12-2009

Automated System to Debug Under-performing Network Flows in Wide Area Networks

Harika Tandra
University of Tennessee - Knoxville

Recommended Citation

Tandra, Harika, "Automated System to Debug Under-performing Network Flows in Wide Area Networks. " Master's Thesis, University of Tennessee, 2009.
https://trace.tennessee.edu/utk_gradthes/562

To the Graduate Council:

I am submitting herewith a thesis written by Harika Tandra entitled "Automated System to Debug Under-performing Network Flows in Wide Area Networks." I have examined the final electronic copy of this thesis for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Computer Engineering.

Hairong Qi, Major Professor

We have read this thesis and recommend its acceptance:

Itamar Arel, Robert C.Ward

Accepted for the Council:
Carolyn R. Hodges

Vice Provost and Dean of the Graduate School

(Original signatures are on file with official student records.)

To the Graduate Council:

I am submitting herewith a thesis written by Harika Tandra entitled "Automated system to debug under-performing network flows in wide area networks". I have examined the final electronic copy of this thesis for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science with a major in Computer Engineering.

Hairong Qi, Major Professor

We have read this thesis and recommend its acceptance.

Itamar Arel

Robert C.Ward

Accepted for the Council:

Carolyn R.Hodges

Vice Provost and Dean of

the Graduate School

(Original signatures are on file with official student records.)

Automated system to debug under-performing network flows in wide area networks

A Thesis Presented for
the Master of Science
Degree
The University of Tennessee, Knoxville

Harika Tandra

December 2009

Dedicated to my parents, ammamma and Rahul.

Acknowledgements

I would like to thank my advisors, Greg S. Cole and Dr.Hairong Qi for their constant support and inspirational guidance, without which this work would not have been possible. I would also like to thank Dr. Robert C. Ward and Dr. Itamar Arel for agreeing to serve on my thesis committee.

I am very thankful to Dr. Qi for giving me the opportunity to work under her and be a part of the Advanced Imaging and Collaborative Processing (AICIP) lab. The time spent in several meetings with her has always motivated me to broaden my horizons and learn new things. I would also like to thank AICIP members for their inputs and suggestion about my work.

I would also like to convey my heartfelt thanks to my mentor and co-advisor Greg S. Cole, for giving me the opportunity to work as a Graduate Research Assistant in the GLORIAD project. Working for GLORIAD has been the best part of my graduate study. It not only provided a great opportunity to horn my technical skills but also gave me the rare opportunity to witness how dedicated work makes things possible. Working under the highly experienced and talented GLORAID team - Anita Colliatie, Predrag Radulovic, and Susie Baker was a great learning experience. Also I am extremely indebted to Anita for her care and support.

I want to thank my brother Rahul Tandra. He has always been my inspiration. I would like to thank him for always setting high standards and encouraging me to do better at each step. He has always given me the confidence to fight back my fears and think positively. Growing up with him is undoubtedly the best part of my life.

I also want to thank my friends Anusha and Neeharika aka ‘chimpu’, who are unquestionably the best gifts I got from being at UT. I cannot imagine life away home without their extraordinary support and encouragement. Anu’s constant chatter, Chimpu’s jokes, the slow extended dinners while watching online movies, our cooking disasters and our ever unsuccessful fitness plans are but a few of the numerous sweet memories that I shall treasure forever. I would also like to thank other friends at UT, Shambhu, Meena and Janak for their wonderful company. I will miss the regular playing badminton session and occasional hiking trips. I would also like to thank Sirisha and Lakshmi for the memorable dance practice sessions.

I would like to thank my family without whose love and support none of this was possible. My parents, for the many sacrifices they made to provide us with the best education and resources. My dad, for instilling good values and principles in me. My mom, for her immense patience and love. My granny (ammamma) for being my oldest best friend. My uncles Uday and Sagar for always being there by my side and for all their love and pampering.

I would also like to thank my teachers Prasad sir, Subramaniam sir and Reddy sir under whose guidance I spent some of the most formative years of my life. Last but not the least, I would like to thank my other friends Divya, Rashmy, Anusha, Swapna, Karthik, Sana and Fariha for being a part of my life.

Abstract

Locating the cause of performance losses in large high performance Wide Area Networks (WAN) is an extremely challenging problem. This is because WANs comprise several distributed sub-networks (Autonomous Networks), with their own independent network monitoring systems. Each individual monitoring system has limited or no access to network devices outside its own network. Moreover, conventional network monitoring systems are designed only to provide information about the health of individual network devices, and do not provide sufficient information to monitor end-to-end performance – thus, adding severe overhead on debugging end-to-end performance issues.

In this thesis, an automated tool is designed that requires no special access to network devices and no special software installations on the network devices or end hosts. The system detects performance losses and locates the most likely problem nodes (routers/links) in the network. A key component of this system is the novel hybrid network monitoring/data collection system. The monitoring/data collection sub-system is designed to obtain the best of both active and passive monitoring techniques. Then, pattern analysis algorithms are designed. They locate the causes of performance loss using the data collected from above sub-system.

This system is being tested on the GLORIAD (Global Ring Network for Advanced Application Development) network. One of the future goals is to integrate this system into the GLORIAD's network monitoring tool set, to provide end-to-end network monitoring and problem mitigation capabilities.

Table of contents

List of figures

1. Introduction

Today there are larger number of applications and technologies that are dependent on high speed, reliable networks. There is rapid increase in the number of distributed applications and applications using services on the cloud, server virtualization, etc. Gigabit data transfers are very common and we are moving from the ‘giga era’ to the ‘tera era’. For example, scientists and researchers around the world require sharing of terabytes of data and computing resources to make advances in the fields of High energy physics, Global climate change studies, Biotechnology etc.

Network monitoring and management has a vital role in ensuring that these goals of high performance are delivered to the end-users. These network-monitoring systems not only ensure smooth running of the network but also provide valuable information about network performance characteristics, routing patterns, routing anomalies, etc. This information is used in planning network upgrades, developing new network models/protocols and for efficient resource management.

In this thesis we consider the problem of locating the cause of performance losses in large high performance WANs (Wide Area Networks). WANs, as the name suggests are networks that are spread in a large area, spanning many ISPs’(Internet Service Providers). They usually consist of several autonomous networks managed by independent network management systems. Debugging performance issues in such large WANs is extremely difficult and challenging. Since there is no centralized network management in these networks, troubleshooting tends to be a time consuming and frustrating process.

To summarize, the challenges in debugging performance issues in WANs are:

1) They span several ISP’s with independent network management systems.
2) There is no direct access or control to network devices outside ones own autonomous network.
3) Limited knowledge of the network topology.
4) Traditional network management systems only monitor for either *up* or *down* states of network resources. For example, SNMP(Simple Network Management Protocol) based counters provide only basic link utilization statistics, i.e. bits per second, error counts etc.

In this thesis, our goal is to develop an automated system that detects performance losses on the network and then locates the likely cause (faulty routers/links) of the performance loss. The main design goal is to have a system that requires no special access to different network devices. Furthermore, this should be easy to install and can run on any network.

1.1. <u>GLORIAD project</u>

The Global Ring Network for Advanced Applications Development (GLORIAD) is an advanced science internet network constructed as an optical ring around the Northern Hemisphere connecting universities, national labs and science facilities across the U.S., Russia, China, Korea, Canada, Netherlands and the Nordic countries with individual network circuits providing up to 10 Gbps.

GLORIAD is a NSF supported project lead by Greg Cole. The foundation of the GLORAID project dates back to 1997, when the first U.S. - Russia network connection was started by the **MIRNET** project. This was followed by the **Little GLORIAD** project in 2001-2004. It connected U.S., Russia, and China with a 155Mbps ring. This was followed by the current GLORIAD project (2005-2009) which added 7 new partner nations. At present, GLORIAD is expanding its network to connect Singapore, India and Egypt under the new GLORIAD - Taj project. Figure 1 shows GLORIAD's network map with the new links under the Taj expansion shown in orange.

In any 24 hour time period GLORIAD sees more than 60,000 IP addresses and more 700,000 significant flows carrying over 2 terabytes of data. Some of the top-users of the network are NASA, Fermilab, NOAA, UC Berkeley, University of Illinois, NIH, the U.S. Department of Energy and the National Center for Atmospheric Research [1].

GLORIAD is not just a network, its main aim is to enable global science collaborations by ensuring reliable network performance. To meet this goal equipment is deployed to obtain metrics for network performance, utilization and security on all its major network links. The data obtained from the monitoring system is used to alert the network-operations team on performance issues for fault mitigation, etc. The network monitoring data collection system is explained in Chapter 5.

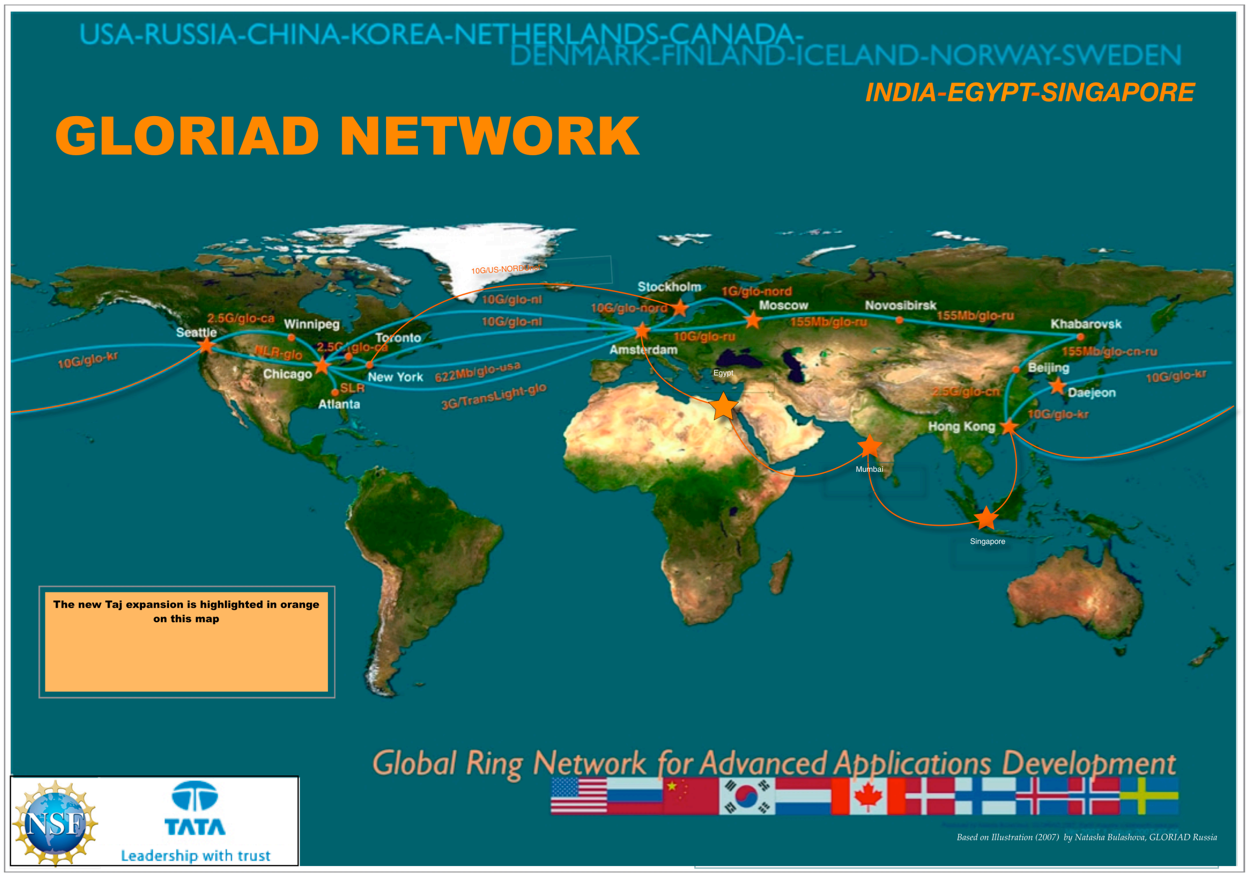


Figure 1. GLORIAD network map

Obtaining network-monitoring data is only one part of the problem. The main challenge lies in analysis of the data and presenting it in a format useful for the network operation teams. To achieve this, GLORIAD has proposed the concept of Distributed Virtual Network Operations Center (dvNOC). The dvNOC is a new model for a distributed and decentralized

network management. It integrates four main areas of networking: *utilization*, *performance*, *operations* and *security*. It is a user interface for international teams to share information at appropriate levels and cooperatively manage and troubleshoot network issues. The dvNOC consists of several tabs, each tab is devoted to some aspect of network monitoring, fault-mitigation or troubleshooting.

1.1.1. GLORIAD- Earth application

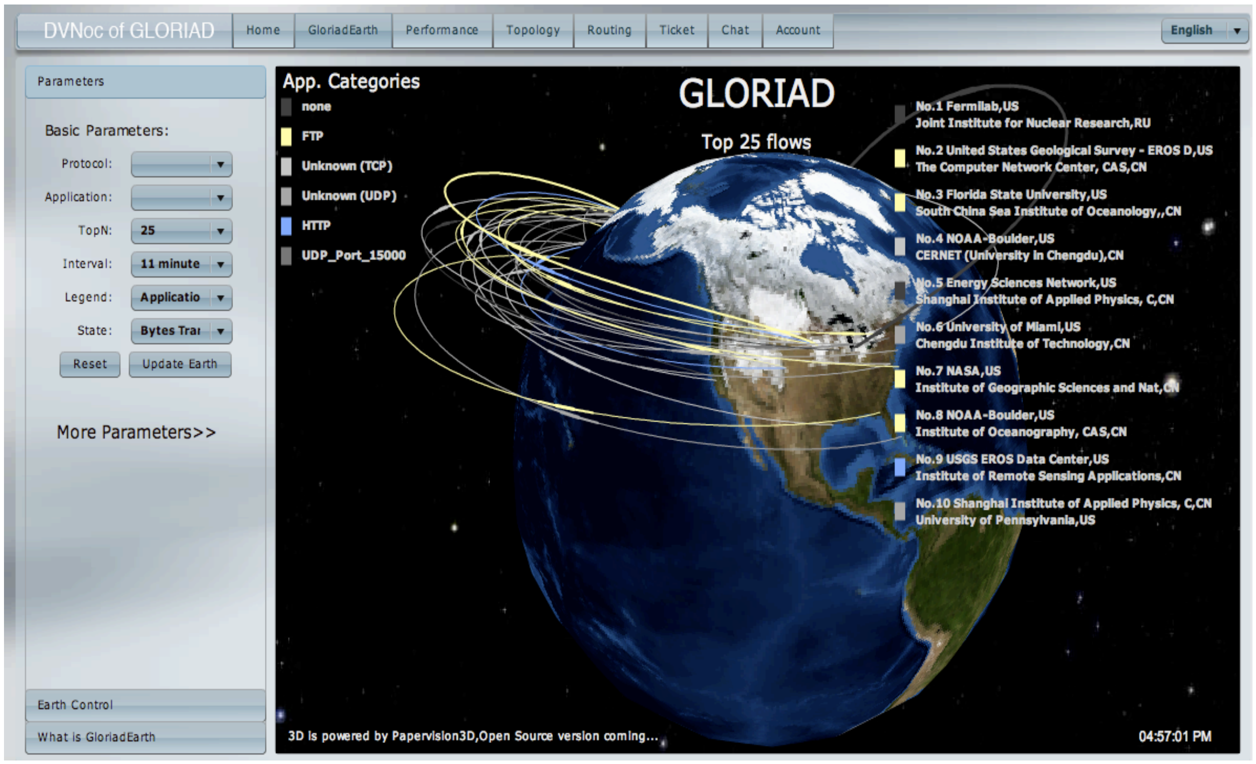
The GLORIAD - Earth is one of the tabs in dvNOC. It shows in real time the top-users of the network. It shows the live network utilization information like the current top-users, top applications using the network, etc. It can give detailed information like top users of specific applications, during a given timeframe, in certain countries, between certain continents, etc.

GLORAID earth is a 3-D animation showing a rotating global on which the current top network-flows are geographically mapped. Figure 2 shows a snapshot the GLORAID- Earth. Each ring in the figure represents a network-flow. The details of the top-users are displayed on the right side of the screen, while the criteria for classification are shown on the top left corner of the screen. More details about GLORIAD can be obtained from [2].

1.2. Thesis contribution

This thesis gives the basic design of a network diagnostic tool to locate causes of performance losses in WANs. It suggests a framework for design of network debugging tools especially suitable for WANs. The design consists of two key components,

1) **The data monitoring/collection sub-system**: This sub-system uses passive and active monitoring methods to detect under-performing flows in the network and gather the data needed to locate the cause of performance losses.
2) **The data analysis and visualization sub-system**: This sub-system implements pattern analysis algorithms to locate the cause of losses in the



Basic Parameters:

Protocol:
Application:
TopN: 25
Interval: 11 minute
Legend: Applicatio
State: Bytes Trai

App. Categories

- none
- FTP
- Unknown (TCP)
- Unknown (UDP)
- HTTP
- UDP_Port_15000

No.1 Fermilab,US
Joint Institute for Nuclear Research,RU

No.2 United States Geological Survey - EROS D,US
The Computer Network Center, CAS,CN

No.3 Florida State University,US
South China Sea Institute of Oceanology,,CN

No.4 NOAA-Boulder,US
CERNET (University in Chengdu),CN

No.5 Energy Sciences Network,US
Shanghai Institute of Applied Physics, C,CN

No.6 University of Miami,US
Chengdu Institute of Technology,CN

No.7 NASA,US
Institute of Geographic Sciences and Nat,CN

No.8 NOAA-Boulder,US
Institute of Oceanography, CAS,CN

No.9 USGS EROS Data Center,US
Institute of Remote Sensing Applications,CN

No.10 Shanghai Institute of Applied Physics, C,CN
University of Pennsylvania,US

3D is powered by Papervision3D,Open Source version coming...

04:57:01 PM

Figure 2. The GLORIAD Earth tab of the dvNOC interface, showing the top network flows in real-time

network flows using the data collected by the monitoring/collection sub-system.

The above two sub-systems are described in detail in the following chapters.

1.3. Thesis outline

Firstly in Chapter 3, the various factors influencing the network performance and the TCP throughput characteristics are explained. In this chapter, various factors affecting the TCP throughput are examined. It explains the effects of latency, MTU (Maximum Transmission Unit) size, TCP window size and loss rate on TCP throughput. Based on the application requirements, users can tune the MTU size and TCP window size to improve TCP throughput. The network management can improve throughput by decreasing delay and loss rate values. It is seen that reducing loss rate by half (with round trip time constant) improves the throughput by 41% [3]. Then a comparison is made between the TCP throughput achieved by the network flows on GLORIAD network and the theoretical upper bound given by the formula by Mathis et. al [3]. This study showed the current performance problems and the improvements possible by reducing the packet loss. This formed the motivation and goal of this thesis, i.e. to improve end-user performance by reducing the packet loss rate.

In Chapter 4, the data monitoring/collection system is explained. The data monitoring/collection system uses a hybrid monitoring approach which combines passive and active monitoring methods. The NetFlow based passive monitoring data collected from a central router in the network is used as the input. An input filter is designed to identify under-performing network flows. Next, it uses the MTR (My Trace Route) tool to launch tests for gathering further data about the health of the route taken by the under-performing flow. The MTR tool gives performance statistics of each router on the route to the given destination.

Chapter 5 explains the data collected from MTR and the various forms of data visualization methods used to represent and explore the data. A single MTR run gives individual router performance statistics. When the MTR results collected over a short time interval are combined it gives a snapshot of the network router topology along with performance statistics of each individual router. These network router topology snapshots are used to represent the data collected and later to localize the weak point or problem nodes in the network. Various forms of these network topology graphs are also explained.

Finally, Chapter 6 explains the criteria used for identifying faulty nodes from the network topology graphs generated in Chapter 5. It explains three different cost functions to identify a faulty node. Based on the costs, nodes are classified into three categories, labeled as *yellow, orange and red*. The nodes labeled *red* being the most likely problem nodes, followed by *orange and yellow* in decreasing order of probability. Lastly, a training set is used to determine the classification boundaries of the node costs.

2. Overview of network monitoring techniques

2.1. Introduction

Network monitoring is defined as the “information collection function of network management” [4]. A network management system uses network monitoring to constantly check for slow or failing network devices or services. There are three basic goals of network monitoring [4]: Performance monitoring, Fault monitoring, Account monitoring. This thesis deals with the performance monitoring aspect of network monitoring.

Significance of performance monitoring

Apart from keeping the network up and running, performance monitoring is used for finding routing behavior and anomalies, planning future network upgrades, enabling better network resource management, creating new network models and protocols, providing feedback to end-user applications for adaptive control and resource tuning like pointing users to specific TCP enhancements based on the application.

Monitoring techniques are typically classified into two categories:

1) Passive monitoring: NetFlow and packet sniffers like tcpdump, Packet analyzer, etc. are examples of few tools that use this method.
2) Active monitoring: Ping, traceroute, tcptraceroute, etc. are commonly used active monitoring based tools.

2.2. <u>Passive monitoring</u>

In passive monitoring technique, the monitoring system captures live data from a particular point(s) within the network. This point is called an *observation point*. The captured data is either processed offline or online (as the packets are captured). SNMP and NetFlow are popular passive monitoring protocols. These are also called router-based monitoring methods, since the monitoring functionalities are embedded into the routers.

2.2.1. Simple Network Monitoring Protocol (SNMP)

SNMP is an application layer protocol that consists of 3 components:

- Network Management System (NMS) – polls managed devices
- Agent – keeps information of managed device
- Managed device – device to be monitored

In this method the NMS system polls network devices implementing SNMP interface for regular performance statistics. The agent on the device keeps device information and responds to requests from NMS. NMS can get metric like CPU utilization, average CPU load, data throughput, packets forwarded etc. The managed device can also generate alerts for certain events. These alerts are called traps.

There are many SNMP based tools that provide user-friendly frontend to SNMP based data. MRTG, Netstatus and Omnibook are a few of them.

2.2.2. NetFlow based monitoring

The NetFlow network protocol was developed by Cisco. It is part of Cisco IOS based equipment. It was designed to collect IP information of traffic grouped by *flows*. A '*flow*' is defined as the set of packets that share the following 7 values [5]:

1) Source IP
2) Destination IP
3) Source port
4) Destination port
5) Protocol type in the IP header
6) Ingress interface of the router
7) IP type of service

Apart from the above 7 values the router keeps information like Number of bytes in the flow, TCP flags, routing information i.e. IP address of the next-hop, etc. The NetFlow version-9 templates are the current standard for the flow records. The '*Internet Protocol Flow Information eXport (IPFIX)*' is the IETF standard developed based on NetFlow version-9 template.

A typical NetFlow based monitoring system architecture is shown in Figure 3. NetFlow exporter emits NetFlow records as UDP packets to the collector. The exporter can be configured to expire flows at regular intervals (even when the flow is not complete). The exported data is collect at the collector. There are many applications that analyze NetFlow records to produce detailed graphs on network activity. For example ntop, NetFlow analyzer, etc.

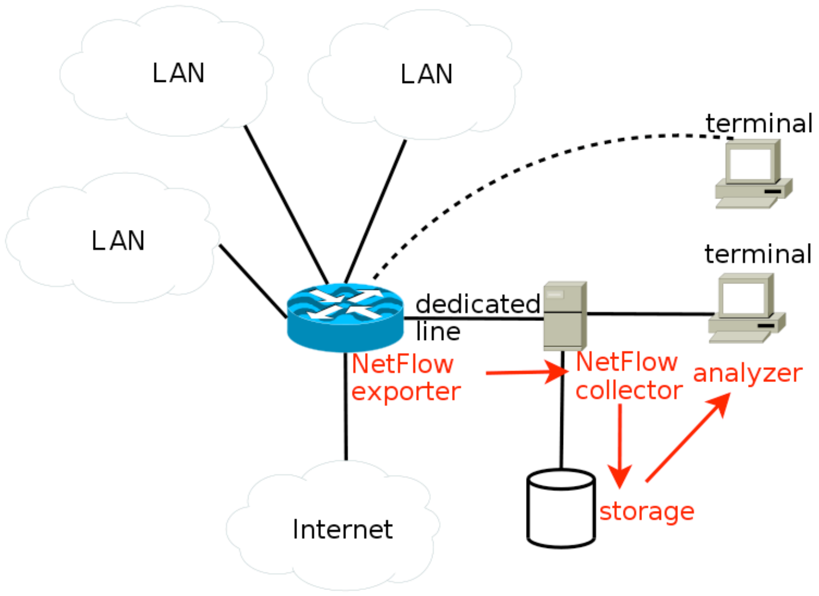


Figure 3 NetFlow architecture. Adapted from NetFlow wiki.

2.2.3. Packet sniffers

Packet sniffing programs capture packets at a network interface. The disadvantage with this method is that the packet analysis is mostly done off-line after the packet capture. It does not scale well on high-speed links. Popular packet sniffing tools are tcpdump, Packet analyzer, Wireshark etc.

2.3. Active monitoring

Active monitoring techniques provide end-to-end performance measures, unlike passive monitoring methods that measure network performance at a single point. In active monitoring test packets are injected into the network between two given end points. The test packets are used to collect metrics like route, packet loss, jitter, round trip time, availability, etc. The disadvantage of active monitoring is that it generates additional network traffic. Traceroute and ping are the most commonly used active monitoring tools.

2.3.1. Traceroute

Traceroute gives the *trace of the route* taken by packets to a given destination. It sends packets with small TLL values to the given destination. At each hop the TTL value of a packet is decreased by 1. When a packet's TTL value expires (TTL becomes 1), an ICMP Time Exceed packet is sent back. On receiving an ICMP Time Exceed packet, the host discovers the address to the router at intermediate hops. Figure 4 illustrates the above process. There are two main types of Traceroute tools:

1) UDP-based: In this, traceroute sends UDP packets to high- numbered destination ports (since these are usually unused) and waits for UDP port unreachable message for discovering the destination. This is the default traceroute in Mac, FreeBSD and Linux systems.

2) ICMP-based: In this, traceroute sends ICMP echo request packets instead of UDP packets. Processing ICMP echo request is simpler compared to sending UDP port unreachable messages. It was found that ICMP-based traceroute reaches more destinations than the UDP- based method [6].

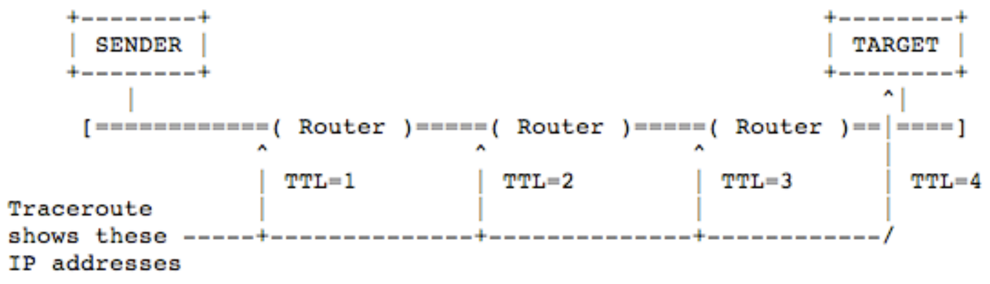


Figure 4. Traceroute illustration Adopted from [7]

2.3.2. Ping

Ping is commonly used diagnostic tool to check availability of a network device. It measures the round trip time and packet loss to the given IP. It sends a series of ICMP echo request packets to the destination and waits for ICMP echo reply packets. For each packet sent it calculates rtt.

2.3.3. ICMP-based tools for end-to-end performance measures

ICMP-based tools are popular tools for measuring end-to-end performance statistics. This is because they require no special software installations, they are easy to use and work universally. But they have a few limitations like:

1) **ICMP filters:** Misuse of ICMP services has led to ICMP packets being blocked by many firewalls. In some other cases the response to ICMP packets is rate limited [8]. Some of the routers block response to ICMP echo request packets.

2) **Loss asymmetry:** The loss rate on the forward path to a host is different from the loss rate on the reverse path. With ICMP-based tools one cannot distinguish if the packet is lost on the forward path or on the reverse path. This distinction can be important for certain application like media streaming protocol, in which only packet loss on the forward path is important.

3. Understanding network performance

3.1. <u>Overview</u>

Network performance is dependent on several factors like latency, loss, window size, MTU size, routing etc. This chapter explains, the main factors that effect TCP throughput and the reasons for why in spite of a high bandwidth link one does not get high throughput. Then the effect of packet loss on TCP throughput is studied using the Mathis formula [3]. Lastly, the throughput achieved on by real network traffic on GLORIAD is studied.

3.2. <u>Factors effecting TCP throughput</u>

3.2.1. Delay or Latency

Latency is the time taken by a packet to move from one point to another in the network. It is measured by Round Trip Time (RTT). There are two types of delay,

<u>Propagation delay</u>: It includes delay introduced by the medium, delays in routers/switches etc. The speed of light imposes a minimum delay of,

$$delay >= distance / speed\ of\ light\ in\ the\ medium$$

The speed of light in various media:

3.0x108m/s in free space

~2.3x108m/s in copper

~2.0x108m/s in fiber = 200 km / ms

Transmission delay: It is the time taken to transmit all the bits in a packet on to the physical link. It depends on the link speed and packet size. For low speed links transmission delay is limiting factor while at high speeds propagation delay can be the limiting factor. Figure 5 illustrates the delay imposed by the speed of light in fiber.

3.2.2. TCP window size

The TCP window size gives the maximum number of bytes that a receiver can buffer at a time. The sender can send a maximum of window size bytes before waiting for acknowledgement from the receiver. The window size can be between 2 to 65,535bytes. On a reliable link, TCP performance

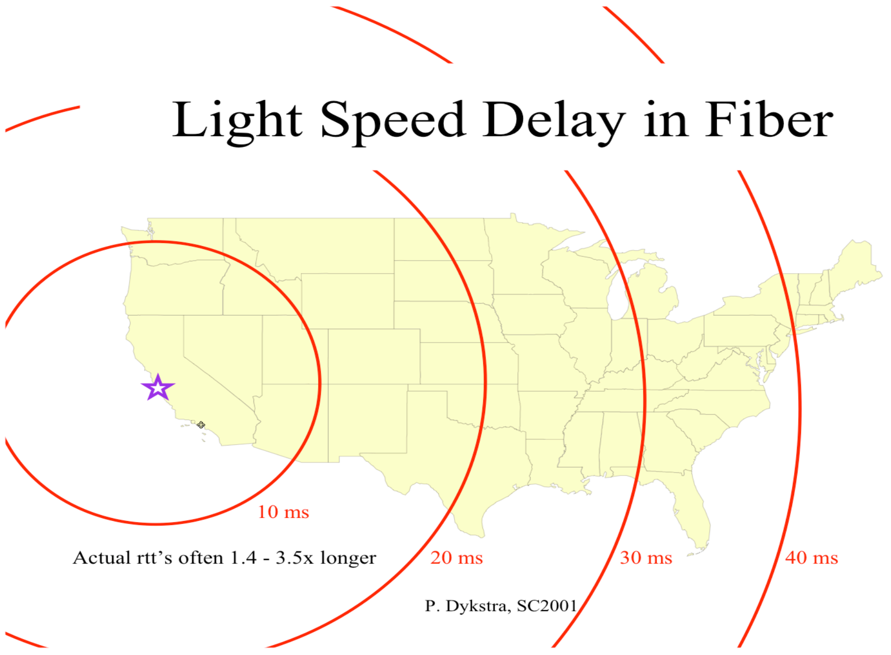

Light Speed Delay in Fiber
10 ms
Actual rtt's often 1.4 - 3.5x longer
20 ms
30 ms
40 ms
P. Dykstra, SC2001

Figure 5 Speed of light delay in Fiber. Adopted from P. Dykstra, SC2001

(throughput) is best when the receive window size is set to be greater than or equal to the *link capacity* or *bandwidth-delay product*.

$$\textit{Window size} >= \textit{bandwidth} * \textit{delay}$$

The disadvantage with a large window size is that if there is a packet loss then the entire window is retransmitted. Thus lowering throughput. This can be avoided by using *selective acknowledgements* (TCP enhancement).

Converse of the above gives the maximum possible throughput possible given the window size and delay.

$$\textit{bps} <= (\textit{TCP window size} / \textit{rtt})$$

3.2.3. MTU (Maximum Transmission Unit)

MTU gives the maximum number of bytes that can be transferred in a single frame. By increasing the MTU size more data can be transferred with less overhead of packet headers and less routing decisions. Thus improving the throughput. On the other hand it may lead to fragmentation of packets.

3.3. Packet loss

The effect of packet loss on throughput can be seen using the formula from the paper *'The macroscopic behavior of the TCP congestion avoidance algorithm by Mathis et al. '*.[3]

Mathis formula gives upper bound on the TCP rate,

$$Rate <= \frac{MSS}{RTT * \sqrt{p}}$$

where:

Rate – is the TCP transfer rate or throughput (in Kbps)

MSS – is the maximum segment size (typically 1460 * 8 bits)

RTT – is the round trip time (in ms)

p – is the packet loss rate

Observations

From the above formula we can see that,

- Doubling MTU size doubles throughput.
- Reducing latency by half, doubles throughput.
- Reducing loss rate by half, bring 41% rise in throughput.

It is difficult to reduce the round trip time but reducing loss rate will bring significant increase in network performance. Figure 6 is a plot of throughput versus rtt for different loss rates. Similarly Figure 7is a 3-dimensional plot of throughput from Mathis formula for varying rtt and loss rates. It is apparent that throughput can be improved by many folds by minimizing the loss rate.

3.4. Throughput seen on GLORIAD network

In this section, the throughput achieved by the flows on the GLORIAD network is analyzed. GLORIAD's monitoring system gives the bytes transferred per flow, the flow duration and the bytes retransmitted per flow. From this data the throughput in bps and loss rate (retransmission per bytes) is computed. The round trip time is estimated using the standard conversion factor of 2.5ms is equivalent to a distance of 100Km [9]. The physical distance is estimated using the geo location (longitude and latitude) information of the source and destination IP's.

Figure 8 shows the 3-dimensional plot of throughput (bps), rtt (ms) and loss rate of flows from US to China over a 24-hour period on the GLORIAD network. The plot is a ‘needle plot’, where each ‘black dot’ is a data point corresponding to a single flow.

The throughput achieved by the flows is compared with the theoretical upper bound given by Mathis formula. It shows some that some flows achieve throughput greater than the upper bound. This could be because of TCP enhancement methods like selective acknowledgements, parallel streams, etc., adopted by the end users.

Throughput derived from loss & RTT using Mathis formula

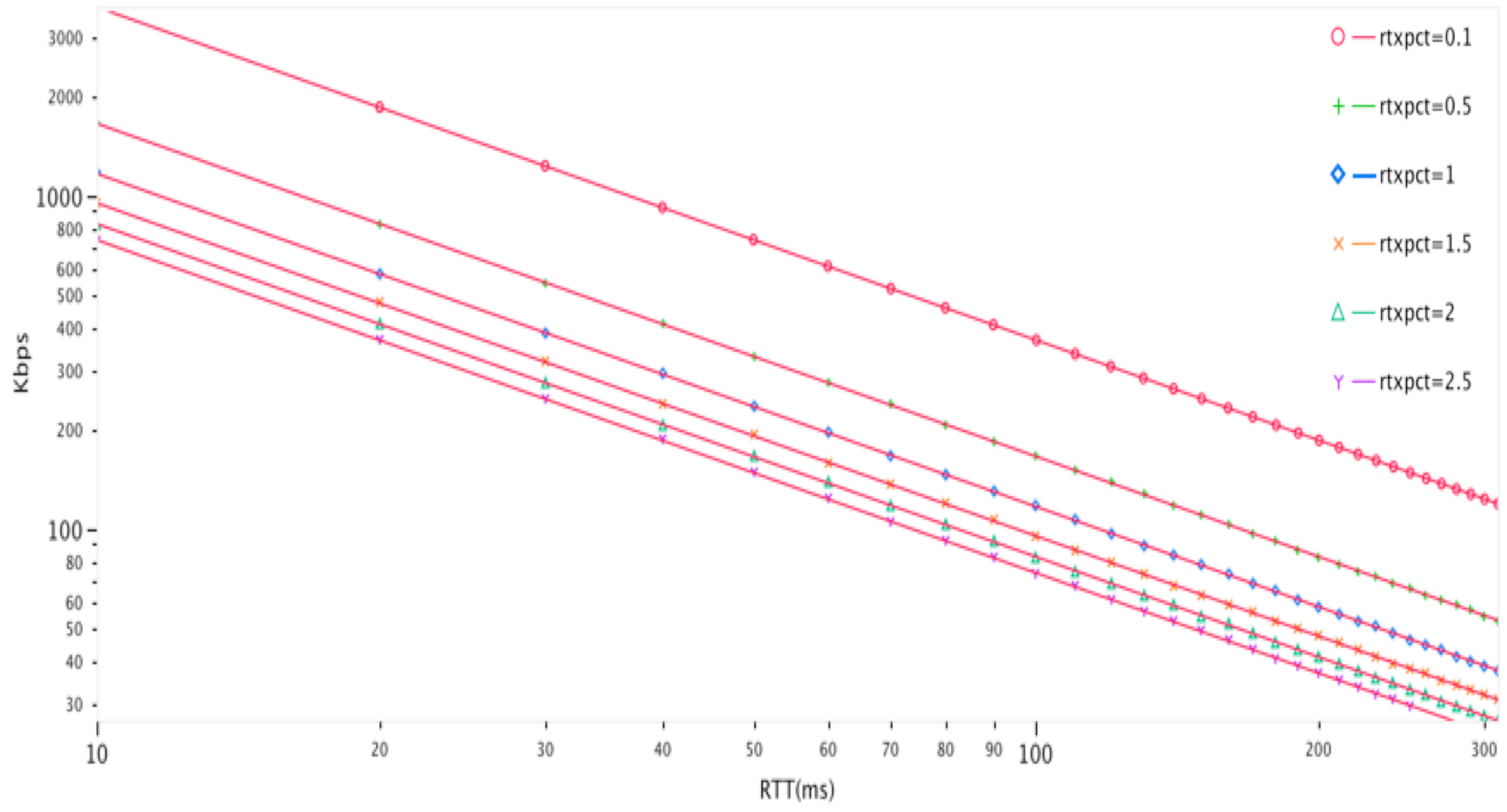


Figure 6. Throughput versus RTT for various loss rates using Mathis formula.

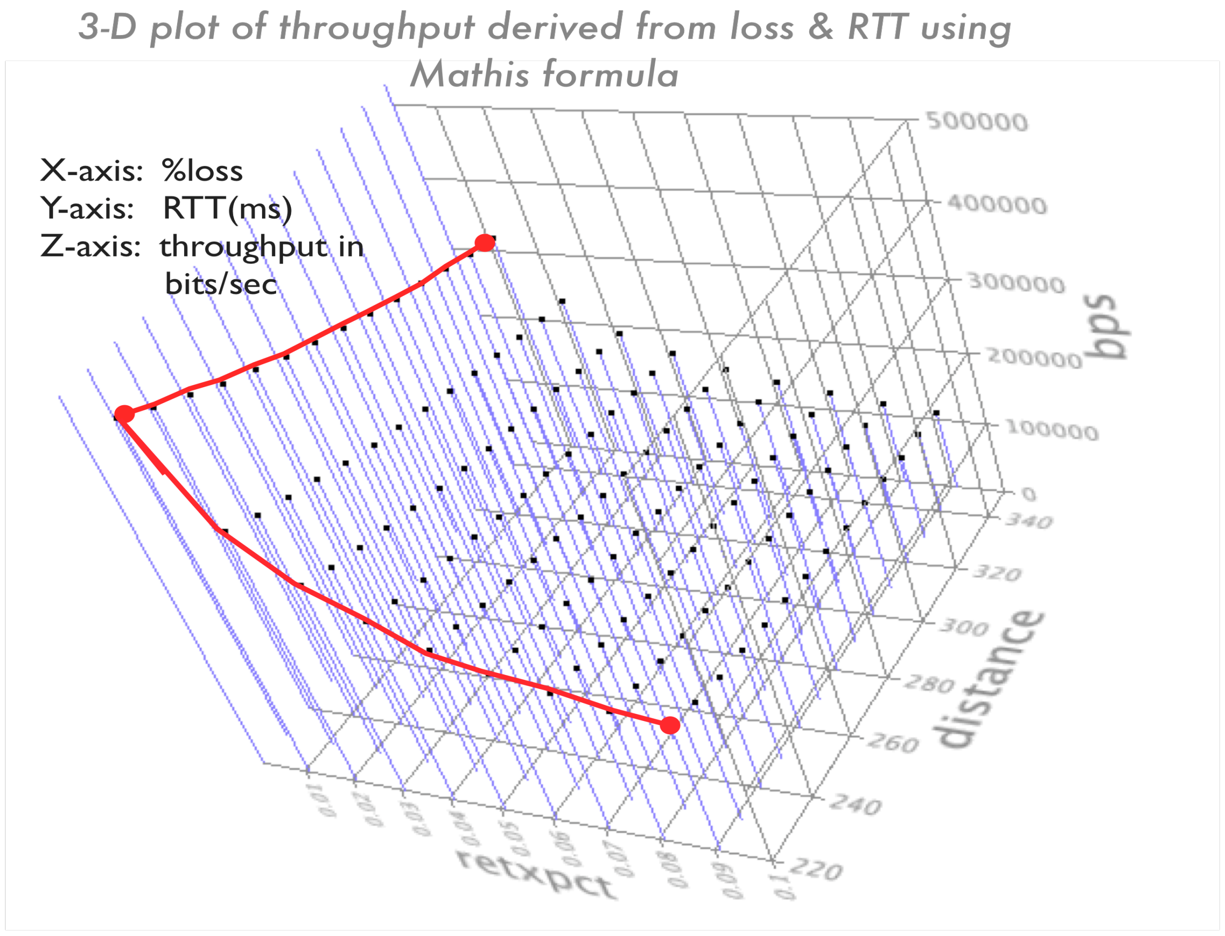


Figure 7. 3D plot loss, rtt, throughput using Mathis formula

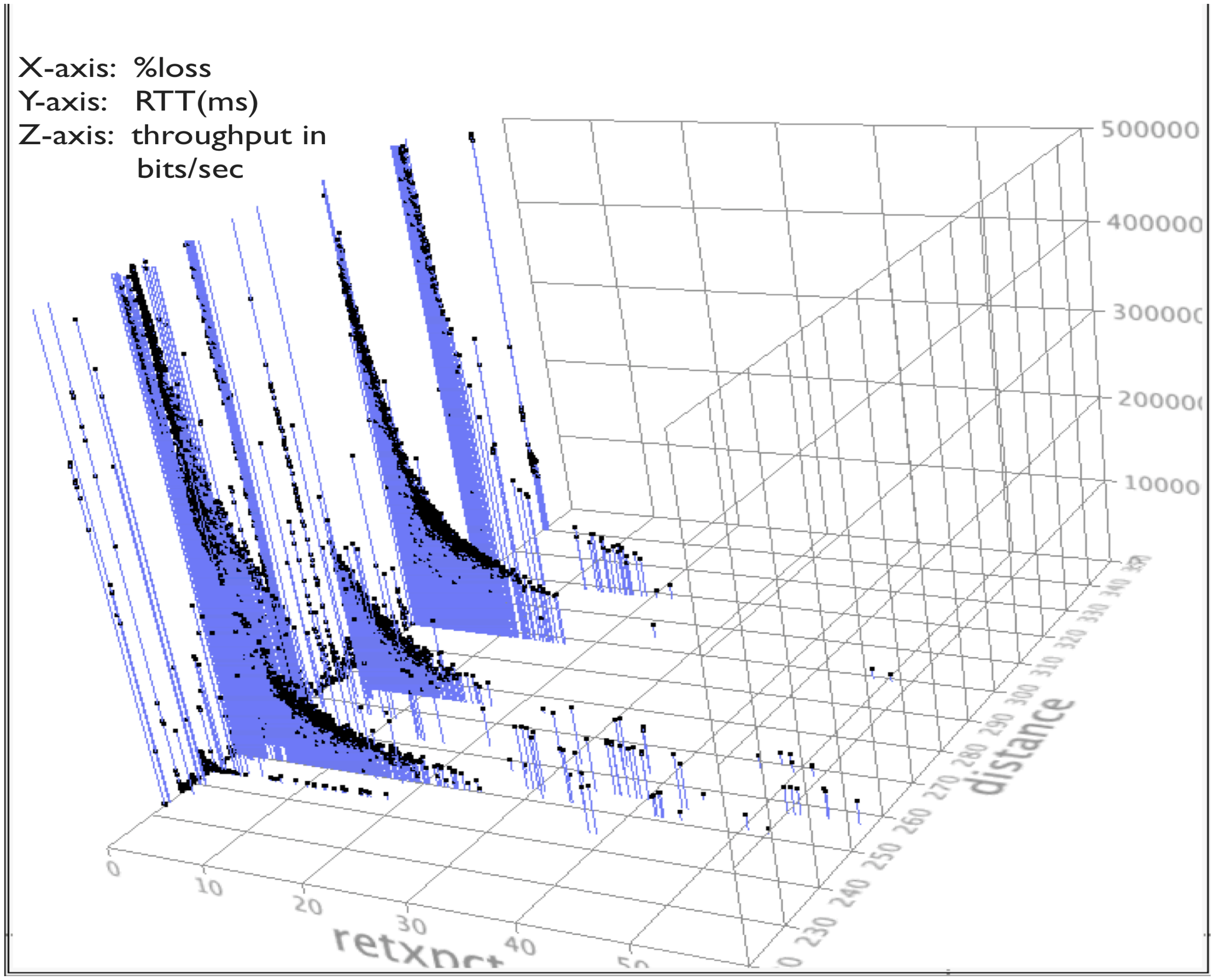


Figure 8. 3D plot of loss, rtt, throughput using GLORIAD flow data from US to CN over a 24hr time period.

4. Monitoring/data collection system

4.1. Overview

In this chapter a novel hybrid network monitoring/data collection system is introduced. It attempts to use the best of both active and passive network monitoring mechanisms. First the under-performing flows are detected from the input flow data, collected using passive NetFlow based monitoring system. Next, the active monitoring mechanism (using MTR tool) is used to investigate further into each under-performing flow and obtain performance statistics of each router in the path. The active monitoring measurements are triggered in near-real-time (at the time when an under-performing flow is detected). The following sections explain the setup and working of the passive and active monitoring/data collection sub-systems

4.2. Passive monitoring sub-system

The automated system to debug under-performing flows requires NetFlow based monitoring data as its input. There are many commercial flow based measuring and monitoring tools like Cisco's NetFlow Analyzer, Packeteer's PacketShaper etc. and free open source tools like nProbe, nfdump, ntop, etc, which can be used to obtain the input NetFlow data required by this system. The current system uses 'Extended-NetFlow' format data from the GLORAID's network monitoring system as input. It can be easily extended to read from other formats like NetFlow v5, v9, sFlow, etc.

4.2.1. Gloriad's passive data acquisition/management system

GLORIAD has established a state-of-the-art network monitoring system to better understand network utilization patterns and performance of individual network flows to improve end-to-end performance. The Figure 9 shows layout of GLORIAD's network monitoring/data collection system. This system is divided into two components:

<u>Data Acquisition system</u>: It uses Packeteer's PacketSeeker 10000 monitoring box to passively monitor the traffic. The PacketSeeker box monitors traffic on all the GLORIAD links terminating or originating at its central router in Chicago. It outputs "extended-NetFlow" format monitoring information. The extended-NetFlow information includes basic performance metrics mentioned in NetFlow v5 format, application based classification of the traffic (using "deep packet inspection"), TCP QoS data like average round trip time and bytes retransmitted, etc. The data exported consists of attributes and metrics like source and destination IP address, flow start time, flow end time, protocol, retransmissions, bytes, port source and destination,

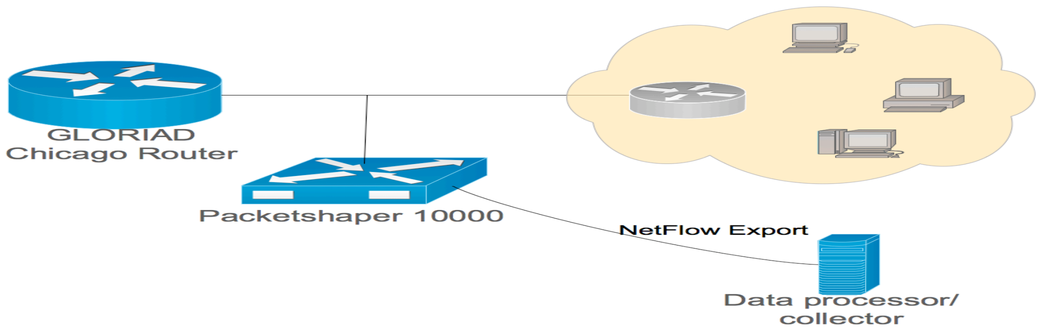


Figure 9. GLORIAD 's passive monitoring system setup

source and destination ASNUMs, etc. per each flow.

<u>Data management system</u>: This module consists of complex set of processes to process and store the data obtained from the Packeteer box. During this processing additional information like the domain names, IP longitude/latitude and ASNUMs are added to the flow data The processed flow data is stored in MySQL databases. The Table 1 below describes few attributes stored in the database that are relevant to the current system.

The figures below show examples of the flow records stored in the database. In Figure 10 the ‘dom_s’, ‘dom_d’ attributes represent domainid of the source and the destination, respectively. Figure 11 shows the domain name and other information corresponding to the domainids’ in the flow records shown in Figure 10

4.2.2. Detect under-performing flows

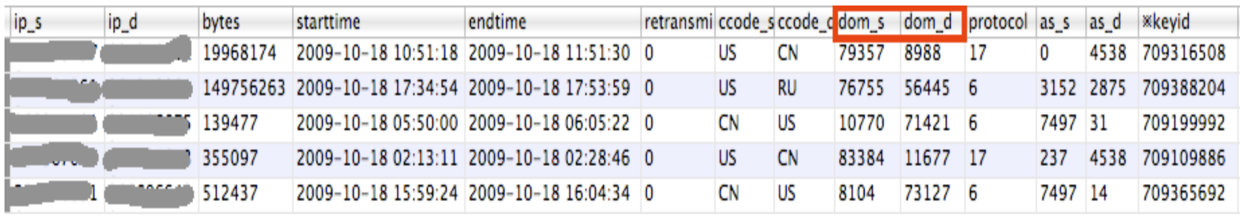
An input filter is designed to detect under-performing flows in real-time i.e. when the flow record are stored in the database. The test parameters are selected based on performance baseline drawn from GLORIAD’s historic data, dating back to 2005.

A flow is labeled as under-performing if it satisfies each of the following criteria:

1) % retransmissions/bytes > 1.0
2) bytes > 5000000 (5MB)
3) *frequency* > *4hr*; same (ip_s, ip_d) pair is not labeled as under-performing for the minimum time period set by the *frequency* parameter. This is done to avoid repeatedly triggering active measurement

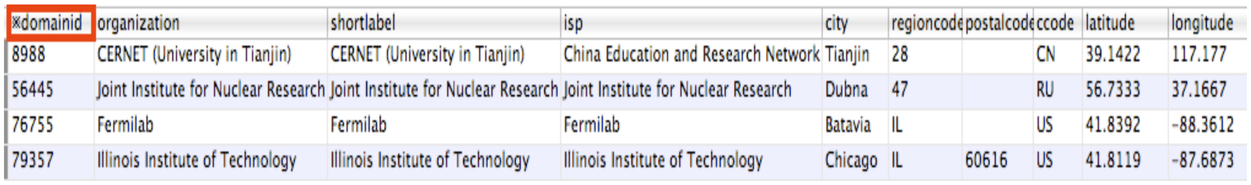
Table 1. Metrics provided by passive monitoring system.

Attributes	Description
ip_src	Source IP address of the flow
ip_dst	Destination IP address of the flow
retransmissions	Bytes retransmitted per flow
bytes	Number of bytes transferred per flow
starttime	Flow start time
endtime	Flow end time
protocol	Type of layer-4 protocol. ICMP=1, TCP=6, UDP=17
asnum_s	Source Autonomous System Number (ASNUM)
asnum_d	Destination Autonomous System Number (ASNUM)
ccode_s	Country code of the source of the flow
ccode_d	Country code of the destination of the flow
dom_s	Domainid that maps source IP to an institutional (i.e., domain) record
dom_d	Domainid that maps destination IP to an institutional (i.e., domain) record
keyid	Unique id that identifies each flow record in the database



ip_s	ip_d	bytes	starttime	endtime	retransmi	ccode_s	ccode_c	dom_s	dom_d	protocol	as_s	as_d	keyid
[redacted]	[redacted]	19968174	2009-10-18 10:51:18	2009-10-18 11:51:30	0	US	CN	79357	8988	17	0	4538	709316508
[redacted]	[redacted]	149756263	2009-10-18 17:34:54	2009-10-18 17:53:59	0	US	RU	76755	56445	6	3152	2875	709388204
[redacted]	[redacted]	139477	2009-10-18 05:50:00	2009-10-18 06:05:22	0	CN	US	10770	71421	6	7497	31	709199992
[redacted]	[redacted]	355097	2009-10-18 02:13:11	2009-10-18 02:28:46	0	US	CN	83384	11677	17	237	4538	709109886
[redacted]	[redacted]	512437	2009-10-18 15:59:24	2009-10-18 16:04:34	0	CN	US	8104	73127	6	7497	14	709365692

Figure 10. Sample flow records showing source and destination domain-id



domainid	organization	shortlabel	isp	city	regioncode	postalcode	ccode	latitude	longitude
8988	CERNET (University in Tianjin)	CERNET (University in Tianjin)	China Education and Research Network	Tianjin	28		CN	39.1422	117.177
56445	Joint Institute for Nuclear Research	Joint Institute for Nuclear Research	Joint Institute for Nuclear Research	Dubna	47		RU	56.7333	37.1667
76755	Fermilab	Fermilab	Fermilab	Batavia	IL		US	41.8392	-88.3612
79357	Illinois Institute of Technology	Illinois Institute of Technology	Illinois Institute of Technology	Chicago	IL	60616	US	41.8119	-87.6873

Figure 11. Domain description for the respective domain id's above

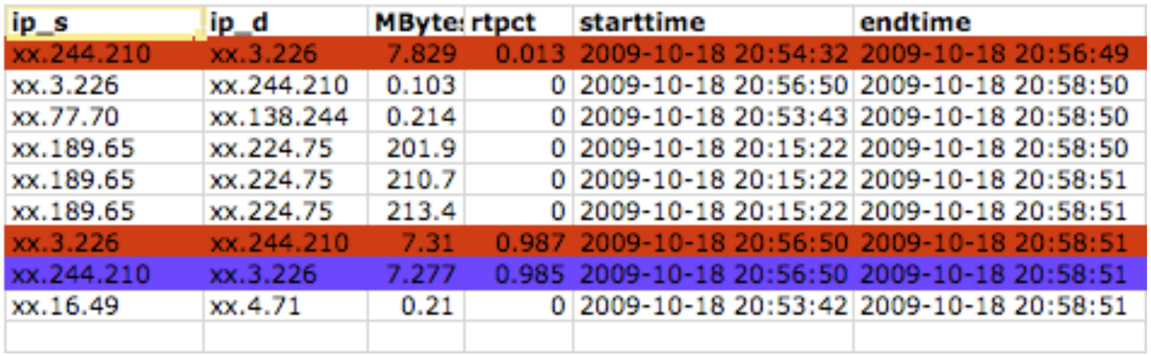
tests against long-running, poor-performing flows.

Figure 12 shows input flows that are processed by the test filter. The flows marked in ‘red’ satisfy the above three conditions and are thus labeled as ‘under-performing’ flows. Figure 13 shows table storing the flows that satisfy the test conditions. In Figure 14 the flow marked in ‘blue’ is not stored as ‘under-performing’ flow as it doesn’t satisfy the 3rd condition on frequency.

4.3. Active monitoring sub-system

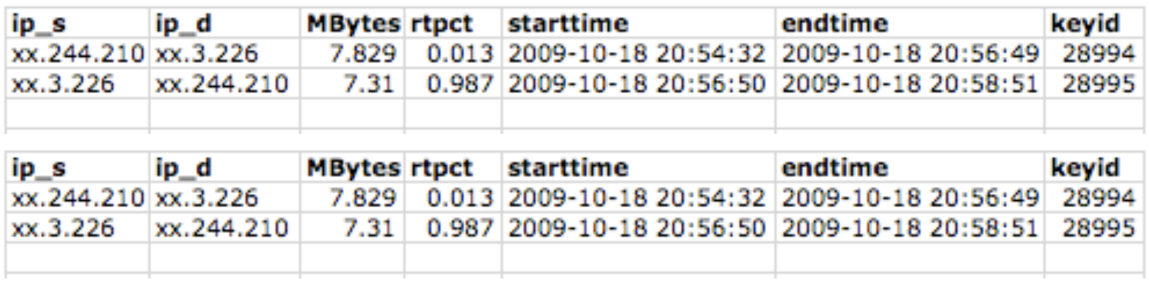
4.3.1. Introduction

To detect the likely cause of performance loss in a flow more information is needed regarding the health of the routers/links on the path taken by the packets in that flow. To get this information the MTR network diagnostic tool is used. MTR is an active network-monitoring tool. First, working of the MTR tool is explained followed by the system design to obtain and store the data.



ip_s	ip_d	MByte:	rtpct	starttime	endtime
xx.244.210	xx.3.226	7.829	0.013	2009-10-18 20:54:32	2009-10-18 20:56:49
xx.3.226	xx.244.210	0.103	0	2009-10-18 20:56:50	2009-10-18 20:58:50
xx.77.70	xx.138.244	0.214	0	2009-10-18 20:53:43	2009-10-18 20:58:50
xx.189.65	xx.224.75	201.9	0	2009-10-18 20:15:22	2009-10-18 20:58:50
xx.189.65	xx.224.75	210.7	0	2009-10-18 20:15:22	2009-10-18 20:58:51
xx.189.65	xx.224.75	213.4	0	2009-10-18 20:15:22	2009-10-18 20:58:51
xx.3.226	xx.244.210	7.31	0.987	2009-10-18 20:56:50	2009-10-18 20:58:51
xx.244.210	xx.3.226	7.277	0.985	2009-10-18 20:56:50	2009-10-18 20:58:51
xx.16.49	xx.4.71	0.21	0	2009-10-18 20:53:42	2009-10-18 20:58:51

Figure 12. Input flows to the filter



ip_s	ip_d	MBytes	rtpct	starttime	endtime	keyid
xx.244.210	xx.3.226	7.829	0.013	2009-10-18 20:54:32	2009-10-18 20:56:49	28994
xx.3.226	xx.244.210	7.31	0.987	2009-10-18 20:56:50	2009-10-18 20:58:51	28995

ip_s	ip_d	MBytes	rtpct	starttime	endtime	keyid
xx.244.210	xx.3.226	7.829	0.013	2009-10-18 20:54:32	2009-10-18 20:56:49	28994
xx.3.226	xx.244.210	7.31	0.987	2009-10-18 20:56:50	2009-10-18 20:58:51	28995

Figure 13. Result of filtering the above input flows.

4.3.2. What is MTR?

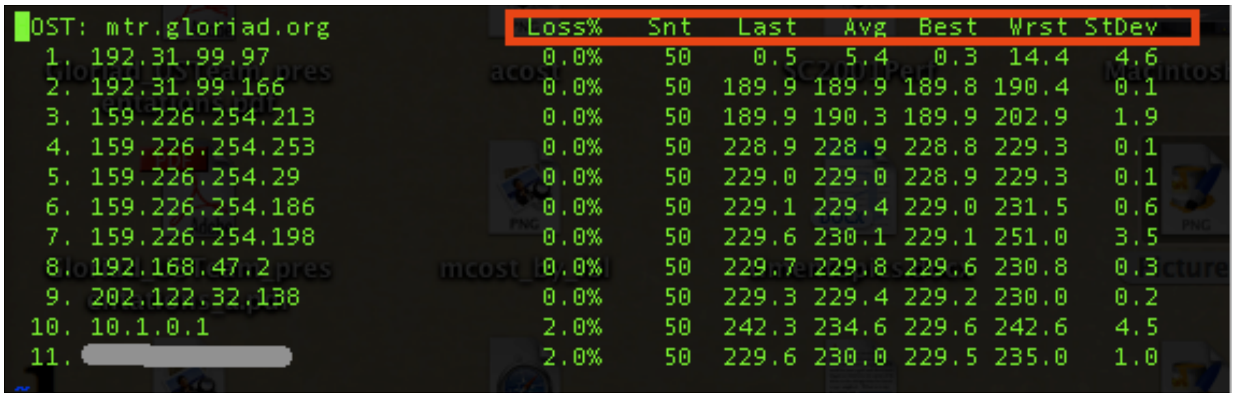
MTR stands for My Trace Route (previously called Matt's Trace Route). It is an open source network diagnostic tool. It combines the functionality of 'trace route' and 'ping' tools.

On triggering MTR run on a destination IP address. It starts by discovering the address of routers that are on the route taken from the host to the destination IP. Next, it sends ICMP request packets to each of the intermediate routers and generates statistics like average round trip time (rtt), best rtt, worst rtt, last rtt, % loss and packets sent. Figure 14 shows a sample MTR output.

4.3.3. Working of MTR

First, MTR discovers the route to the destination IP by sending ICMP echo request packets with low ttl (time to live) values to the destination IP. The ttl value of packets is decreased by 1 at each hop. When an intermediate router receives a packet with ttl value equal to 1, an ICMP time exceed packet is sent back to the MTR host machine. On reaching the actual destination an ICMP echo reply packet is sent to the MTR host machine. Thus, MTR discovers the route to the given destination.

Once MTR discovers the route there are several ways to get the



```
 OST: mtr.gloriad.org                Loss%   Snt   Last   Avg  Best  Wrst StDev
  1. 192.31.99.97                     0.0%    50    0.5   5.4   0.3  14.4   4.6
  2. 192.31.99.166                    0.0%    50  189.9 189.9 189.8 190.4   0.1
  3. 159.226.254.213                  0.0%    50  189.9 190.3 189.9 202.9   1.9
  4. 159.226.254.253                  0.0%    50  228.9 228.9 228.8 229.3   0.1
  5. 159.226.254.29                   0.0%    50  229.0 229.0 228.9 229.3   0.1
  6. 159.226.254.186                  0.0%    50  229.1 229.4 229.0 231.5   0.6
  7. 159.226.254.198                  0.0%    50  229.6 230.1 229.1 251.0   3.5
  8. 192.168.47.2                     0.0%    50  229.7 229.8 229.6 230.8   0.3
  9. 202.122.32.138                   0.0%    50  229.3 229.4 229.2 230.0   0.2
 10. 10.1.0.1                         2.0%    50  242.3 234.6 229.6 242.6   4.5
 11. [redacted]                       2.0%    50  229.6 230.0 229.5 235.0   1.0
```

Figure 14. Sample MTR output

subsequent statistics of rtt and loss. One method is to send ICMP echo request packets addressed to each intermediate router and wait for ICMP echo reply packets. However it is seen that most of the routers do not send ICMP echo-reply messages but send ICMP time exceed messages. Therefore, MTR sends ICMP request packets with low ttl values addressed to the destination IP and waits for ICMP time exceed reply.

One can specify the number of packets MTR sends to each intermediate router with the '-c' parameter. This parameter stands for number of cycles. MTR counts each second as a cycle and it sends one packet per intermediate route per cycle. Hence, if the route has 8 hops, at every 1/8th of a second MTR sends a packet with ttl values from 1 to 8 respectively. Therefore, each intermediate router gets a total of 'c' packets per MTR run.

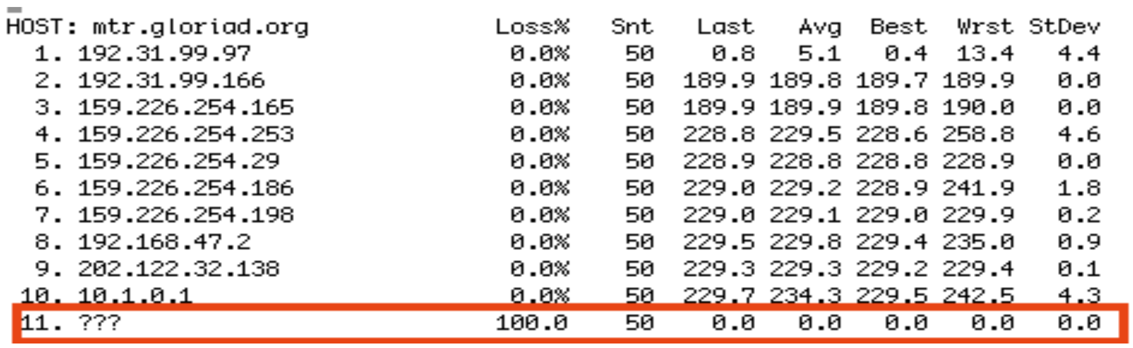
4.3.4. Few observation about the MTR results

In some cases, MTR results show non-increasing rtt values as the hops increase. For example, in the above figure *'best rtt'* of router at hop 9 is 229.2ms while that of router at hop 8 is 229.6ms. Such cases can be explained by the fact that some routers handle error packet reply in software (which is slower, while simple forwarding of packets in hardware is very fast) or routers give lower priority to sending ICMP error packets and so the delay factor is added to the rtt value.

When ICMP is blocked on a router, MTR does not receive any ICMP echo-reply or times exceed reply. In such cases MTR times out after 1000ms and considers the packet to be lost. When MTR receives no reply at a hop it displays “???” sign for that hop. For example, in the MTR run shown in Figure 15, no router beyond hop 10 was discovered. Therefore, it displayed “???” sign for hop 11. In such cases the end host is said to be unreachable, i.e. “*Destination or Target IP is unreachable*”.

Advantages:

1) MTR gives better results than other ICMP based tools. Because it depends on ICMP time exceed replies instead of the ICMP echo-reply messages that are blocked on most routers.
2) It requires no kernel level changes in the OS.
3) It requires no special software to installed on the intermediate routes or at the end host in the network.



```
HOST: mtr.gloriad.org              Loss%   Snt   Last   Avg  Best  Wrst StDev
  1. 192.31.99.97                   0.0%    50    0.8   5.1   0.4  13.4   4.4
  2. 192.31.99.166                  0.0%    50  189.9 189.8 189.7 189.9   0.0
  3. 159.226.254.165                0.0%    50  189.9 189.9 189.8 190.0   0.0
  4. 159.226.254.253                0.0%    50  228.8 229.5 228.6 258.8   4.6
  5. 159.226.254.29                 0.0%    50  228.9 228.8 228.8 228.9   0.0
  6. 159.226.254.186                0.0%    50  229.0 229.2 228.9 241.9   1.8
  7. 159.226.254.198                0.0%    50  229.0 229.1 229.0 229.9   0.2
  8. 192.168.47.2                   0.0%    50  229.5 229.8 229.4 235.0   0.9
  9. 202.122.32.138                 0.0%    50  229.3 229.3 229.2 229.4   0.1
 10. 10.1.0.1                       0.0%    50  229.7 234.3 229.5 242.5   4.3
 11. ???                           100.0    50    0.0   0.0   0.0   0.0   0.0
```

Figure 15. Example of Unreachable hosts in MTR results.

4.3.5. System setup

When an under-performing flow is detected, MTR is used to gather more detailed data about the health of routers/links in the path taken by the flow. MTR is launched in near-real-time (i.e. at the same time when the flow occurs) to both the source and destination IP's of the flow. The system is designed such that MTR results are collected within a maximum time interval of 2 minutes from when the flow is detected as under-performing. Since most of the big flows run for long periods, triggering MTR runs in real-time ensures that the MTR test packets are injected when network conditions are similar to those seen by real traffic.

The MTR runs are launched from a host machine placed next to the router, whose traffic is analyzed by the passive monitoring system. In the current system, MTR runs are launched from a machine that is one hop away from GLORIAD's Chicago router. The Figure 16 shows the system setup. This system design, gives an approximate measure of the end-to-end performance by simultaneously triggering MTR runs to the flow source and destination from the observation point (Chicago router) and combining the two results. The dotted line in Figure 17 shows the true end-to-end measurement path. The solid line represents the measurements made by the current system. With the assumption that the network routing is symmetric, the measurement triggered from source is approximately equal to the one triggered in the opposite direction i.e. from the observation point to the source.

The assumption of symmetric routing is valid with the current Packeteer-based passive monitoring system on GLORIAD. The Packeteer monitoring box gets the retransmits measures only on symmetric flows. Since under-

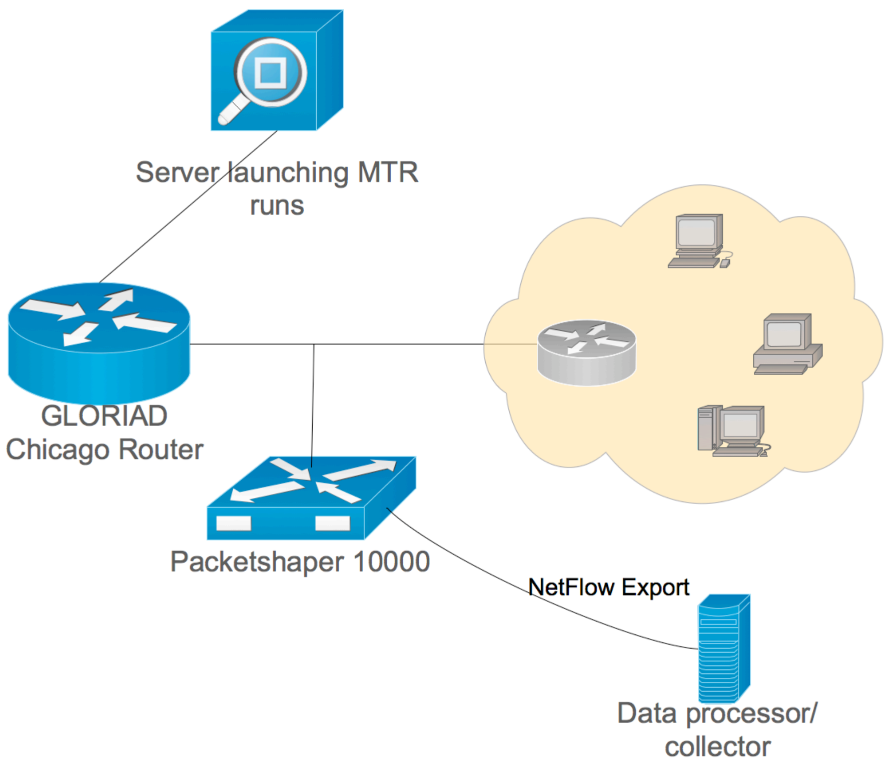


Figure 16 Active and passive monitoring system setup

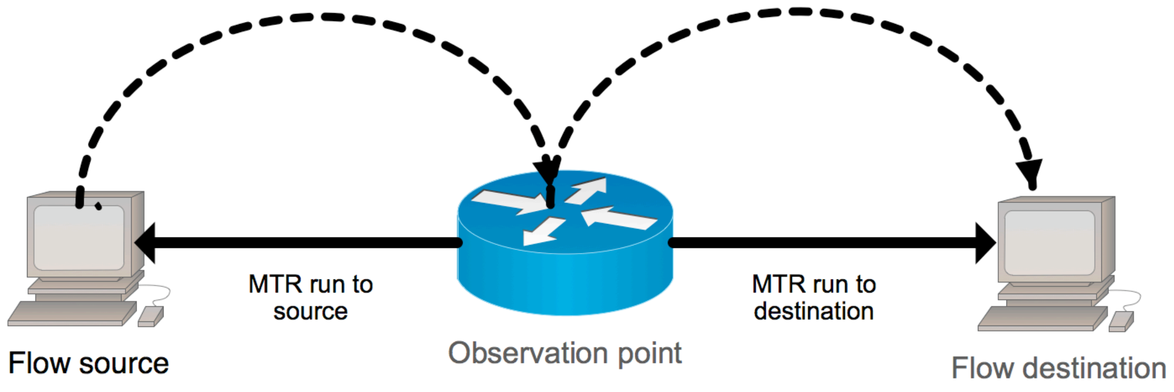


Figure 17. Approximation of end-to-end measurement.

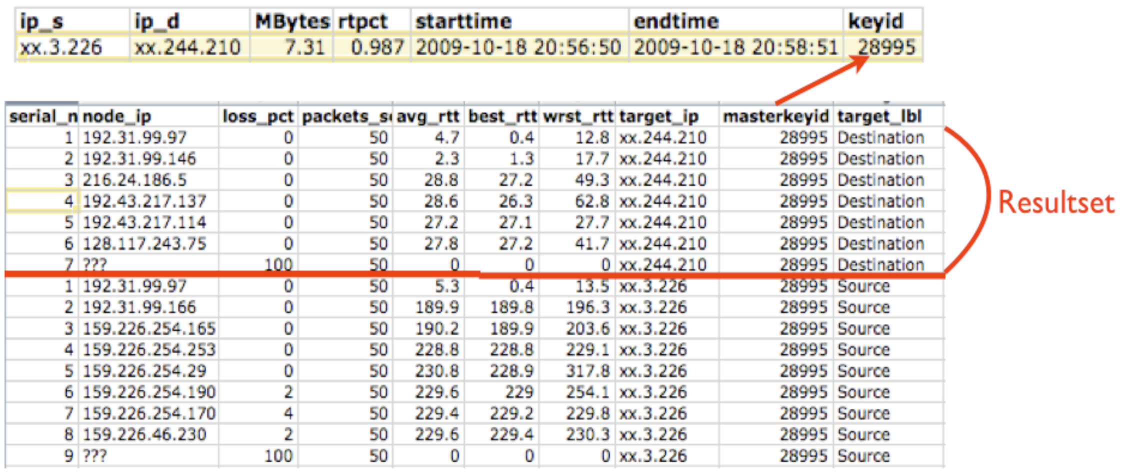
performing flows are detected based on the retransmissions values in the flow data, all such flows are symmetric flows.

This design is a trade off between requiring special permissions at the end host and a simple design requiring access only to the observation points. Collecting data by triggering runs-runs from the source machine to the destination machine would give the best results. But triggering runs-runs from the end host machine will require special access permission that is not preferred due to end user security and privacy concerns. Therefore in this thesis, a good approximation is used by triggering runs-runs from the intermediate router that is guaranteed to be in the path from the source to the destination machine.

4.3.6. Data storage structure

Each MTR run is mapped back to the original flow (flow for which MTR run was triggered) using the flow's '*keyid'*. Each flow has two separate MTR results, one each to source and destination. They are distinguished using '*target_lbl*' attribute. Figure 18 shows an under-performing flow and the corresponding MTR runs to the source and destination IPs'. The attributes of the database table storing MTR results are explained below:

> *target_ip* attribute indicates the destination IP address towards which MTR was launched.
>
> *masterkeyid* is a foreign key that points to the flow keyid.
>
> *(masterkeyid, target_lbl)* pair identifies a single MTR run. We refer to the set of records from a single MTR run as 'resultset'.



ip_s	ip_d	MBytes	rtpct	starttime	endtime	keyid
xx.3.226	xx.244.210	7.31	0.987	2009-10-18 20:56:50	2009-10-18 20:58:51	28995

serial_n	node_ip	loss_pct	packets_s	avg_rtt	best_rtt	wrst_rtt	target_ip	masterkeyid	target_lbl
1	192.31.99.97	0	50	4.7	0.4	12.8	xx.244.210	28995	Destination
2	192.31.99.146	0	50	2.3	1.3	17.7	xx.244.210	28995	Destination
3	216.24.186.5	0	50	28.8	27.2	49.3	xx.244.210	28995	Destination
4	192.43.217.137	0	50	28.6	26.3	62.8	xx.244.210	28995	Destination
5	192.43.217.114	0	50	27.2	27.1	27.7	xx.244.210	28995	Destination
6	128.117.243.75	0	50	27.8	27.2	41.7	xx.244.210	28995	Destination
7	???	100	50	0	0	0	xx.244.210	28995	Destination
1	192.31.99.97	0	50	5.3	0.4	13.5	xx.3.226	28995	Source
2	192.31.99.166	0	50	189.9	189.8	196.3	xx.3.226	28995	Source
3	159.226.254.165	0	50	190.2	189.9	203.6	xx.3.226	28995	Source
4	159.226.254.253	0	50	228.8	228.8	229.1	xx.3.226	28995	Source
5	159.226.254.29	0	50	230.8	228.9	317.8	xx.3.226	28995	Source
6	159.226.254.190	2	50	229.6	229	254.1	xx.3.226	28995	Source
7	159.226.254.170	4	50	229.4	229.2	229.8	xx.3.226	28995	Source
8	159.226.46.230	2	50	229.6	229.4	230.3	xx.3.226	28995	Source
9	???	100	50	0	0	0	xx.3.226	28995	Source

Figure 18. Sample MTR result set for a particular flow as stored in the database

4.4. <u>Complete event flow</u>

The flowchart below (Figure 19) summarizes the flow of events in the network monitoring/date collection system.

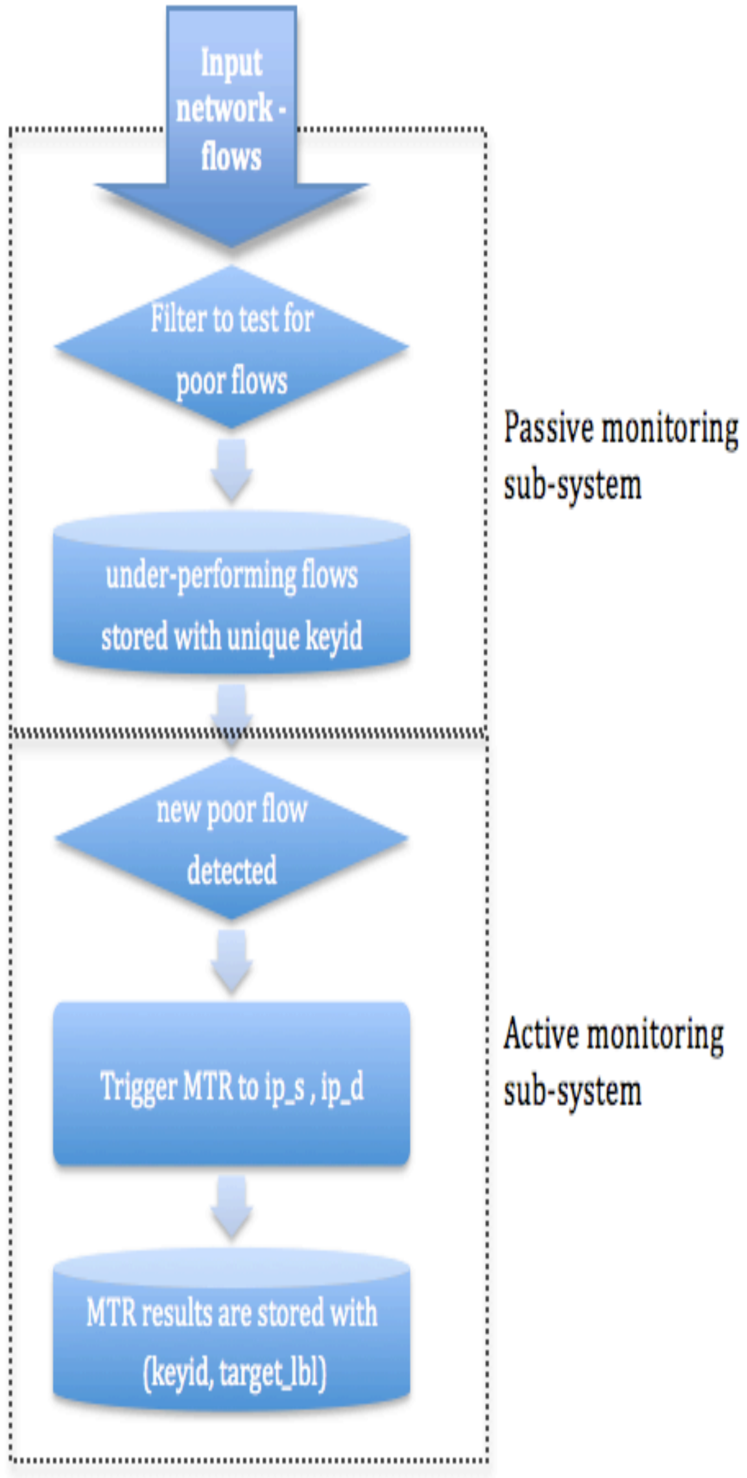

Input network - flows
Filter to test for poor flows
under-performing flows stored with unique keyid
Passive monitoring sub-system
new poor flow detected
Trigger MTR to ip_s , ip_d
MTR results are stored with (keyid, target_lbl)
Active monitoring sub-system

Figure 19. Flow of events in the data collection system.

5. Data exploration and interpretation

5.1. Overview

Collecting the data is just one piece of the problem. The other part is to analyze the data to locate the cause of performance loss in the network. A single MTR resultset gives individual router performance data. When such MTR resultsets, collected over a short time interval are combined it gives a snapshot of the network router topology, along with performance statistics of each individual router. The key idea behind the data analyses is to take advantage of this combined knowledge to localize the cause of performance loss on the network. The following sections provide details about the dataset and the various data visualization methods used to make the complex data more tractable.

5.2. Data interpretation

A single resultset gives performance statistics at each hop of the route from the MTR host machine to a given target IP. The route from the MTR host machine to the given target IP is referred to as a '*path*'.

A *path* can be represented as a graph. Individual routers are the vertices of the graph. While the edges connect routers in consecutive hops. Figure 20 represents a hypothetical *path* to destination 'D_1'.

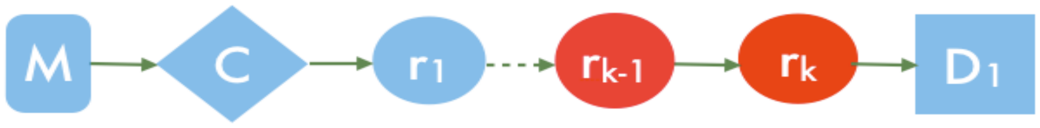


Figure 20. Model graph representation of a single resultset.

Notation

M – represents the MTR host machine.

C – represents the Chicago router (traffic through which is passively monitored).

r_i -- represents an intermediate router in the path.

D_1 – represents the target or destination of the path.

A router is in **'red'** if MTR shows packet loss at the router.

5.2.1. Observations

In the above example (Figure 20) both routers r_{k-1} and r_k show packet loss. The goal of this system is to determine with some confidence the true cause of performance loss in the flow. Just by considering a single path in isolation, the cause of loss cannot be localized i.e. determine if r_{k-1} or r_k or both are the cause of loss.

As discussed in the previous chapter ICMP based tools have some limitations. So the loss shown at any router r_i can be due to r_i itself or due to the loss in the forward or backward transit of the packet. Therefore, it is not sufficient to consider each path in isolation.

5.2.2. Network graphs

Consider MTR resultsets of all the under-performing flows over a short time interval. This combined dataset represents network router topology for tha time interval. This dataset of paths can be represented as a graph similar to the above graph where

Vertices – represent individual routers.

Edges – represent link between routers in consecutive hops.

Leaf nodes – Target IP’s.

These graphs are called ‘***Network graphs’***. Figure 21 shows a sample network graph. Network graphs show individual router behavior cutting across several MTR runs, to different target IP’s. This knowledge can be used to deduce the probable faulty router in the network. For example in

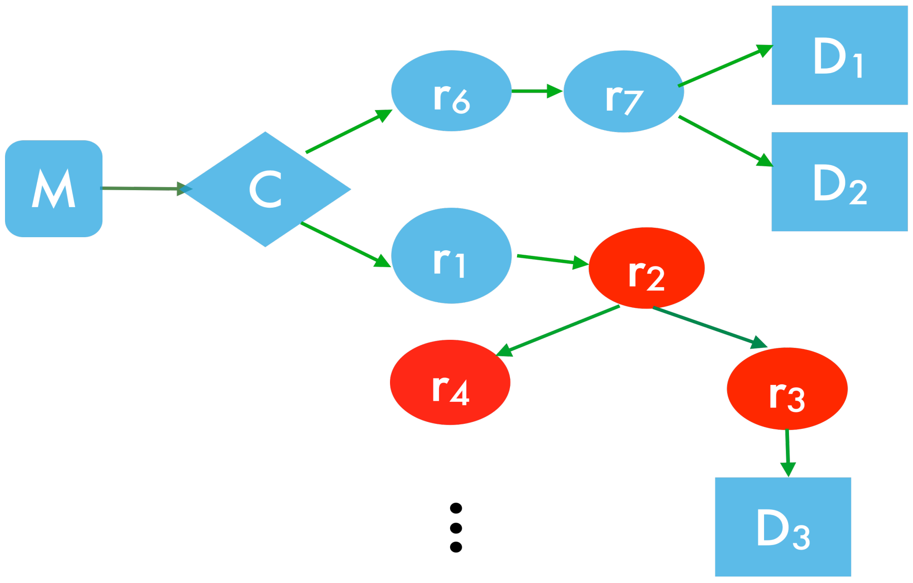


Figure 21. Block diagram of a network graph

Figure 21 routers r_2, r_3, r_4 show packet loss. The true problem node can be any one or all of these nodes. Now the packet loss at r_2 across paths to different destinations can be compared. It can be deduced that r_2 is faulty if packet loss at r_2 is across all the runs via r_2.

Thus, the logic to identify faulty nodes can be designed using the network graphs, which represent the combined knowledge from several MTR resultsets.

5.3. Data visualization

Each time an under-performing flow is detected by the input filter the data collection system triggers MTR runs (ref. chapter 2). The data collected over a period of 90 days consists of 15,284 under-performing flows. This consists of 7,704 unique target IP's. To study the criteria for localizing the cause of performance losses, only a subset of this large dataset is considered.

Initially data collected from under-performing flows from US to the High Energy Physics (HEP) domain in China is considered. The HEP domain IP's are the most frequently occurring IP's among the total 15,284 flows marked as under-performing flows. There are a total of 59 target IP's in this subset. Therefore, a set of 59 target IP's is used in the current analyses. There are 1,337 resultsets to these target IP's, collected over a period of 90 days.

The network graphs are generated using these 1,337 resultsets. The graphs are generated using Perl Graphviz module. Figure 22 shows a basic network graph generated from this data.

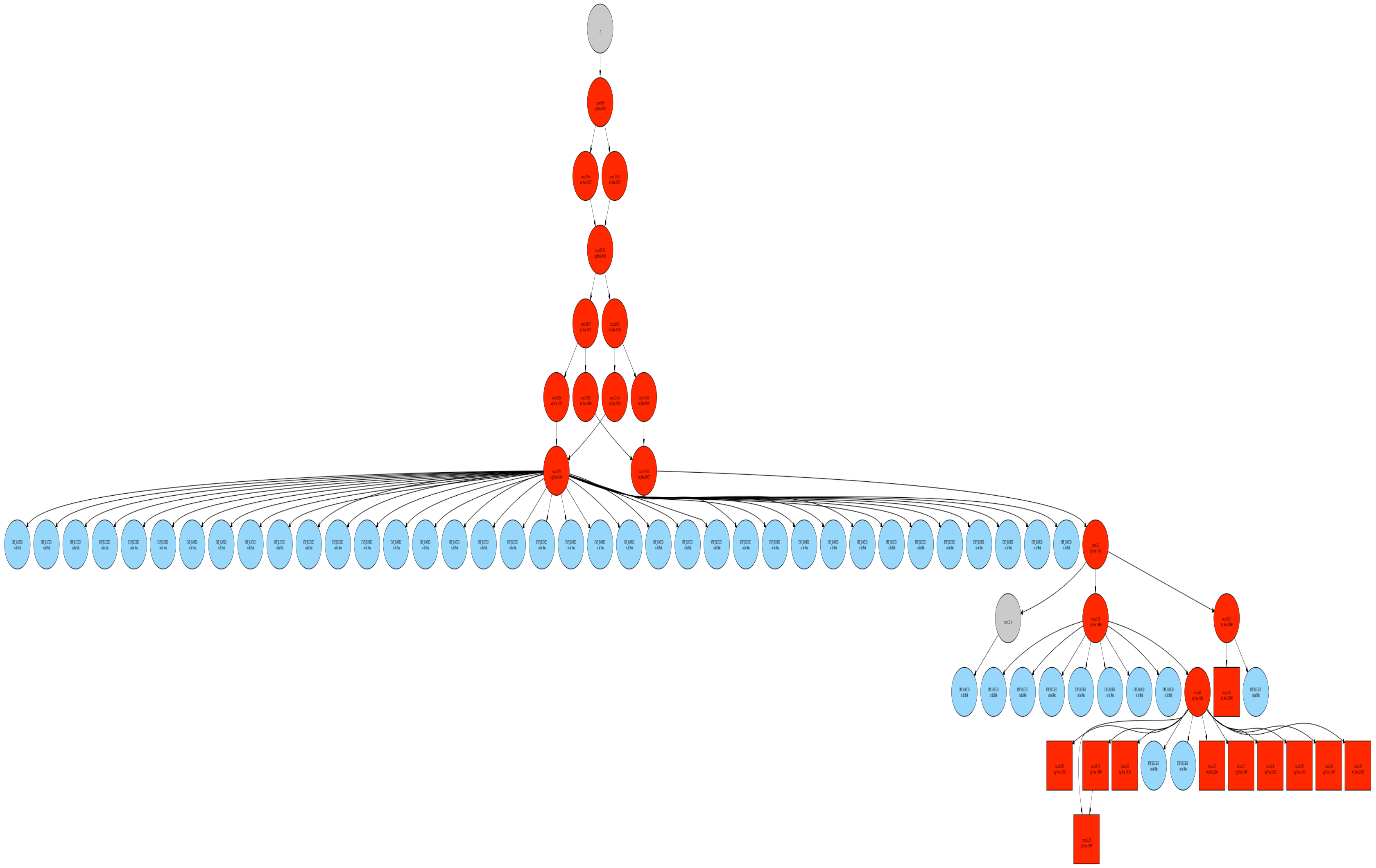


Figure 22. Basic network graph to the 59 HEP target IP’s

Note: All the IP addresses in the graphs are anonymized.

Graph notations

1) Root node is the Chicago router 'C', i.e. the 1st hop of each MTR run.
2) Elliptical nodes represent intermediate routers.
3) Rectangular nodes represent the last hop of the resultset i.e. either a target IP or an unreachable node.
4) Color code:
    - Red – Nodes with non-zero packet loss.
    - Blue - Unreachable nodes.
    - Gray – Nodes with zero packet loss.
5) Labels:
    - First line of each node shows the node IP. In case of unreachable nodes the label is '*ICMP Blocked on the node*'.
    - The following lines show other details dependent on the graph (explained in the following sections).

5.3.1. Various forms of network graphs

To help with the data exploration, several variations of the network graphs are generated. The basic graph representation and notations are the same as explained above. All the graph variations are designed to better understand the router behavior in two cases,

Case 1: Across multiple runs to different target IP's.

Case 2: Across multiple runs to a single target IP.

The main variations of the network graph are explained below,

1) Graphs with list of packet loss at each node.
2) Graphs with mean packet loss at each node.
3) Graphs with 'Clickable nodes'.
4) Graphs with counts of total runs versus the runs seeing packet loss.
5) Graphs with Color-mapping.

Graphs with list of packet loss

In these graphs the node label contains a list of non-zero %loss values. The list is hyphen delimited, with 40 values in each line. This gives a quick indication of the number of times there is a packet loss at the node and what the loss is. Figure 23 is a subtree of the network graph in Figure 22. The first line of node label gives the node IP. The following lines list the non-zero %loss values.

Graphs with mean packet loss

In these graphs the node label contains the mean %loss value. The mean loss values are shown in 2nd line of the node label. Figure 24 and Figure 25 show the sub-trees of the graph in Figure 22 with mean loss values.

Clickable graphs

The above graph variations are suitable to understand the overall router behavior, independent of the specific paths on which the loss is seen. But it is important to see on which runs or resultsets the packet loss is seen. For example if router '*r*' has the loss list 'l_1-l_2-l_3'. Then, it is useful to see if l_1, l_2, l_3 are seen on the path to a single target IP or if the loss is spread across

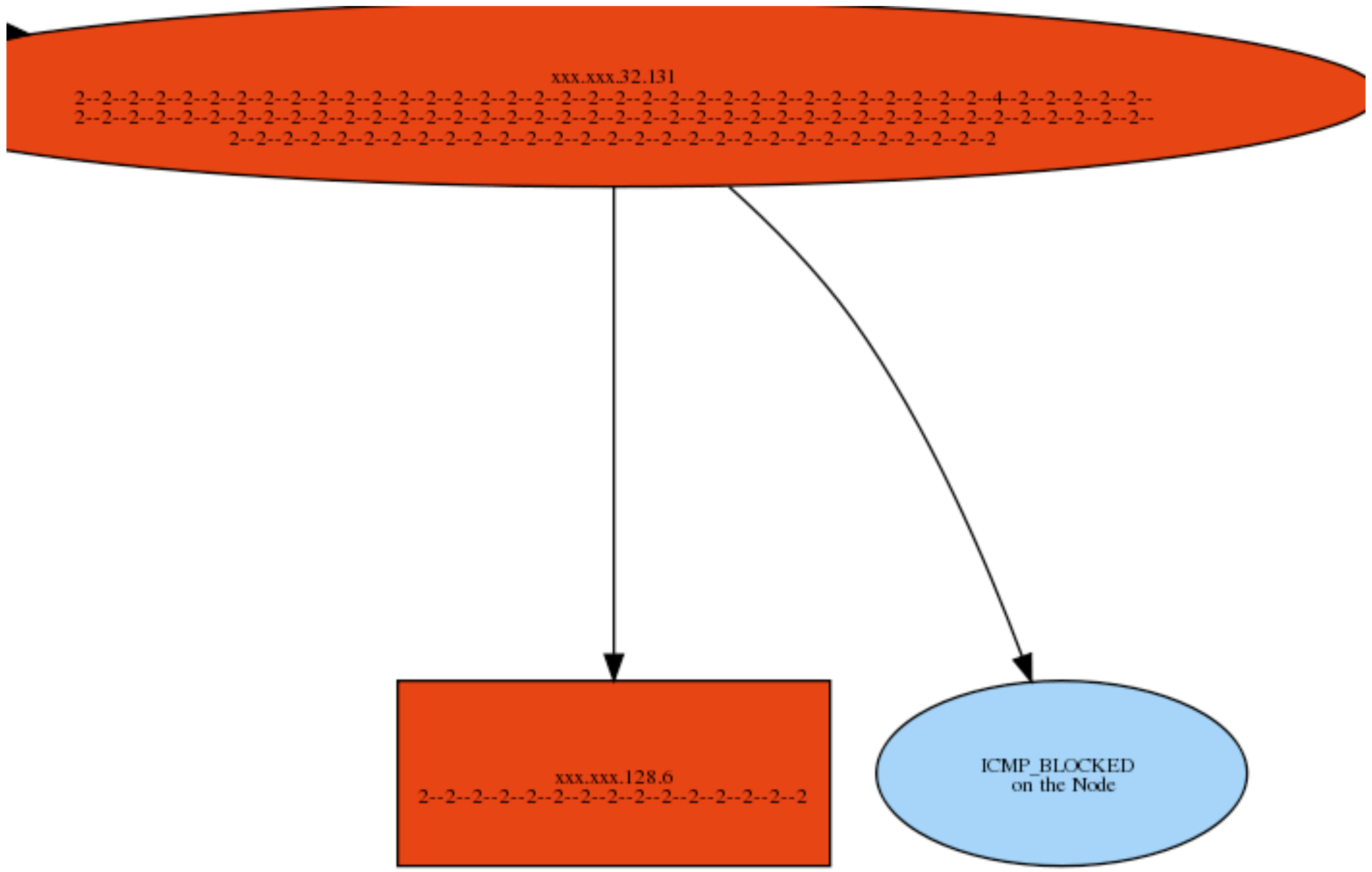


Figure 23. Example 1- Subtree of a network graph with List of %loss values.

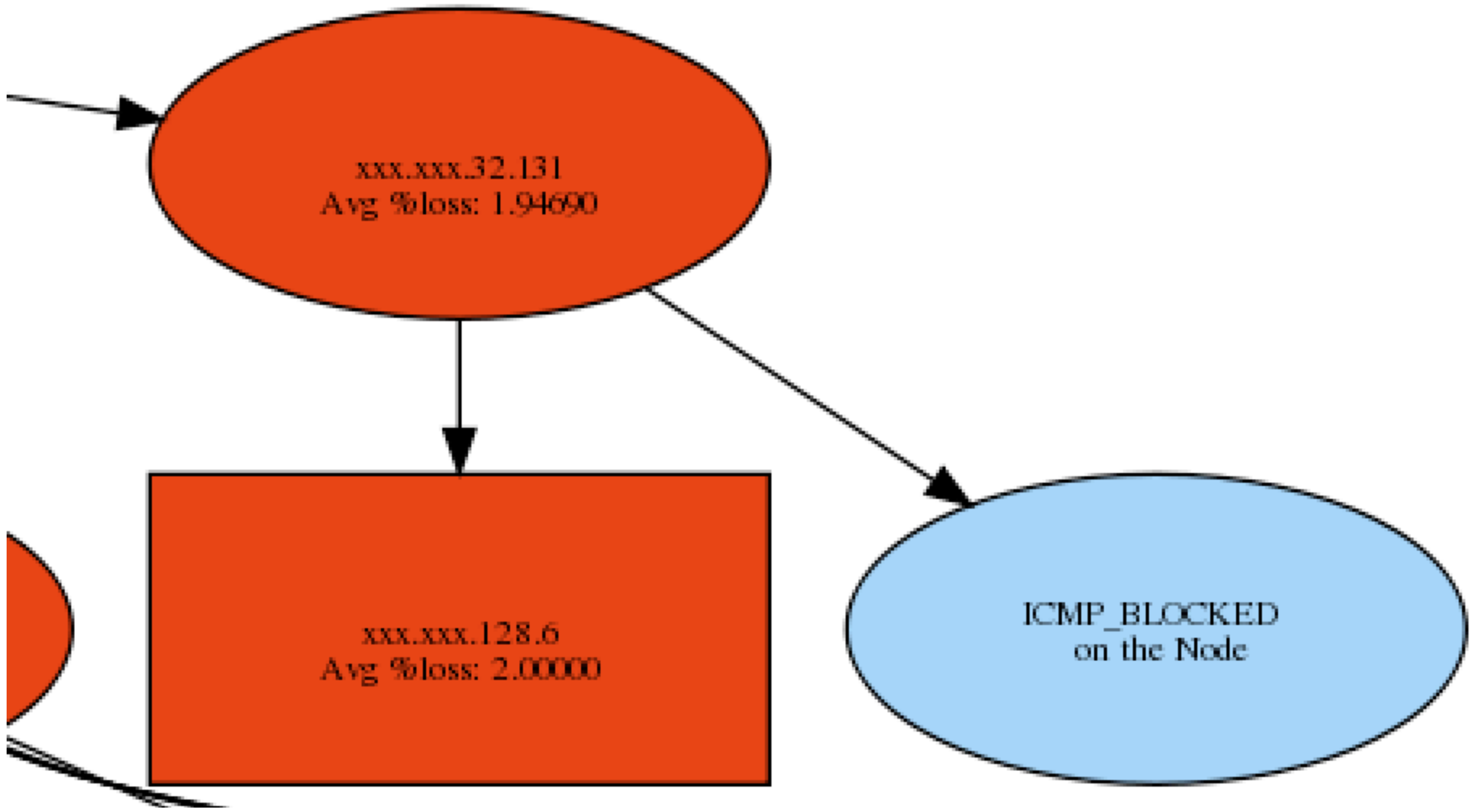


Figure 24. Subtree of network graph with mean %loss.

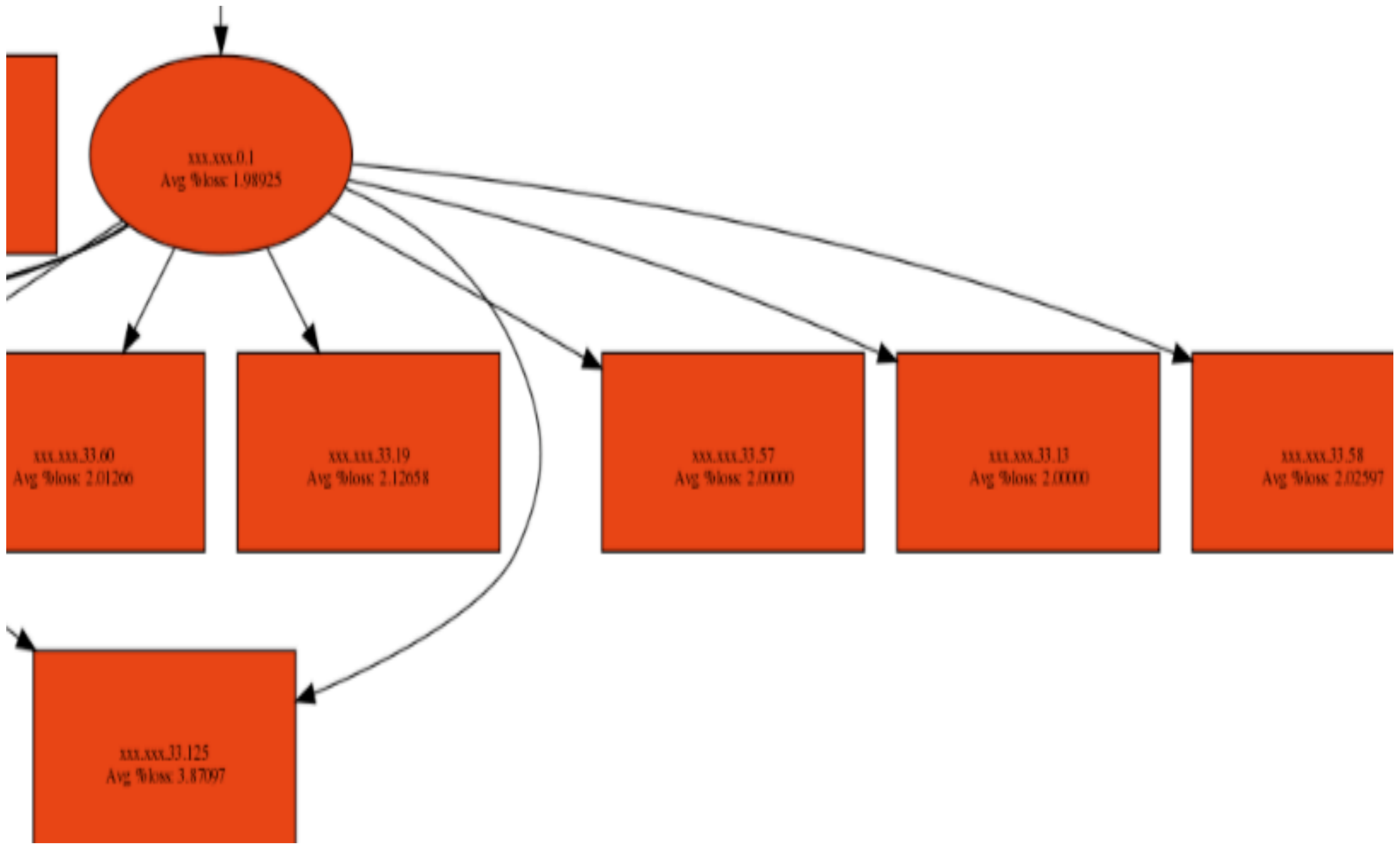


Figure 25. Example 2- Subtree of network graph with mean %loss.

all the paths via router *r*. The clickable graphs are designed to provide this capability.

A clickable graph is a normal network graph with drill down capability. Each node can be clicked to see further details of all the paths through the clicked node. In other words the %loss values on each path through the clicked node can be seen. The results are grouped by the target IP of the path. Therefore, on clicking a node the data of all the paths that pass through the node, grouped by the destination IP of the paths is shown. Consider the model network graph in Figure 26. If r_2 is clicked the tables shown in Figure 27 will be generated. In every table the header row list the nodes in the path from M to the destination. Other rows display %loss values of MTR runs for that path. The first two tables in the figure show runs to target D_3 and the last table shows runs to D_4.

Figure 28 shows a snapshot of data obtained when the node xx.32.131 is clicked in the network graph shown in Figure 24. The header row lists the nodes in the route taken. The column header in 'green' marks the node that

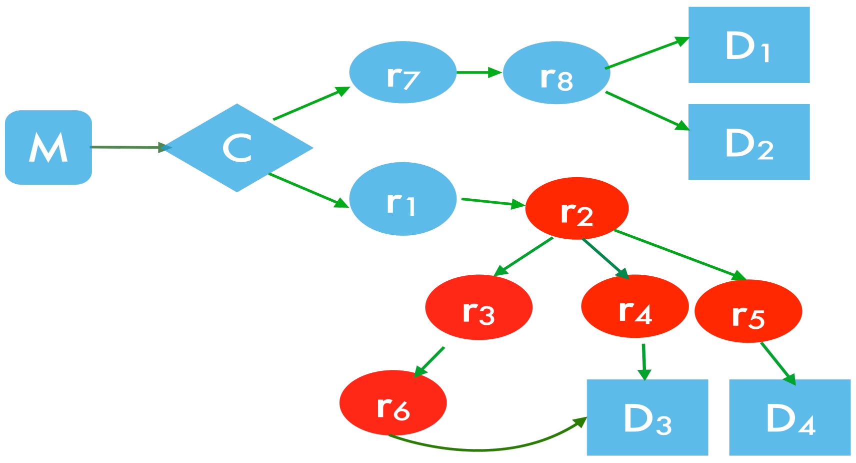


Figure 26. Model network graph to illustrate clickable graphs.

Target IP: D_3

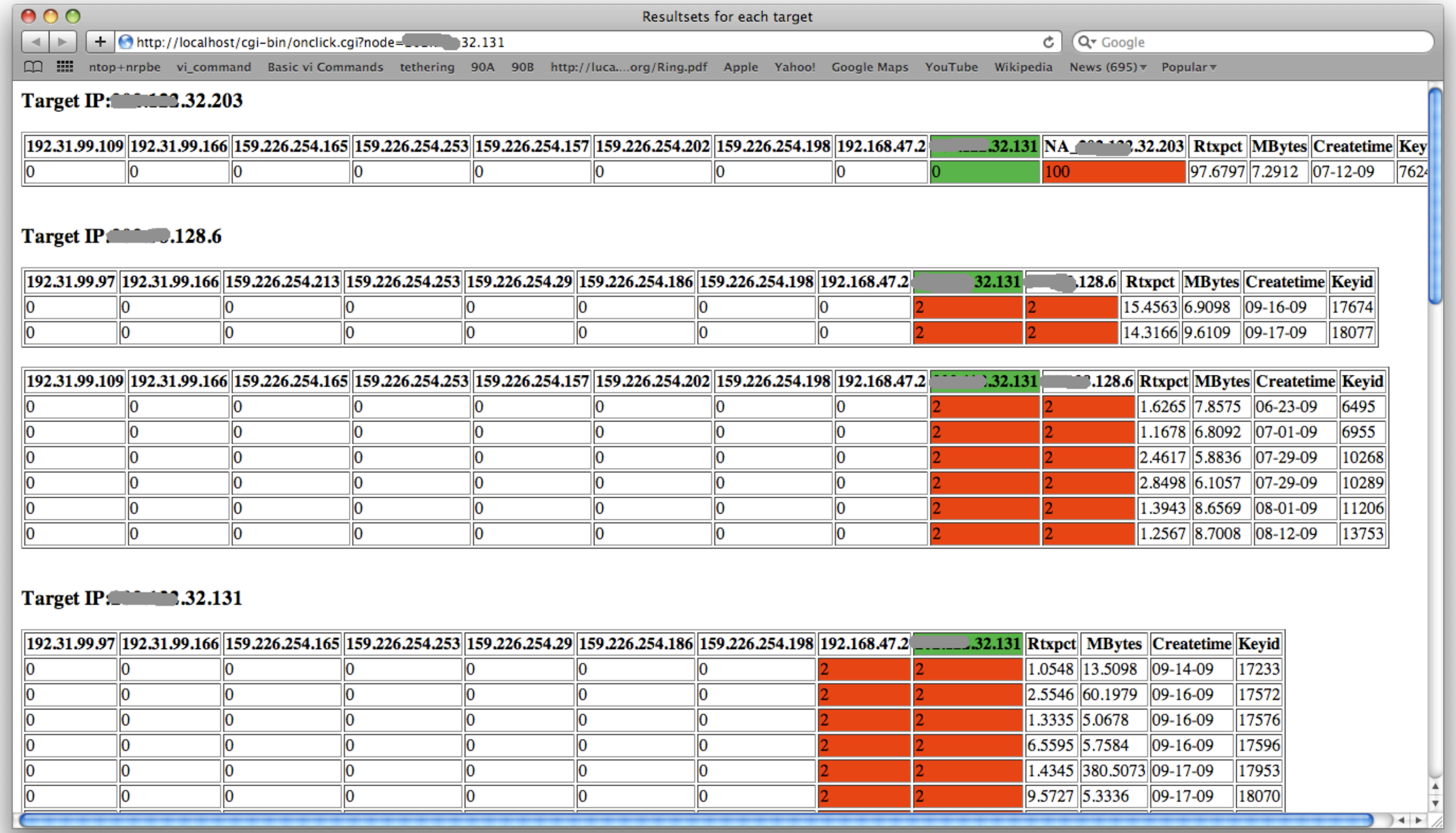
Nodes / Runs	M	C	r_1	r_2	r_3	r_6	D_3
1.	l_1	l_2	l_3	l_4	l_5	l_6	l_7
2.	.	.	.	.	.	.	.
k.	l_{k1}	l_{k2}	l_{k3}	l_{k4}	l_{k5}	l_{k6}	l_{k7}

Nodes / Runs	M	C	r_1	r_2	r_4	D_3
1.	x_1	x_2	x_3	x_4	x_5	x_6
2.	.	.	.	.	.	.
k.	x_{k1}	x_{k2}	x_{k3}	x_{k4}	x_{k5}	x_{k6}

Target IP: D_4

Nodes / Runs	M	C	r_1	r_2	r_5	D_4
1.	y_1	y_2	y_3	y_4	y_5	y_6
2.	.	.	.	.	.	.
k.	y_{k1}	y_{k2}	y_{k3}	y_{k4}	y_{k5}	y_{k6}

Figure 27. Tables displayed on clicking router r_2 in the network graph shown in Figure 26



Resultsets for each target

http://localhost/cgi-bin/onclick.cgi?node=[redacted]32.131

ntop+nrpbe vi_command Basic vi Commands tethering 90A 90B http://luca....org/Ring.pdf Apple Yahoo! Google Maps YouTube Wikipedia News (695) Popular

Target IP:[redacted].32.203

192.31.99.109	192.31.99.166	159.226.254.165	159.226.254.253	159.226.254.157	159.226.254.202	159.226.254.198	192.168.47.2	[redacted].32.131	NA_[redacted].32.203	Rtxpct	MBytes	Createtime	Key
0	0	0	0	0	0	0	0	0	100	97.6797	7.2912	07-12-09	7624

Target IP:[redacted].128.6

192.31.99.97	192.31.99.166	159.226.254.213	159.226.254.253	159.226.254.29	159.226.254.186	159.226.254.198	192.168.47.2	[redacted]32.131	[redacted].128.6	Rtxpct	MBytes	Createtime	Keyid
0	0	0	0	0	0	0	0	2	2	15.4563	6.9098	09-16-09	17674
0	0	0	0	0	0	0	0	2	2	14.3166	9.6109	09-17-09	18077

192.31.99.109	192.31.99.166	159.226.254.165	159.226.254.253	159.226.254.157	159.226.254.202	159.226.254.198	192.168.47.2	[redacted].32.131	[redacted].128.6	Rtxpct	MBytes	Createtime	Keyid
0	0	0	0	0	0	0	0	2	2	1.6265	7.8575	06-23-09	6495
0	0	0	0	0	0	0	0	2	2	1.1678	6.8092	07-01-09	6955
0	0	0	0	0	0	0	0	2	2	2.4617	5.8836	07-29-09	10268
0	0	0	0	0	0	0	0	2	2	2.8498	6.1057	07-29-09	10289
0	0	0	0	0	0	0	0	2	2	1.3943	8.6569	08-01-09	11206
0	0	0	0	0	0	0	0	2	2	1.2567	8.7008	08-12-09	13753

Target IP:[redacted].32.131

192.31.99.97	192.31.99.166	159.226.254.165	159.226.254.253	159.226.254.29	159.226.254.186	159.226.254.198	192.168.47.2	[redacted].32.131	Rtxpct	MBytes	Createtime	Keyid
0	0	0	0	0	0	0	2	2	1.0548	13.5098	09-14-09	17233
0	0	0	0	0	0	0	2	2	2.5546	60.1979	09-16-09	17572
0	0	0	0	0	0	0	2	2	1.3335	5.0678	09-16-09	17576
0	0	0	0	0	0	0	2	2	6.5595	5.7584	09-16-09	17596
0	0	0	0	0	0	0	2	2	1.4345	380.5073	09-17-09	17953
0	0	0	0	0	0	0	2	2	9.5727	5.3336	09-17-09	18070

Figure 28. Snapshot of data generated on clicking node xx.32.131

is clicked. The cells with non-zero loss percentage are shown in 'red'.

Clickable graphs are generated using image maps of the network graphs. An image map consists of a list of coordinates that map different areas of an image to the given hyperlinks. So clicking a particular area on the image maps to the corresponding URL. Graphviz is used to generate image maps with each node mapped to an URL. The URL contains the node IP as a parameter and calls a CGI script. On clicking a node, the CGI script executes and produces the data tables as shown in Figure 28.

Note: There are 11 reachable target IP's in the dataset described above. 91% of the resultsets (1228 of the 1337 resultsets) are to these 11 target IP's. To make the graph more readable from here onwards the unreachable nodes are neglected and only the dataset with 11 target IP's are used.

Graphs with counts of total runs versus the runs seeing packet loss.

These graphs show the count of the number of runs through a node along with the number of runs in which there was a non-zero packet loss at that node. The notations used are explained below.

Graph notations

All the basic graph notations are the same as mentioned before. The new edge and node labels are explained below,

- *Node label* consists of the node IP in the first line followed by the mean %loss in the second line.
- *Edge labels*: Label '*a-b*' for edge between node *A* and *B* denotes,

  a = total # of runs through nodes *A to B.*

  b = # of runs with non-zero %loss among the *'a'* runs from *A to B.*

Graphs with color mapping

To make it easy to eyeball the relative change of a given property across the graph color-mapping is used. The colormap used changes from yellow through orange to red. Figure 30 shows the network graph color-mapped by the mean %loss value. Nodes in yellow have minimum mean loss while the nodes with maximum mean are mapped to red color.

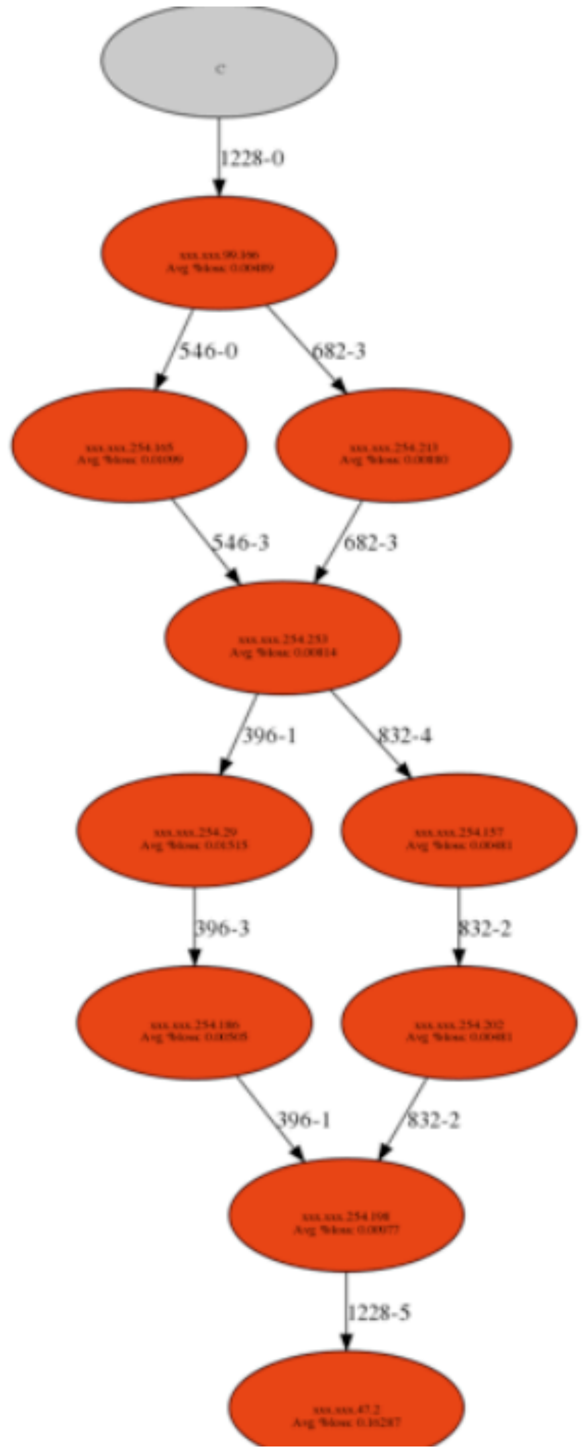


Figure 29. Graph with edge labels to show number of runs versus number of runs with loss

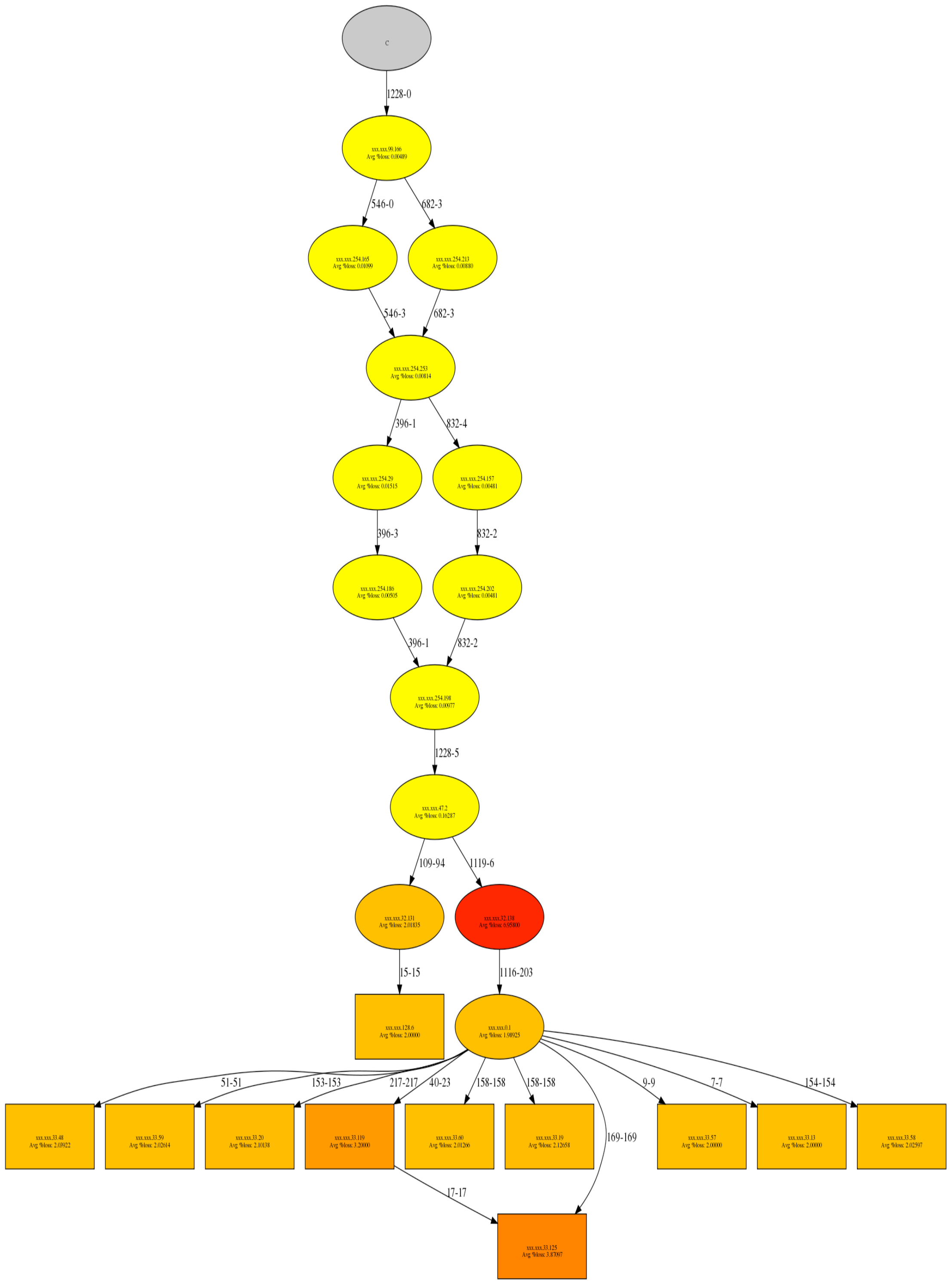


Figure 30. Graph Color-mapped by the mean %loss value of nodes

6. Data analysis and results

6.1. Overview

From the network graphs it can be seen that there are many nodes that show significant packet loss. But locating the true cause of packet loss cannot be based only on the mean packet loss values. As explained in the previous chapter, the packet loss seen at a node may not be due to the node itself. A faulty node at higher levels of the graph can reflect as packet loss in all its descendant nodes (ref. Chapter 4). This chapter explains the methods used to overcome this problem and locate the most probable faulty nodes in the network graphs. First a faulty node is defined and cost functions are designed to identify them. Then pattern analysis algorithms are used to determine the accuracy of the cost functions.

6.2. Faulty nodes

Packet loss at a node can be explained in two ways: Due to the node itself, due to problem node(s) above it in the graph i.e. ancestor nodes. The goal is to identify nodes that are most probable causes of performance loss in the network flows. Thus firstly a problem node or a faulty node is defined.

6.2.1. Defining a faulty node

Suppose a node is faulty. Then it can be said that, it would show high packet loss and packet loss would be independent of the resultset i.e. it will show packet loss irrespective of the target IP of the runs via the node. Consider the network graph in Figure 31. Assume all nodes in the figure

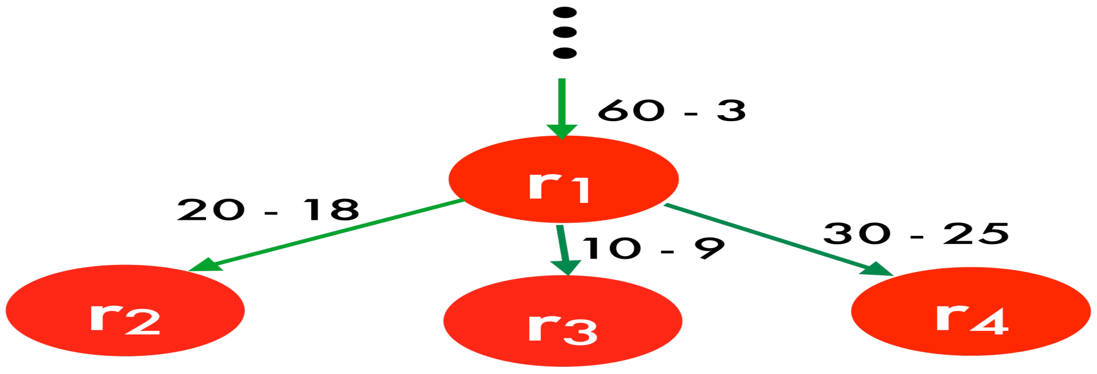


Figure 31. Node with uniform packet loss across all paths.

show high mean packet loss.

Case 1: Node r_1 is faulty.
Then we can explain that r_1 losses packets irrespective of the resultsets i.e. it losses packets 90% of the time on paths to r_2 and r_3 and comparable amount (83%) on the path to r_4.

Case 2: Node r_1 is not faulty.
Then there should be a node in the graph above r_1, which is the reson for losses at r_1 and in other common descendants.

Now considering the opposite scenario where the packet loss distribution is not uniform among all the branches of the node. Figure 32 shows example of such a distribution. It can be said that r_1 is not a faulty node since it doesn't satisfy the second condition of the above definition. Also, it can be deduced that the loss on path r_1 - r_2 is due to the presence of a faulty node above r_1 in path to r_2 but not common to paths to r_3 and r_4.

To summarize, the above two conditions are necessary but not sufficient for a node to be faulty. Therefore if a node doesn't satisfy both the

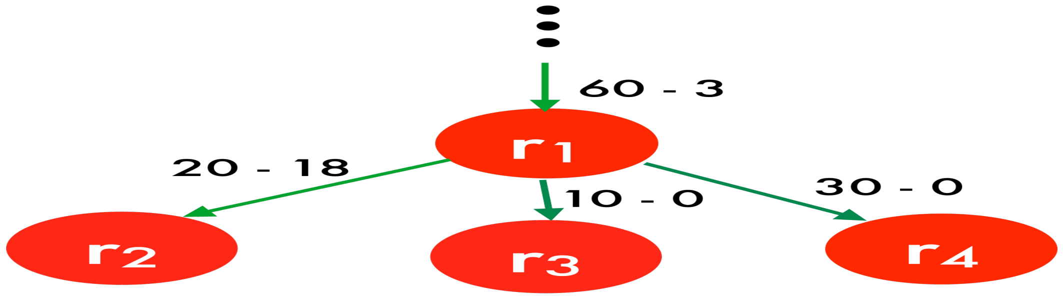


Figure 32. Node with non-uniform packet loss across the paths

conditions then it is not a faulty node.

6.2.2. Locating faulty nodes

The conditions for a node to be faulty are established. Now a cost function is designed such that it will quantify the above-mentioned conditions. Firstly '*probability of loss*' is defined. Using the notation shown in Figure 33,

$$p_i = l_i / t_i$$

Where,

p_i = probability of loss on i^{th} branch of a router r.

t_i = # of runs through r_2 - r_i.

l_i = # of runs with non-zero %loss.

Faulty Node

A node is said to be faulty if its probability of loss is high and is uniformly distributed across all its branches.

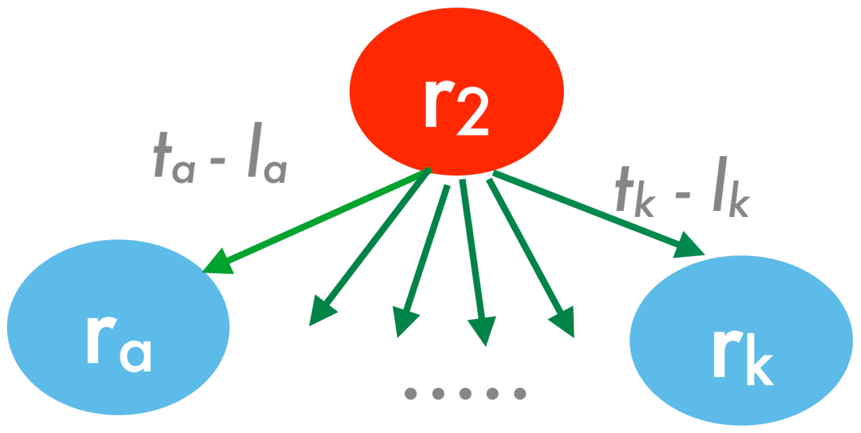


Figure 33. Example to define faulty node

6.3. Cost functions

Three cost functions that quantify the above definition of a faulty node are defined below:

1) Simple multiplicative cost function

$$C(r) = p_1 * p_2 * .... * p_k$$

2) Additive cost function

$$C(r) = (l_1 + .. + l_k) / (t_1 + ... + t_k)$$

3) Standard deviation based cost function

$$C(r) = std.\ dev(p_1, p_2, ..., p_k)$$

Where, f_1 is fraction of times p_i value repeats in the set of probabilities of router *r.*

Observations

1) Simple multiplicative cost function is highly sensitive to outliers. For example in Figure 31 if just one of the branches showed zero loss,

the cost becomes zero even while all the other braches have a probability of 1.

2) Additive cost function is not sensitive to zero value outliers mentioned above. But additive function does not reduce sharply when the probabilities are not uniformly distributed.

3) Standard deviation just captures the deviation in the probability of loss but it does not account for the value of the loss.

6.4. Cost function results

The results of calculating the cost of each node using the three cost functions are shown in the network graphs below.

Notation

1) **Node label** – The value in the 1st line is the cost of the node. The 2nd line gives node IP and the last line gives the mean %loss value.

2) **Colormaps –**The colormap ranges from yellow through orange to red color. The yellow corresponds to the minimum cost while the red corresponds to maximum cost.

Figure 34, Figure 35 and Figure 36 show network graphs with multiplicative, additive and standard deviation based cost function values respectively. The graphs are color-mapped based on the cost function values. The properties of cost functions mentioned above are illustrated using the variations in cost of the node marked by 'red star' symbol. It can be seen that the node has less cost value from Multiplicative cost function than the cost from Additive cost function. This is because the multiplicative cost is highly

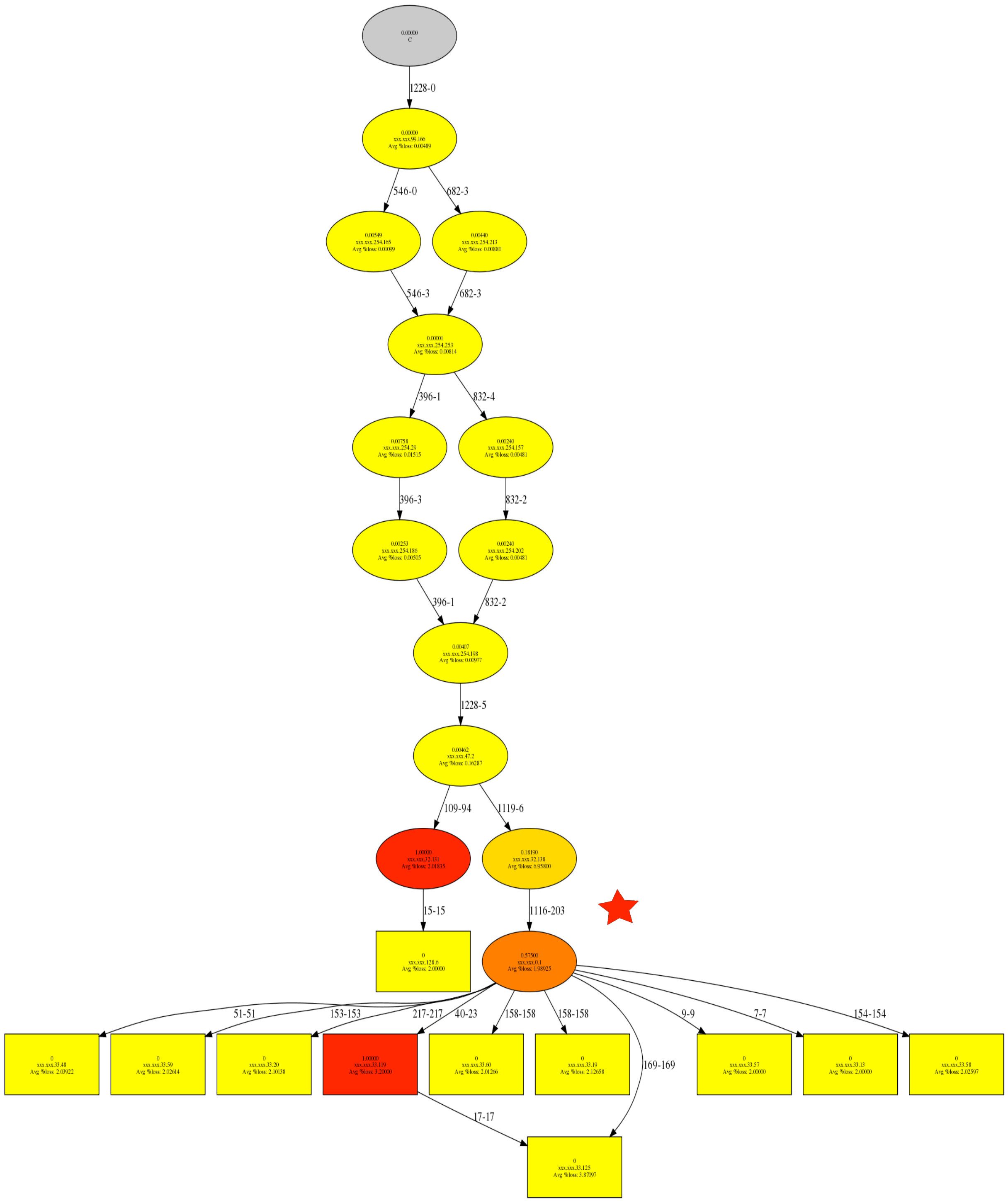


Figure 34 Network graph color-mapped by Multiplicative cost function values

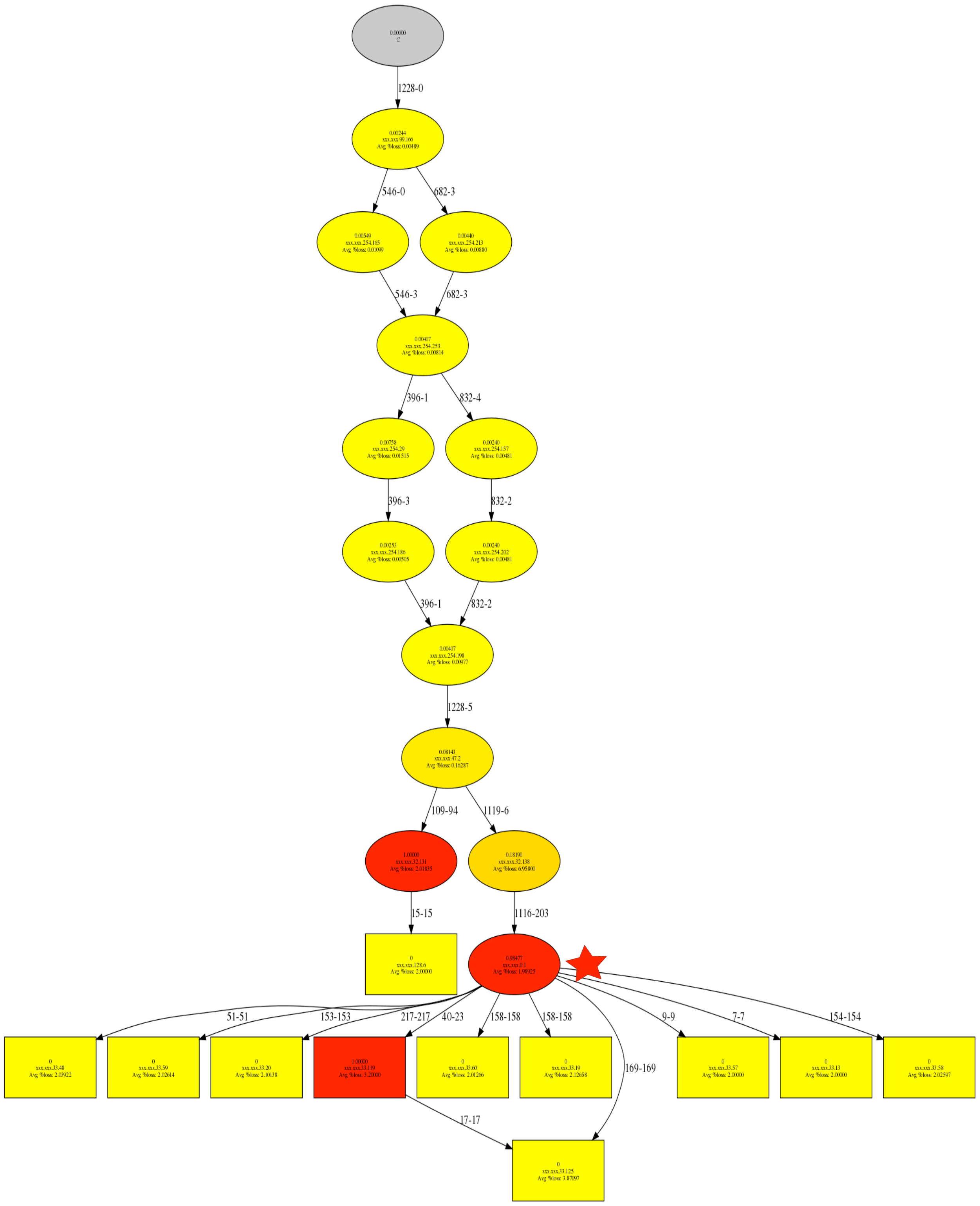


Figure 35 Network graph color-mapped by Additive cost function values

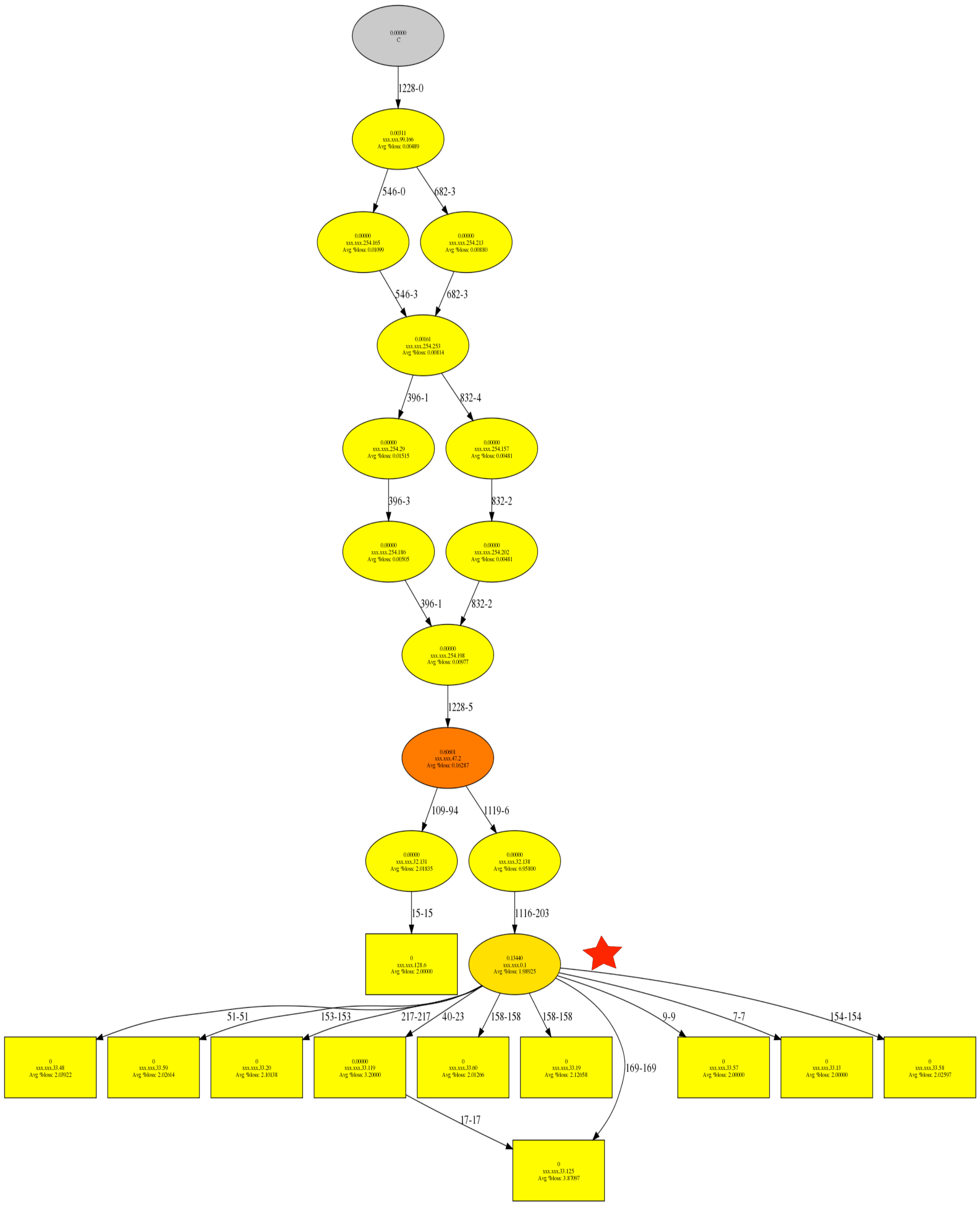


Figure 36 Network graph color-mapped by Standard Deviation based cost function values

sensitive to non-uniformity in loss distribution while the additive cost function is more tolerant to non-uniformity.

As seen above each cost function has its own advantages in different scenarios. In the next section training sets are created and accuracy of the cost functions is computed using Min-distance and KNN pattern analysis algorithms.

6.5. <u>Accuracy of cost functions</u>

Pattern analysis based algorithms are used to test the accuracy of the cost functions. For this purpose the node are divided into three groups. The groups are labeled as -- ‘***red’, ‘orange’*** and ***‘yellow’***, where nodes labeled as ***red*** are the most likely problem nodes, followed by ***orange and yellow*** in decreasing order.

6.5.1. Test setup

A training data set is created where each record in the set consists of the following 5 values,

$$(node\_ip, c1, c2, c3, label)$$

Where,

> c_1, c_2, c_3 – are the costs derived from the multiplicative, additive and standard deviation based cost functions respectively.
>
> ***label*** – the label is given to a node based on the ground truth (prior knowledge).

The dataset contains 20 records per label, making a total of 60 records. The three plots below (Figure 37, Figure 38, Figure 39) show the cost function values grouped by the given label. For multiplicative cost and additive cost function the boundaries are apparent and are marked on the charts.

0 < multiplicative cost <= 0.033 are labeled ***yellow****.*
0.033 < multiplicative cost <= 0.733 are labeled ***orange****.*
0.733 < multiplicative cost <= 1 are labeled ***red****.*

For additive cost function the boundaries are,

0 < additive cost <= 0.133 are labeled ***yellow****.*
0.133 < additive cost <= 0.8 are labeled ***orange****.*
0.8 < additive cost <= 1.0 are labeled ***red****.*

The standard deviation based cost function doesn’t show clear classification boundaries like the above two functions.

6.5.2. Testing accuracy using KNN and Min-distance algorithms

The accuracy of the cost functions is tested using KNN and Min-distance pattern classification algorithm with each cost function as a feature for classification. The above dataset of 60 records is divided randomly into three sets of 20 records each. Each set containing equal distribution of records from each of the three groups. Thus each set contains the following distribution: S_1 (7- **R**, 6 - **O**, 7 - **Y**), S_2 (7- **R**, 7 - **O**, 6 - **Y**), S_3 (6- **R**, 7 - **O**, 7 - **Y**)

Where **R** represents nodes labeled a **Red, O** represents nodes in **Orange** group and **Y** represents the **Yellow** group.

Each test uses two sets as training sets and the third set as the test set to

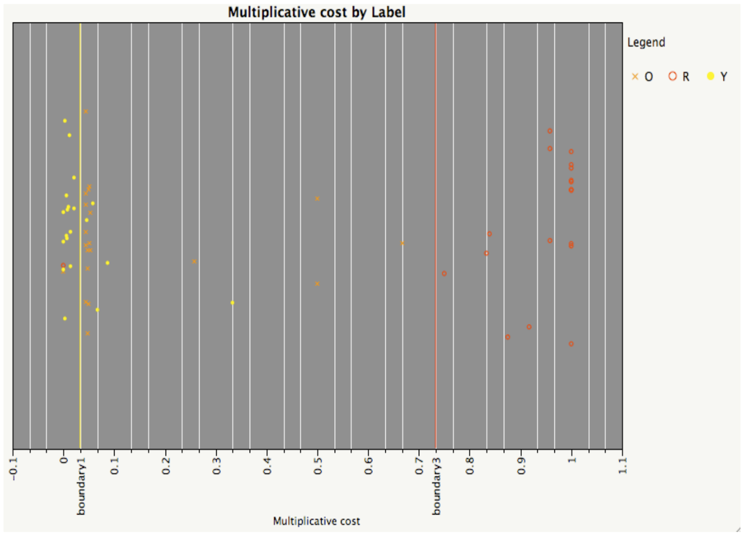


Figure 37. Multiplicative cost grouped by label

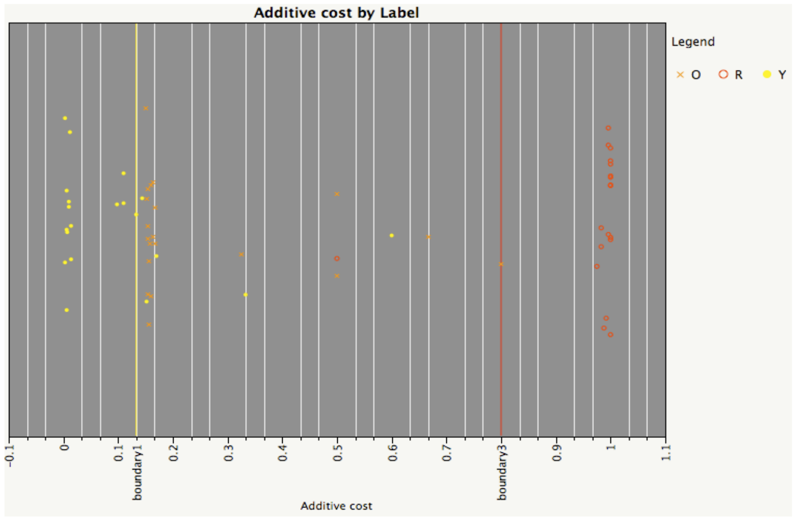


Figure 39. Additive cost grouped by label

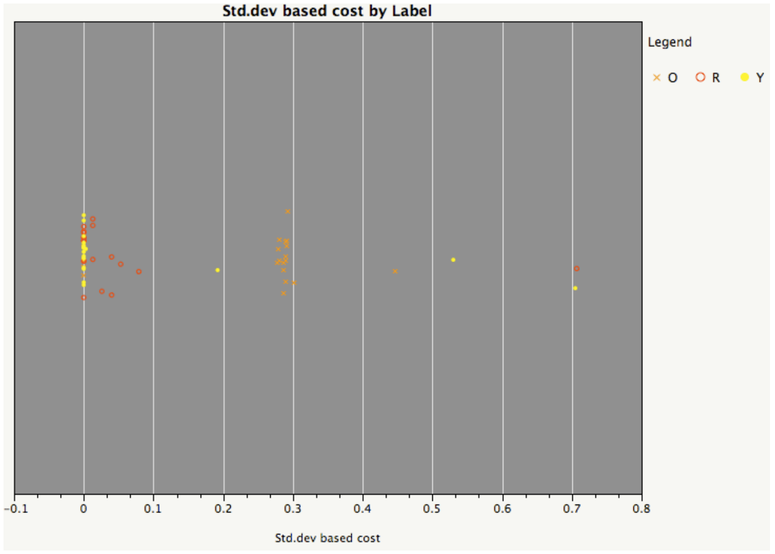


Figure 38. Standard Deviation based cost grouped by label

compute the accuracy. The results are cross-validated using various combinations of training and test sets. The accuracy values obtained are listed in the tables below.

Each column in Table 2 represents the cost function used for classification and each row shows the sets used for test and training respectively. Each cell gives the accuracy value from the test. Each cell is divided into two rows. The row highlighted in blue shows the accuracy obtained by using KNN algorithm with k = 5 and the other row gives the results obtained by using Min-distance algorithm. Table 3 gives the accuracy values of same data with K values of 7, 9 and 11.

Next, the classification accuracy obtained by using a combination of cost functions is tested. Each column in Table 4 represents the combination of cost functions used for classification and each row shows the sets used for test and training respectively. Each cell is divided into three columns with each column showing results for k values of 5, 9 and 11 respectively.

Table 2. Accuracy of cost function using Min-distance and KNN (k = 5) algorithms

Cross validation	**Multiplicative cost**	**Additive cost**	**Std. deviation based cost**
Training (S_1, S_2) Test (S_3)	75%	75%	60%
	95%	100%	70%
Training (S_1, S_3) Test (S_2)	70%	65%	60%
	95%	95%	70%
Training (S_2, S_3) Test (S_1)	70%	75%	65%
	95%	95%	75%

Table 3. Accuracy of cost functions using KNN with K = 7, 9 and 11

Cross validation	**Multiplicative cost**			**Additive cost**			**Std. deviation based cost**		
	k= 7	k=9	k=11	k=7	k=9	k=11	k =7	k=9	k=11
Training (S_1, S_2) Test (S_3)	95%	100%	100%	90%	90%	90%	70%	75%	65%
Training (S_1, S_3) Test (S_2)	95%	100%	100%	90%	90%	90%	70%	65%	65%
Training (S_2, S_3) Test (S_1)	95%	100%	100%	90%	90%	90%	75%	65%	65%

Table 4. Accuracy of combinations of cost functions using KNN with k = 5, 9, 11

Cross valida-tion	**Mcost + Acost**			**Mcost + Scost**			**Acost + Scost**			**Mcost + Acost + Scost**		
	k= 5	**k=9**	**k=11**	**k=5**	**k=9**	**k=11**	**k=5**	**k=9**	**k=11**	**k =5**	**k=9**	**k=11**
Training (S_1, S_2) Test (S_3)	90%	90%	90%	95%	85%	85%	90%	90%	90%	85%	85%	90%
Training (S_1, S_3) Test (S_2)	90%	90%	90%	95%	85%	85%	90%	90%	90%	85%	85%	90%
Training (S_2, S_3) Test (S_1)	90%	90%	90%	95%	85%	85%	90%	90%	90%	85%	85%	90%

Note: In Table 4, Mcost stands for Multiplicative cost, Acost stands for Additive cost and Scost for Std. deviation based cost.

6.5.3. Results

Based on the results obtained from the pattern classification tests it is seen that Simple multiplicative cost function gives best results followed by the additive cost function. Also, it is seen that using cost functions individually gives better results than using a combination of cost functions.

7. Conclusions

This thesis considered the problem of locating causes of performance losses in network flows on large high performance WANs. A system is designed to detect under-performing network flows and locate the most likely problem nodes (routers/links) within the network flow. The main goal of the system was to provide a means to proactively solve end-to-end performance problems in large networks. Thereby improving the end-user application's performance before the user realizes the performance degradation.

Through the current system a basic framework for debugging end-to-end performance problems is developed. Also, it successfully implemented a hybrid data monitoring/collection system and pattern analysis based algorithms to analyze the data and locate problem nodes. It is being test on real data from the GLORIAD network.

7.1. Future work

There are two main areas in the current system that can be further improved to make it more accurate and fault tolerant. Firstly, the current system is purely based on the monitoring information collected from a single point on the network. The active monitoring system implemented from an observation point on the network gives an approximation of end-to-end performance data. In future this can be extended to include probes that launch MTR runs from the end hosts to get more accurate results. Also, MTR runs can be designed to launch from different points on the network and the data collected could be correlated to get more accurate round trip time measurements.

Secondly, other algorithms to localize the faulty nodes in a network graph can be developed. This system implements a simple algorithm based on a single metric (%loss) to locate problem nodes. In future algorithms can be designed using multiple metrics like delay, jitter,etc.

List of references

[1] GLORIAD newsletter,
http://www.gloriad.org/gloriad.newsletter.october.2008.pdf

[2] GLORIAD website, http://www.gloriad.org/

[3] Matthew Mathis, Jerey Semke, Jamshid Mahdavi, and Teunis Ott. The macroscopic behavior of the TCP Congestion Avoidance algorithm. Computer Communications Review, 27(3), July 1997

[4] Data and Computer Communications, Seventh Edition. by William Stallings

[5] NetFlow, http://en.wikipedia.org/wiki/Netflow

[6] Matthew Luckie , Young Hyun , Bradley Huffaker, Traceroute probe method and forward IP path inference, Proceedings of the 8th ACM SIGCOMM conference on Internet measurement, October 20-22, 2008, Vouliagmeni, Greece

[7] Trace route, http://www.freesoft.org/CIE/Topics/54.htm

[8] ICMP rate limiting, IEPM group Stanford,
http://www.iepm.slac.stanford.edu/monitoring/limit/limiting.html

[9] Throughput versus loss, SLAC center, Stanford,
http://www.slac.stanford.edu/comp/net/wan-mon/thru-vs-loss.html

[10] Vern Paxson, “Automated packet trace analysis of TCP implementations”, Proceedings of the ACM SIGCOMM '97 conference on Applications, technologies, architectures, and protocols for computer communication, p.167-179, September 14-18, 1997, Cannes, France

[11] V. Paxson, “End-to-End Internet Packet Dynamics,” *Proceedings of SIGCOMM '97*, September 1997.

[12] M. Fomenkov, k. claffy, E. Katz-Bassett, R. Beverly, B. Cox, and P. Haga, ``The Workshop on Active Internet Measurements (AIMS) Report,'', in Internet Statistics and Metrics Analysis (ISMA). 12-13 Feb 2009, CAIDA, San Diego Supercomputer Center, UC San Diego.

[13] Matthew Luckie , Young Hyun , Bradley Huffaker, Traceroute probe method and forward IP path inference, Proceedings of the 8th ACM SIGCOMM conference on Internet measurement, October 20-22, 2008, Vouliagmeni, Greece

[14] Paramvir Bahl , Ranveer Chandra , Albert Greenberg , Srikanth Kandula , David A. Maltz , Ming Zhang, Towards highly reliable enterprise network services via inference of multi-level dependencies, ACM SIGCOMM Computer Communication Review, v.37 n.4, October 2007

[15] Wenli Liu*, Raouf Boutaba,” pMeasure: A peer-to-peer measurement infrastructure for the internet”, Computer Communications 29 (2006) 1665–1674

[16] Guest Editorial,” Special issue: monitoring and measurements of IP networks”, Computer Communications 29 (2006) 1561–1563

[17] Vern Paxson, End-to-end routing behavior in the Internet, IEEE/ACM Transactions on Networking (TON), v.5 n.5, p.601-615, Oct. 1997

[18] PingER (Ping End to end Reporting), http://www.iepm.slac.stanford.edu/pinger/

[19] The Cooperative Association for Internet Data Analysis,
http://www.caida.org/home/

[20] tcpdump. http://www.tcpdump.org

[21] Self-Configuring Network Monitor Project,
http://acs.lbl.gov/NetMon/Self-Config.html

[22] D. Agarwal, J. M. Gonzalez, G. Jin, and B. Tierney. An infrastructure for passive network monitoring of application data streams. In Proc. Passive and Active Measurement Workshop, 2003.

Vita

Harika Tandra was born in Hyderabad, India on December 4th 1984. She completed her high school from Dayanand Anglo Vedic Public School, Hyderabad. She graduated with a Bachelors degree in Computer Science from the Osmania University, Hyderabad in 2007. In Fall 2007 she joined the department of Electrical Engineering and Computer Science at University of Tennessee, Knoxville to pursue Masters degree in Computer Engineering. During her graduate studies she joined the *Advanced Imaging and Collaborative Processing Laboratory* of Dr.Hairong Qi to work in the field of computer networks. In Summer 2008, she joined the *Global Ring Network for Advanced Applications Development* (GLORIAD) project as a Graduate Research Assistant, where she has been working in the area of network performance analysis, network monitoring and data visualization. She will be graduating with a Master of Science Degree in Computer Engineering in December 2009.