CHAPTER 6

IMPLEMENTATION OF TRACE ANALYZER FOR NS3

This chapter presents a framework for NS3 Trace Analyzer which is implemented in java. NS3 trace analyzer reads the .tr file and automatically calculates network performance parameters like throughput, packet delivery ratio, packet drop, packet sent, packet received. It makes post-processing of .tr file very easy. This NS3 Trace Analyzer was developed with the intention that the researchers can solely focus on design and development of new protocol than wasting much of his/her time in analyzing the .tr file.

6.1 INTRODUCTION

NS3 is an open source free software freely and publicly available for research and development. It is licensed under GNU GPLv2 license. The main aim of this NS3 project is to give network researchers the open simulation environment to test their network protocols so that these protocols give same performance in the real time screnario.NS3 is developed by Tom Henderson, George, Sally Floyd and Sumit Roy. The team received funding from US National Science Foundation (NSF) for the development of NS3.NS3 is written from the scratch in c++ programming language. Waf build systems and framework for generating Python bindings are contributed by Gustavo Carneiro.

Today NS3 is used by many researchers to perform their research in networking field. New protocol can be written and tested in NS3 by using c++ language with an extension of .cc, after compiling these .cc files NS3 generates .pcap file and .tr files. These .tr files generated by the NS3 can be

analyzed to get the throughput, delay, jitter, packet delivery ratio for new protocol developed. Trace file format of NS3 is totally different from that of trace format of NS2. We have developed new trace file analyzer using Java so that researcher can spend much of his/her time and efforts in developing a new protocol than analyzing a trace file. The trace file format of NS2 and NS3 are totally different. Extracting data from NS3 trace file is more difficult than NS2 trace file. Awk script or perl script are written to extract the exact data from the column of .tr file to calculate the performance. Graph can be plotted by this extracted through .awk or perl script in gnuplot.

NS3 trace analyzer reads the .tr file and automatically calculates network performance parameters like throughput, packet delivery ratio, packet drop, packet sent, packet received. It makes post-processing of .tr file very easy. This NS3 Trace Analyzer was developed with the intention that the researchers can solely focus on design and development of new protocol than wasting much of his/her time in analyzing the .tr file.

In this chapter section 4.2 gives a brief introduction about using of NS-3 simulator, section 4.3 describes the trace file format of NS3 in detail, Section 4.4 describes the development of NS-3 Trace analyzer, network performance parameters and finally Section 4.5 concludes and also discusses about future improvement which can make NS3 more user friendly.

6.2 USING NS3

NS3 is a powerful network simulator which is being used by researchers to perform the research today.NS3 has a set of network simulation models implemented in C++ and wrapped in Python. User program written in C++ or Python uses library which instantiates a set of simulation models to set up the simulation scenario, enters the simulation main loop, and exits

when the simulation is completed. User writes a new protocol using C++ in NS3. After compiling this file NS3 stores the results in .pcap files

or .tr files. NS3 tracing of a file can be done in two ways

- Ascii tracing: AsciiTraceHelper that is associated to a NetDevice object is used to create .tr file like that was created in NS2. Note that the trace file format of NS2 and NS3 are completely different.
- Pcap tracing: This generates .pacap files which can be analysed in wireshark or tcpdump. Wireshark is a GUI application but tcpdump shell command are used to analyze a file.

As discussed above .pcap file are analyzed in wireshark but .tr files generated by AsciiTraceHelper are very big and .awk scripts or perls scripts are used to extracts the required data from these files to get the performance metrics of the protocol like throughput,delay, pdf etc. Trace file format of NS3 is totally different from that of NS2.Extracting the data from the NS3 trace file needed some additional operations to be performed on the trace file. We developed simple NS3 trace file analyzer in lava that reads the NS3 .tr file and gives throughput,delay, pdf etc.

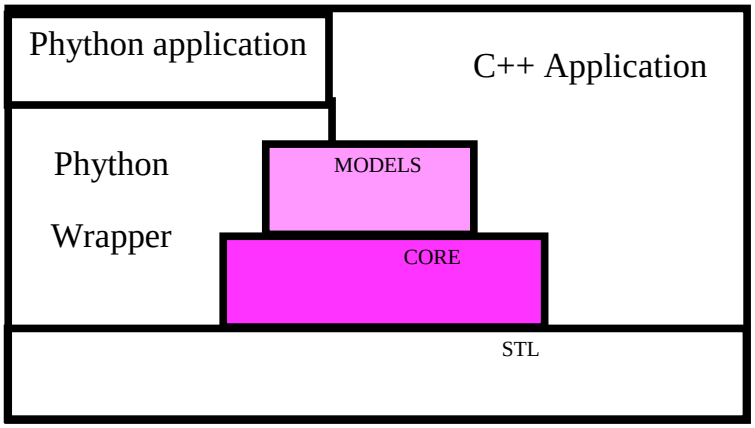

Figure 6.1 Program binding of NS3

6.3 NS3 TRACE FILE

The trace data in NS3 trace file is in ASCII code and is organized in 12 fields which are discussed below. The Figure 6.2 shows the format of the ns3 packet trace.Where Event: Every NS3 trace file starts file an event descriptor which can either +, -, d or r where

Event	Time	From	To	Type	Size	Flags	Class	Source	Dest	Seq	Id

Figure 6.2 NS3 Trace File Format

+ indicates a packet was enqueued.

- indicates a packet was dequeued.

d indicates a packet was dropped.

R indicates a packet was received.

Time: Next field in the NS3 file is time which indicates the time at which event occurred.

From: Starting node for the link on which event has occurred.

To: Ending node for the link on which event has occurred.

Type: Type indicates type of packets

Size: Size indicates size of packets in bytes.

Flags: ignore

Class: The class of the packet, which can be used to identify particular tcp connection.

Source: Source address

Destination: Destination address.

seq: Sequence Number of the packet

Id: Identifier of the packet.

Sample of NS3 trace file is showed in Figure 6.3.

```
+ 1 /NodeList/0/DeviceList/0/$ns3::PointToPointNetDevice/TxQueue/Enqueue ns3::PppHeader (Point-to-Point Protocol: IP (0x0021)) ns3::Ipv4Header
(tos 0x0 DSCP Default ECN Not-ECT ttl 64 id 0 protocol 6 offset (bytes) 0 flags [none] length: 56 10.1.1.1 > 10.1.1.2) ns3::TcpHeader (49153 >
8080 [SYN] Seq=0 Ack=0 Win=32768 ns3::TcpOptionWinScale(2) ns3::TcpOptionTS(1000;0) ns3::TcpOptionEnd(EOL))
- 1 /NodeList/0/DeviceList/0/$ns3::PointToPointNetDevice/TxQueue/Dequeue ns3::PppHeader (Point-to-Point Protocol: IP (0x0021)) ns3::Ipv4Header
(tos 0x0 DSCP Default ECN Not-ECT ttl 64 id 0 protocol 6 offset (bytes) 0 flags [none] length: 56 10.1.1.1 > 10.1.1.2) ns3::TcpHeader (49153 >
8080 [SYN] Seq=0 Ack=0 Win=32768 ns3::TcpOptionWinScale(2) ns3::TcpOptionTS(1000;0) ns3::TcpOptionEnd(EOL))
r 1.00509 /NodeList/1/DeviceList/0/$ns3::PointToPointNetDevice/MacRx ns3::PppHeader (Point-to-Point Protocol: IP (0x0021)) ns3::Ipv4Header
(tos 0x0 DSCP Default ECN Not-ECT ttl 64 id 0 protocol 6 offset (bytes) 0 flags [none] length: 56 10.1.1.1 > 10.1.1.2) ns3::TcpHeader (49153 >
8080 [SYN] Seq=0 Ack=0 Win=32768 ns3::TcpOptionWinScale(2) ns3::TcpOptionTS(1000;0) ns3::TcpOptionEnd(EOL))
+ 1.00509 /NodeList/1/DeviceList/0/$ns3::PointToPointNetDevice/TxQueue/Enqueue ns3::PppHeader (Point-to-Point Protocol: IP (0x0021))
ns3::Ipv4Header (tos 0x0 DSCP Default ECN Not-ECT ttl 64 id 0 protocol 6 offset (bytes) 0 flags [none] length: 56 10.1.1.2 > 10.1.1.1)
ns3::TcpHeader (8080 > 49153 [SYN|ACK] Seq=0 Ack=1 Win=32768 ns3::TcpOptionWinScale(2) ns3::TcpOptionTS(1005;1000) ns3::TcpOptionEnd(EOL))
- 1.00509 /NodeList/1/DeviceList/0/$ns3::PointToPointNetDevice/TxQueue/Dequeue ns3::PppHeader (Point-to-Point Protocol: IP (0x0021))
ns3::Ipv4Header (tos 0x0 DSCP Default ECN Not-ECT ttl 64 id 0 protocol 6 offset (bytes) 0 flags [none] length: 56 10.1.1.2 > 10.1.1.1)
ns3::TcpHeader (8080 > 49153 [SYN|ACK] Seq=0 Ack=1 Win=32768 ns3::TcpOptionWinScale(2) ns3::TcpOptionTS(1005;1000) ns3::TcpOptionEnd(EOL))
r 1.01019 /NodeList/0/DeviceList/0/$ns3::PointToPointNetDevice/MacRx ns3::PppHeader (Point-to-Point Protocol: IP (0x0021)) ns3::Ipv4Header
(tos 0x0 DSCP Default ECN Not-ECT ttl 64 id 0 protocol 6 offset (bytes) 0 flags [none] length: 56 10.1.1.2 > 10.1.1.1) ns3::TcpHeader (8080 >
49153 [SYN|ACK] Seq=0 Ack=1 Win=32768 ns3::TcpOptionWinScale(2) ns3::TcpOptionTS(1005;1000) ns3::TcpOptionEnd(EOL))
+ 1.01019 /NodeList/0/DeviceList/0/$ns3::PointToPointNetDevice/TxQueue/Enqueue ns3::PppHeader (Point-to-Point Protocol: IP (0x0021))
ns3::Ipv4Header (tos 0x0 DSCP Default ECN Not-ECT ttl 64 id 1 protocol 6 offset (bytes) 0 flags [none] length: 52 10.1.1.1 > 10.1.1.2)
ns3::TcpHeader (49153 > 8080 [ACK] Seq=1 Ack=1 Win=32768 ns3::TcpOptionTS(1010;1005) ns3::TcpOptionEnd(EOL))
- 1.01019 /NodeList/0/DeviceList/0/$ns3::PointToPointNetDevice/TxQueue/Dequeue ns3::PppHeader (Point-to-Point Protocol: IP (0x0021))
ns3::Ipv4Header (tos 0x0 DSCP Default ECN Not-ECT ttl 64 id 1 protocol 6 offset (bytes) 0 flags [none] length: 52 10.1.1.1 > 10.1.1.2)
ns3::TcpHeader (49153 > 8080 [ACK] Seq=1 Ack=1 Win=32768 ns3::TcpOptionTS(1010;1005) ns3::TcpOptionEnd(EOL))
+ 1.01019 /NodeList/0/DeviceList/0/$ns3::PointToPointNetDevice/TxQueue/Enqueue ns3::PppHeader (Point-to-Point Protocol: IP (0x0021))
ns3::Ipv4Header (tos 0x0 DSCP Default ECN Not-ECT ttl 64 id 2 protocol 6 offset (bytes) 0 flags [none] length: 588 10.1.1.1 > 10.1.1.2)
ns3::TcpHeader (49153 > 8080 [ACK] Seq=1 Ack=1 Win=32768 ns3::TcpOptionTS(1010;1005) ns3::TcpOptionEnd(EOL)) Payload Fragment [0:536]
```

Figure 6.3 NS3 Trace File

6.4 PROPOSED SYSTEM

The NS3 trace analyzer is developed on Ubuntu platform in Java. This trace analyzer analyzes the NS3 trace file. Trace file format of NS3 is totally different from that of NS2. NS3 trace file is composed of both text and numbers suppose we want to read the packet id alone from the trace file from a particular column it gives ns3::TcpOptionTS(9994;9979) similarly for packet size etc hence some operations are performed on the .tr file to extract the required value from the NS3 trace file.

6.4.1 Architecture of NS3 Trace Analyzer

Architecture of NS3 trace analyzer can be divided into three parts. Read layer, Process layer and then Presentation layer. Read layer reads the NS3 file, compile .cc file generate .tr file. Process layer does the text Processing of the .tr files. Awk scripts or perl scripts are used to extract the data from .tr file column by column. Required data is filtered column wise to calculate throughput, delay, packet delivery ratio, jitter etc. NS3 Trace analyzer automatically does the processing like data extraction from .tr file

calculation of performance parameters so that it is viewed in the presentation layer. Presentation layer displays the results and generates graphs.

Architecture of NS3 Trace Analyzer is shown in Figure 6.4

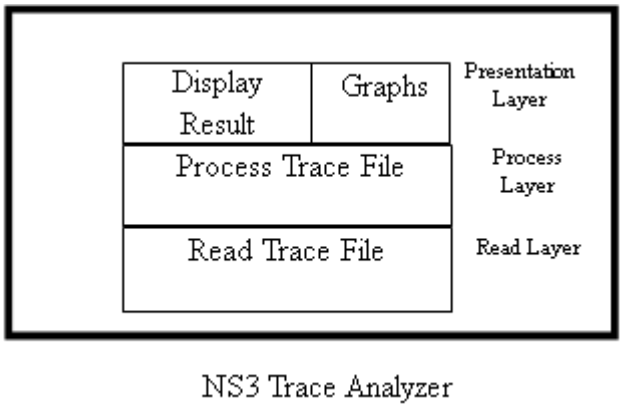

Figure 6.4 Architecture of NS3 Trace Analyzer

6.4.2 Performance Parameters

Network Performance refers to measures of quality of service like bandwidth, throughput, delay, jitter, error etc. NS3 Trace Analyzer has considered following performance parameters they are

1. Packet sent
2. Packet Received
3. Packet Dropped
4. Packet Delivery Ratio
5. Throughput

Packet Sent : Number of Packets sent by the source node

Packet Sent is calculated by following formula

Ps – Let Ps be number of packet sent
initially Ps=0;

event – Gets the first column of the tr file

here + refers packets enqueued (trace file format of NS3)

if(event == ' +')

Ps=Ps+1

Packet Received : Number of Packets received by the destination node

Packet Received is calculated by following formula

Pr-Let Pr be number of packet received

initially Pr=0;

event – Gets the first column of the tr file

here r refers packets received (trace file format of NS3)

if(event == ' r')

Pr=Pr+1

Packet Dropped : Number of Packets dropped

Packet dropped is calculated by following formula

Ds – Let Ds be number of packet received

initially Ds=0;

event – Gets the first column of the tr file

here d refers packets dropped (trace file format of NS3)

if(event == 'd ')

Ds=Ds+1

Packet Delivery Ratio: Packet Delivery Ratio (PDF) is ratio of number of packets delivered to the number of packets received

Pdf is calculated by formula

Let pdf be the packet delivery ratio

pdf=(Pr/Ps)*100

Throughput: Throughput is number of packets sent per unit time. Throughput is measured in bits per second. Throughput is calculated as

Let Tr be throughput

Tr=(Pr*100)/time

6.4.3 Graphical User Interface For NS3 Trace Analyzer

NS3 trace analyzer is designed in Java. NS3 trace analyzer has three tab. First tab is used to open and save your .tr file. The second tab is used to generate graphs. The third tab will display the result that is the network performance parameter like throughput, pdf,packet received, packet dropped. At the command prompt user can also see .tr file being read.

GUI screen shot are shown in Figure 6.5, 6.6, 6.7.

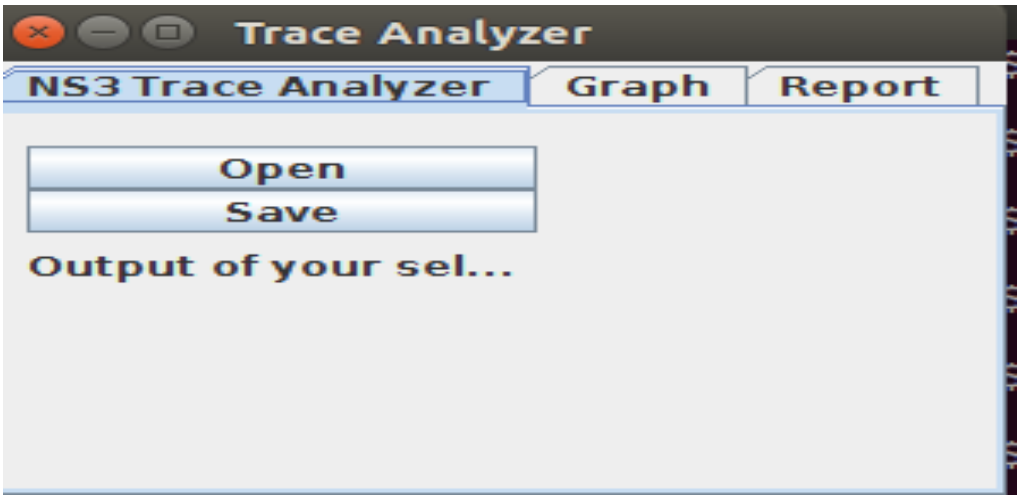

Figure 6.5 GUI Screen Shot 1

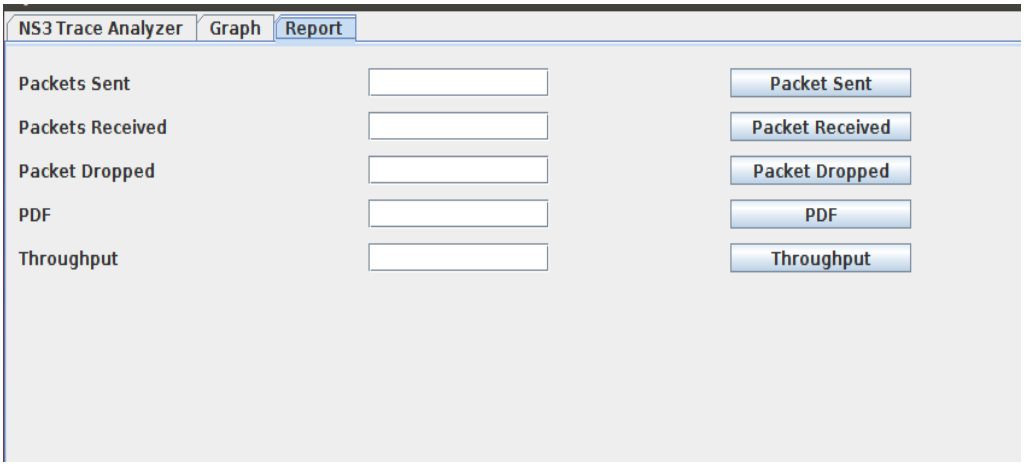

Figure 6.6 GUI Screen Shot 2

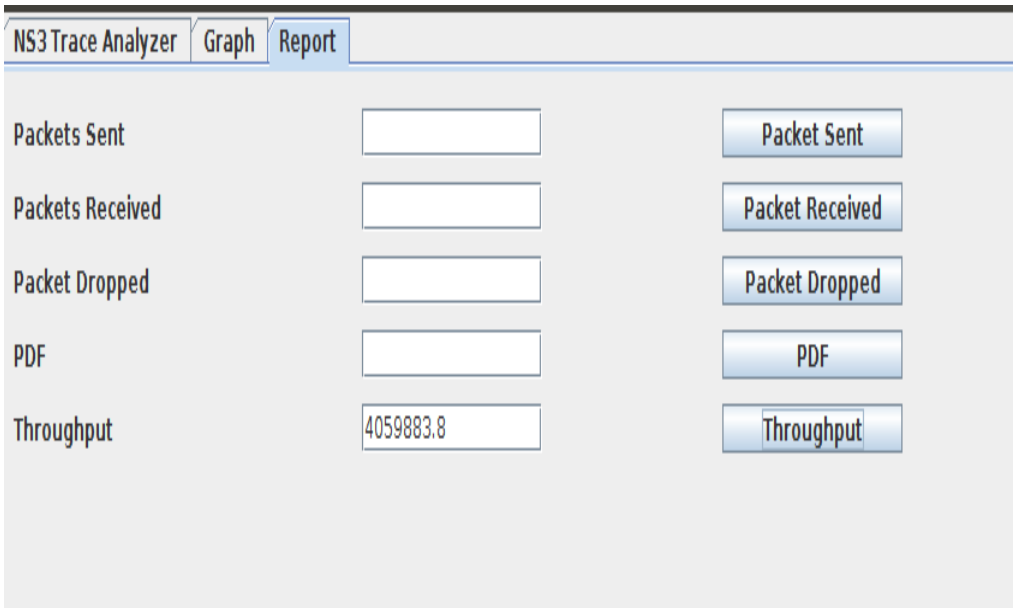

Figure 6.7 GUI Screen Shot 3

6.5 SUMMARY

The proposed system is developed in java which reads the .tr files of NS3 and displays the network parameters like throughput, pdf, pack loss etc. also generates graphs. Trace file format of NS3 is totally different from

that of NS2. NS3 trace file is composed of both text and number suppose we want to read the packet id alone from the trace file from a particular column it gives ns3::TcpOptionTS(9994;9979) similarly for packet size etc. hence some operations were performed on the .tr file to extract the required value from the NS3 trace file. . Extracting the required data from the .tr file is difficult since the string should separated with space to get only values required for calculating performance metrics. Since it is developed in Java it can run on both windows as Linux platform team required lot of time to understand this .tr file of ns3 and then to extract the exact values from the file for the calculation of network performance metrics. With the aim that researcher can solely contribute in development and testing of a new protocol this tool is being developed. This tool can be further developed by adding more network parameters making it more user friendly.