multicast over FR NBMA part 2 – PIM NBMA mode and Static RP”.31 July 2008. <http://cciethebeginning.wordpress.com/2008/07/31/multicast-over-fr-nbma-part2-%E2%80%93-pim-nbma-mode-and-static-rp/>.

APPENDIX-A

Routers and Switches Configuration

```
A#sh run
Building configuration...

Current configuration : 1214 bytes
!
version 12.2
service timestamps debug datetime msec
service timestamps log datetime msec
no service password-encryption
!
hostname A
!
logging queue-limit 100
!
ip subnet-zero
!
!
no ip domain lookup
!
ip multicast-routing
mpls ldp logging neighbor-changes
!
!
!
!
!
!
!
!
!
no voice hpi capture buffer
no voice hpi capture destination
!
!
mta receive maximum-recipients 0
!
!
!
!
interface Loopback0
 ip address 1.1.1.1 255.255.255.255
 ip pim sparse-mode
!
interface FastEthernet0/0
 ip address 192.168.1.1 255.255.255.0
 ip pim sparse-mode
 duplex auto
 speed auto
!
interface Serial0/0
 ip address 172.16.0.1 255.255.255.252
 ip pim sparse-mode
 clockrate 8000000
 no fair-queue
!
interface Serial0/1
```

```
 no ip address
 shutdown
!
router ospf 100
 router-id 1.1.1.1
 log-adjacency-changes
 redistribute connected subnets
 network 172.16.0.0 0.0.0.3 area 0
 network 192.168.1.0 0.0.0.255 area 1
!
ip http server
ip classless
!
ip pim rp-address 3.3.3.3
!
!
!
!
call rsvp-sync
!
voice-port 1/0/0
!
voice-port 1/0/1
!
!
mgcp profile default
!
dial-peer cor custom
!
!
!
!
line con 0
line aux 0
line vty 0 4
!
!
end

A#
------------------------------------------------------------
B#sh run
Building configuration...

Current configuration : 1233 bytes
!
version 12.2
service timestamps debug datetime msec
service timestamps log datetime msec
no service password-encryption
!
hostname B
!
logging queue-limit 100
!
memory-size iomem 10
ip subnet-zero
!
!
no ip domain lookup
!
ip multicast-routing
```

```
mpls ldp logging neighbor-changes
!
!
!
!
!
!
!
!
!
no voice hpi capture buffer
no voice hpi capture destination
!
!
mta receive maximum-recipients 0
!
!
!
!
interface Loopback0
 ip address 2.2.2.2 255.255.255.255
 ip pim sparse-mode
!
interface FastEthernet0/0
 no ip address
 shutdown
 duplex auto
 speed auto
!
interface Serial0/0
 ip address 172.16.0.2 255.255.255.252
 ip pim sparse-mode
 no fair-queue
!
interface Serial0/1
 ip address 172.16.1.1 255.255.255.252
 ip pim sparse-mode
 clockrate 8000000
!
router ospf 100
 router-id 2.2.2.2
 log-adjacency-changes
 redistribute connected subnets
 network 172.16.0.0 0.0.0.3 area 0
 network 172.16.1.0 0.0.0.3 area 2
!
ip http server
ip classless
!
ip pim rp-address 3.3.3.3
!
!
!
!
call rsvp-sync
!
voice-port 1/0/0
!
voice-port 1/0/1
!
!
mgcp profile default
```

```
!
dial-peer cor custom
!
!
!
!
line con 0
line aux 0
line vty 0 4
!
!
end

B#
-----------------------------------------------------

C#sh run
Building configuration...

Current configuration : 1389 bytes
!
version 12.2
service timestamps debug datetime msec
service timestamps log datetime msec
no service password-encryption
!
hostname C
!
logging queue-limit 100
!
memory-size iomem 10
ip subnet-zero
!
!
no ip domain lookup
!
ip multicast-routing
mpls ldp logging neighbor-changes
!
!
!
!
!
!
!
!
!
no voice hpi capture buffer
no voice hpi capture destination
!
!
mta receive maximum-recipients 0
!
!
!
!
interface Loopback0
 ip address 3.3.3.3 255.255.255.255
 ip pim sparse-mode
!
interface FastEthernet0/0
 ip address 160.1.1.1 255.255.255.252
```

```
 ip pim sparse-mode
 duplex auto
 speed auto
!
interface Serial0/0
 ip address 172.16.1.2 255.255.255.252
 ip pim sparse-mode
 no fair-queue
!
interface FastEthernet0/1
 ip address 150.8.8.1 255.255.255.0
 ip pim sparse-mode
 duplex auto
 speed auto
!
interface Serial0/1
 no ip address
 shutdown
!
router ospf 100
 router-id 3.3.3.3
 log-adjacency-changes
 network 3.3.3.3 0.0.0.0 area 2
 network 150.8.8.0 0.0.0.255 area 2
 network 160.1.1.0 0.0.0.3 area 2
 network 172.16.1.0 0.0.0.3 area 2
!
ip http server
ip classless
!
ip pim rp-address 3.3.3.3
!
!
!
!
call rsvp-sync
!
voice-port 1/0/0
!
voice-port 1/0/1
!
!
mgcp profile default
!
dial-peer cor custom
!
!
!
!
line con 0
line aux 0
line vty 0 4
!
!
end

C#
------------------------------------------------------------

D#sh run
Building configuration...
```

```
Current configuration : 1299 bytes
!
version 12.4
service timestamps debug datetime msec
service timestamps log datetime msec
no service password-encryption
!
hostname D
!
boot-start-marker
boot-end-marker
!
!
no aaa new-model
!
resource policy
!
memory-size iomem 10
!
!
ip cef
!
!
no ip domain lookup
ip multicast-routing
!
!
voice-card 0
 no dspfarm
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
interface Loopback0
 ip address 4.4.4.4 255.255.255.255
!
interface GigabitEthernet0/0
 ip address 160.1.1.2 255.255.255.252
 ip pim sparse-mode
 duplex auto
 speed auto
!
interface GigabitEthernet0/1
 ip address 161.1.2.1 255.255.255.252
```

```
 ip pim sparse-mode
 duplex auto
 speed auto
!
interface FastEthernet0/1/0
!
interface FastEthernet0/1/1
!
interface FastEthernet0/1/2
!
interface FastEthernet0/1/3
!
interface Serial0/0/0
 no ip address
 shutdown
 no fair-queue
 clock rate 2000000
!
interface Vlan1
 no ip address
!
router ospf 100
 router-id 4.4.4.4
 log-adjacency-changes
 redistribute connected subnets
 network 160.1.1.0 0.0.0.3 area 2
 network 161.1.2.0 0.0.0.3 area 2
!
!
ip http server
no ip http secure-server
ip pim rp-address 3.3.3.3
!
!
!
!
control-plane
!
!
!
!
!
!
!
!
!
line con 0
line aux 0
line vty 0 4
 login
!
scheduler allocate 20000 1000
!
end

D#
-----------------------------------------------------------
E#sh run
Building configuration...

Current configuration : 1288 bytes
!
```

```
version 12.2
service timestamps debug uptime
service timestamps log uptime
no service password-encryption
!
hostname E
!
!
ip subnet-zero
!
!
no ip domain-lookup
!
ip multicast-routing
call rsvp-sync
!
!
!
!
!
!
!
!
interface Loopback0
 ip address 5.5.5.5 255.255.255.255
!
interface Serial0/0
 no ip address
 shutdown
 no fair-queue
!
interface Serial0/1
 no ip address
 shutdown
!
interface Serial0/2
 no ip address
 shutdown
!
interface Serial0/3
 no ip address
 shutdown
!
interface Serial0/4
 no ip address
 shutdown
!
interface Serial0/5
 no ip address
 shutdown
!
interface Serial0/6
 no ip address
 shutdown
!
interface Serial0/7
 no ip address
 shutdown
!
interface Ethernet1/0
 ip address 161.1.2.2 255.255.255.252
 ip pim sparse-mode
```

```
 half-duplex
!
interface ATM2/0
 no ip address
 shutdown
 no atm ilmi-keepalive
!
interface Ethernet3/0
 ip address 162.1.1.2 255.255.255.0
 ip pim sparse-mode
 ip cgmp
 half-duplex
!
router ospf 100
 router-id 5.5.5.5
 log-adjacency-changes
 redistribute connected subnets
 network 161.1.2.0 0.0.0.3 area 2
 network 162.1.1.0 0.0.0.255 area 2
!
ip classless
ip http server
ip pim rp-address 3.3.3.3
!
!
!
dial-peer cor custom
!
!
!
!
line con 0
line aux 0
line vty 0 4
!
end

E#


------------------------------------------------------------
Switch3750#sh run
Building configuration...

Current configuration : 1587 bytes
!
version 12.2
no service pad
service timestamps debug uptime
service timestamps log uptime
no service password-encryption
!
hostname Switch
!
!
no aaa new-model
switch 1 provision ws-c3750g-24ps
ip subnet-zero
no ip domain-lookup
!
ip igmp snooping vlan 1 mrouter learn cgmp
ip igmp snooping vlan 1 mrouter interface Gi1/0/11
```

```
ip igmp snooping vlan 1 static 239.255.0.1 interface Gi1/0/5
!
!
!
no file verify auto
spanning-tree mode pvst
spanning-tree extend system-id
!
vlan internal allocation policy ascending
!
!
interface GigabitEthernet1/0/1
!
interface GigabitEthernet1/0/2
!
interface GigabitEthernet1/0/3
!
interface GigabitEthernet1/0/4
!
interface GigabitEthernet1/0/5
!
interface GigabitEthernet1/0/6
!
interface GigabitEthernet1/0/7
!
interface GigabitEthernet1/0/8
!
interface GigabitEthernet1/0/9
!
interface GigabitEthernet1/0/10
!
interface GigabitEthernet1/0/11
!
interface GigabitEthernet1/0/12
!
interface GigabitEthernet1/0/13
!
interface GigabitEthernet1/0/14
!
interface GigabitEthernet1/0/15
!
interface GigabitEthernet1/0/16
!
interface GigabitEthernet1/0/17
!
interface GigabitEthernet1/0/18
!
interface GigabitEthernet1/0/19
!
interface GigabitEthernet1/0/20
!
interface GigabitEthernet1/0/21
!
interface GigabitEthernet1/0/22
!
interface GigabitEthernet1/0/23
!
interface GigabitEthernet1/0/24
!
interface GigabitEthernet1/0/25
!
interface GigabitEthernet1/0/26
```

```
!
interface GigabitEthernet1/0/27
!
interface GigabitEthernet1/0/28
!
interface Vlan1
 no ip address
 shutdown
!
ip classless
ip http server
!
!
!
control-plane
!
!
line con 0
line vty 5 15
!
end

Switch3750#
```