A Computational Framework

for Quality of Service Measurement, Visualization and Prediction

in Mission Critical Communication Networks

by

Muhammet Burhan Senturk

A Thesis Presented in Partial Fulfillment
of the Requirement for the Degree
Master of Science

Approved July 2014 by the
Graduate Supervisory Committee:

Jing Li, Chair
Mustafa Gokce Baydogan
Teresa Wu

ARIZONA STATE UNIVERSITY

August 2014

ABSTRACT

Network traffic analysis by means of Quality of Service (QoS) is a popular research and development area among researchers for a long time. It is becoming even more relevant recently due to ever increasing use of the Internet and other public and private communication networks. Fast and precise QoS analysis is a vital task in mission-critical communication networks (MCCNs), where providing a certain level of QoS is essential for national security, safety or economic vitality. In this thesis, the details of all aspects of a comprehensive computational framework for QoS analysis in MCCNs are provided. There are three main QoS analysis tasks in MCCNs; QoS measurement, QoS visualization and QoS prediction. Definitions of these tasks are provided and for each of those, complete solutions are suggested either by referring to an existing work or providing novel methods.

A scalable and accurate passive one-way QoS measurement algorithm is proposed. It is shown that accurate QoS measurements are possible using network flow data.

Requirements of a good QoS visualization platform are listed. Implementations of the capabilities of a complete visualization platform are presented.

Steps of QoS prediction task in MCCNs are defined. The details of feature selection, class balancing through sampling and assessing classification algorithms for this task are outlined. Moreover, a novel tree based logistic regression method for knowledge discovery is introduced. Developed prediction framework is capable of making very accurate packet level QoS predictions and giving valuable insights to network administrators.

To my family.

ACKNOWLEDGEMENTS

First, I would like to express my sincere gratitude to my advisor and committee chair Dr. Jing Li for her invaluable guidance and support. Working with her has been a true pleasure. Without her great advice, insight and creative ideas, this work would not have been completed.

I would like to thank Dr. Mustafa Gokce Baydogan for his invaluable guidance, trust, and assistance throughout this project. He has been a great teacher, mentor and friend to me. All my studies in Arizona State University, including this thesis, would not have been possible without him.

I would also like to thank my committee member Dr. Teresa Wu for her valuable comments and suggestions.

Special thanks to ASURE / SDSI team: Prof. Werner J. A. Dahm, Dr. Kevin Buell, Mark Giddings, Dr. Ali Elahi, Hema Krishnamurthy for their intellectual and financial support.

I sincerely thank to my family; my parents and sister, for their continuous support and blessings for the work I choose to pursue.

I would like to express my appreciation to my friends Mehmet Yigit Yildirim, Ali Mancar and Orcun Elitez for their endless help, support and brilliant ideas.

I am also grateful to my Turkish friends who live in Arizona for being a family to me far away from home. I would never forget their help and support.

Finally, I would like to express my deep gratitude to my dearest wife, friend and biggest supporter Ayse Gundogdu Senturk. Whatever I achieved in the last couple of years, it is all because of her. I always feel her boundless love, understanding and patience, no matter how far apart we have been. Thank you for all your support and encouragement that made me complete this thesis.

TABLE OF CONTENTS

CHAPTER Page

LIST OF TABLES

LIST OF FIGURES

Chapter 1

INTRODUCTION

1.1 Background and Objective

Recent years have witnessed a significant increase in Internet services and use of private, academic or government communication networks. Network performance analysis using network traffic data has become an important research area. One critical dimension of network performance is Quality of Service (QoS), defined as network performance measured in terms of some parameters that reflect the service performance perceived by the user, such as throughput, packet loss, latency/delay, and jitter [1], [2]. QoS analysis helps to characterize network health and behavior, therefore supports many network management functions like anomaly detection, root cause diagnosis, and damage prevention.

While QoS analysis is important in any communication network, it is particularly vital in mission-critical communication networks (MCCNs). An MCCN refers to a communication network whose failure may put life or livelihood at risk. Examples of MCCNs include the communication networks used in military operations, law enforcement, emergency, crisis intervention, and healthcare domains. MCCNs demand reliable and timely communication between dispersed, diverse, and usually mobile agents hence enable exchange of mission-critical information that is essential for national security, safety, or economic vitality. QoS analysis is vital for MCCNs in order to detect, fix, or prevent any performance degradation in the networks. Potential causes of QoS problems in MCCNs could be hardware failure, traffic congestion, misconfiguration, environmental interference, or malicious intrusion.

The objective of this thesis is to develop computational and statistical methods

to address three important tasks in QoS analysis of MCCNs:

QoS measurement aiming at tracking network packets from senders to receivers in order to detect QoS problems such as packet loss and delay.

QoS visualization aiming at providing visual analytics to network administrators to help identify QoS problems.

QoS prediction aiming at predicting QoS using network traffic data to help planning or failure prevention.

QoS measurement and prediction draw more attention from research community whereas QoS visualization plays a complementary role for the other two and has great practical value. QoS measurement and visualization together give network administrators valuable insight into the general network health and status. QoS prediction provides the advantage of taking action proactively to prevent any network problem.

1.2 Literature Review

This section reviews the existing work in QoS measurement, visualization, and prediction in general communication networks, not just MCCNs.

1.2.1 QoS Measurement and Visualization

There are two general approaches in QoS measurement: active measurement and passive measurement. Active measurement injects test traffic into the network and measures the QoS of the network in transmitting the traffic data. The main advantage of active measurement is its being controllable. That is, experiments can be performed at any time and with different patterns of interest. The downside is that it

does not reflect the true QoS experienced by a user; also, injection of test traffic generates additional load on network links and nodes, which may affect the measurement significantly.

Passive measurement utilizes observational traffic data collected at so-called data collectors of the network. This thesis focuses on passive measurement. There are two primary types of data used for passive measurement: packet data and flow data.

A packet is the smallest unit of message sent from a sender to a receiver in a network. A packet includes a header and content. The header contains transmission related information such as source and destination addresses, ports, protocol, packet length, and packet version [3]. The content contains the true message such as text, audio, video, or connection request. The header is usually much smaller in size than the content in a packet. There are commonly used libraries, such as tcpdump/libpcap [4], to decode packets for subsequent analysis after the packets are captured at data collectors.

Another type of data is flow data. A flow is a unidirectional data transmission between a source X and a destination Y. X and Y are defined by their IP addresses, their port numbers, and the transport protocol of both (this five tuple defines endpoints of the transmission and the direction). A flow stores the aggregate information about some number of packets observed in a specific time frame that share a number of properties (i.e. five tuple). Nowadays, many network operators have deployed monitoring infrastructures to collect NetFlow [5] or IPFIX [6] flow records [7]. Flow data provides a picture of traffic flow and traffic volume in a network in a scalable manner since the information is aggregated over multiple packets and the packet contents are not stored. Consequently, network flow data is preferred by many applications, including ours.

Measuring QoS using the packet data is a well-studied problem [8] [9] [10] [11] but

the idea of using flow data to measure QoS is relatively new. The use of flow data for measuring one-way loss is explored by [12]. Their main motivation is to estimate one-way loss without the need of any information related to data collectors in a scalable manner. To enable scalability, sampled flow level statistics are used. Similarly, [13] also tries to measure one-way loss from IPFIX records. They aggregate the flow records that match the same key and are in temporal proximity and calculate one-way loss based on this aggregated information. [14] considers the problem of obtaining delay measurements for each flow observed on the network. Basically, a flow record has two timestamps corresponding to the first packet and last packet of the flow. If the same flow is observed in two data collectors, they generate two delay samples by simply differencing the start and end times from each data collector. These delay samples obtained from multiple flows are refined to estimate the delay.

QoS visualization is usually a subsequent advanced capability following QoS measurement, with a purpose of facilitating understanding the QoS measurement results through visual analytics. Many existing traffic analysis software packages provide visualization capabilities, such as ntop [15], argus [16], wireshark [17], and nfsen [18].

1.2.2 QoS Prediction

There is limited research in literature about QoS prediction. Main focus of existing QoS prediction research is web service selection and dynamic traffic control. The web service selection can be defined as selecting the best set of services required by an application from a pool of available and probably distributed services in order to make the composite by considering QoS, availability and users' constrains [19]. In [20], [21], [22] and [23], sole purpose of the prediction is web service selection and composition. [24] presents a QoS prediction framework to be used for congestion avoidance and proactive error recovery. Similarly, in [25] and [26], the aim of QoS prediction is to

reduce the traffic rate or adjust delays between packets when a congestion is predicted and adjust retransmission rate for UDP traffic based on packet loss rate predictions in order for packet loss recovery. [27] tries to help applications to better react to congestion by means of QoS prediction. Besides web service selection and dynamic traffic control, QoS prediction is useful for assisting to provide a certain level of QoS [28] or monitoring the network behavior [29].

Regardless of the purpose, almost all existing QoS prediction literature make use of historical QoS data to make QoS predictions. It is shown by many researchers that latency and packet loss statistics are correlated and one can be used to predict the other [24], [25], [26], [30], [31]. In [24], authors develop an empirically derived formula to predict end-to-end packet loss as a function of bandwidth, delay variation, and trend. [25] generates statistics from historical delay and packet loss rate data and uses them as predictors of packet loss rate in a sparse basis prediction model. [32] proposes a new method, Vivaldi, that predicts delay using only historical delay data.

Markov models are widely used in QoS modeling and prediction. [30] and [33] models internet traffic behavior using hidden Markov model (HMM). Similarly, [29] employs HMM to model packet loss and delay behavior in wireless networks and predicts packet loss and delay statistics. Moreover, [34] tries to predict service degradations based on past degradations using HMM where a service degradation is defined as deviations in delay between two end points in the network. In [20] semi-Markov model is used for QoS prediction, where historical QoS data are the input of the model. Gilbert model [35], which is a two-state Markov model, is particularly popular in QoS prediction literature, since it can describe bursty behavior of QoS well [36], [27], [26].

There also exists a limited number of published works on QoS prediction that

do not require historical QoS data for prediction. [21], for example, does not collect historical QoS data, however latency is measured real-time and predictions are made based on the instantaneous QoS information of the nodes in the same cluster. [37] generates universal indicators of traffic by which loss rate can be predicted. Traffic indicators are a set of statistics derived from collected network traffic such as mean, variance or peak value of number of bytes arrived to a network component over a certain period of time. Then, the authors use a single hidden layer neural network with a sigmoid activation function to predict packet loss rate where the input of the model is the universal indicators of traffic.

1.3 Challenges of QoS Analysis in MCCNs

Measurement, visualization and prediction of QoS is already a challenging task, in MCCNs it presents even more challenges where near-perfect QoS has to be provided [38]. The number of communicating nodes in MCCNs may vary from a few nodes to a few hundred or even more. Depending on the communication frequency, such networks may generate massive amounts of data. Processing this data alone is a very challenging task. It may be possible to reduce the data size while sacrificing the QoS analysis accuracy, but in MCCNs accuracy is essential. Moreover, the results should be presented in a timely fashion. Therefore, accuracy and speed are two main aspects of QoS analysis in MCCNs that create extra challenges.

1.3.1 QoS Measurement and Visualization

In MCCNs, since each packet may contain critical information, tracking each packet and measuring its QoS such as packet loss and delay is vital. This requires processing of large amounts of packet data, which can be slow mainly because of disk performance (disk I/O bandwidth) [39]. One may consider using flow data,

which has a much smaller size than the packet data for QoS measurement in order to avoid the disk I/O bandwidth bottleneck. However, accurate tracking of individual packets is not possible when flows are used, because, flows are usually aggregated packet data. Also, flows do not have unique identifiers, leading to low accuracy in QoS measurement. A potential solution to improve the accuracy is to include packet content, served as a unique identifier, in the QoS measurement algorithm. One important concern for including packet content is security. MCCN packet content usually contains security sensitive information, so an analyst may not be permitted to access the content. In summary, the challenge of QoS measurement in MCCNs is how to reduce the size of data for fast processing without compromising measurement accuracy.

QoS visualization in MCCNs should be seamlessly integrated with the QoS measurement algorithm. In addition, the visualization platform should provide multiple different views, allow interaction with the user, and support playback capabilities. Also, since a typical MCCN is a mobile network, geographical information should be displayed, e.g., placing the network nodes on a geographical map and showing their movement. Essentially, all these capability should be provided fast and on demand.

1.3.2 QoS Prediction

There are two main goals of QoS prediction in MCCNs. QoS prediction can be used for mission planning. With the help of QoS prediction results, one can deploy network components, i.e., soldiers in a battlefield or paramedic units in a disaster situation, more efficiently, and forecast network performance under different scenarios. This is the use of QoS prediction before the mission and requires long-term QoS prediction. The other use of QoS prediction is failure prevention that is during the mission use case. QoS prediction helps to foresee and prevent potential communica-

tion failures in the near-future while the network is operating. This requires short-term and timely QoS prediction. Therefore, a good QoS prediction framework should provide accurate long-term prediction and accurate and fast short-term prediction.

The main challenge in QoS prediction in MCCNs is to identify good predictors that can provide high prediction accuracy. A natural potential predictor of future QoS is the past QoS. However, in MCCNs QoS predictions should be made almost immediately and QoS measurements are not readily available for short-term predictions. Therefore, using past QoS to predict future QoS may not be possible in MCCNs.

Secondly, QoS prediction is usually a classification problem where packets are classified either loss or delivered when predicting packet loss or either high delayed or not when predicting latency. A data set is defined imbalanced when data is distributed unequally between classes [40]. In a typical network only a small portion of the packets are expected to be lost or show high latency. Because of this nature of the data, it is identified as imbalanced data. Performances of most of the classification algorithms suffer when applied to an imbalanced data [40]. This situation creates another challenge for QoS prediction.

Finally, due to the huge size of the network traffic data, only a certain portion of the historical data can be used for QoS prediction. Usually it is not possible to store all the collected data permanently. Even if all the data are available, processing and feeding all these data to a prediction algorithm is usually not feasible. It is also possible that only the recent historical status of the network affects the future status. Therefore, studying how much historical data to be used is required. Another restriction on historical data is about the immediate state of the network. Since network data are needed to be stored and processed after collection and this requires time, a prediction algorithm may not have access to the immediate state of the network while doing short-term online prediction.

1.4 Expected Original Contribution

Tables 1.1 and 1.2 summarize the existing research in the literature, which is discussed in section 1.2, and how it falls short in addressing the unique challenges presented in section 1.3.

The expected original contributions of this thesis are summarized as follows:

- We lay out the details of a complete QoS analysis framework in MCCNs.
- We develop a fast, scalable, highly accurate, and security-preserved QoS measurement algorithm.
- We integrate the developed QoS measurement algorithm running in the back-end with a front-end visualization platform supporting multiple, diverse functionalities to enable visual analytics.
- We develop a toolkit consisting of various statistical and machine learning algorithms to support fast and accurate QoS prediction in both long-term and short-term. We intend to answer the following questions in QoS prediction: (1) *What are the important predictors of QoS and what should be collected and stored in order to make accurate predictions?* (2) *How much in the future can the QoS be predicted with the desired accuracy?* (3) *How much historical data should be stored and used in order to make accurate and scalable predictions?*
- We develop prediction algorithms that are highly interpretable while providing a satisfactory accuracy in order to help network planning and damage prevention in MCCNs.
- All the research developments use open-source software.
- We demonstrate the developed QoS measurement, visualization, and prediction capabilities using a real MCCN's data.

Table 1.1: Summary of Existing Research in the Literature on QoS Measurement and Visualization and Where It Falls Short in Addressing Our Unique Challenges.

QoS measurement and visualization

Existing Research	**Limitations addressing our unique challenges**
Active measurement	MCCNs do not allow superimposing test traffic.
Passive measurement based on packet data	Large data size makes processing slow because of disk I/O bandwidth
Passive measurement based on flow data	• Flow data does not allow measuring of the QoS of individual packets. • Flow data does not have unique identifiers, leading to low accuracy. • Including packet content could improve accuracy, which raises security concerns.
Visualization capabilities are usually provided in popular traffic analysis software packages.	• Visualization of traffic volume is available, but visualization of QoS is limited. • QoS visualization needs to be seamlessly integrated with the QoS measurement algorithm. • Visualization platform needs to provide multiple functionalities and support display of geographical information of mobile devices in MCCNs.

Table 1.2: Summary of Existing Research in the Literature on QoS Prediction and Where It Falls Short in Addressing Our Unique Challenges.

QoS prediction	
Purpose is web service selection and dynamic traffic control.	To serve this purpose needs only short-term prediction, but MCCNs need both short-term and long-term prediction.
Past QoS is used to predict future QoS.	• QoS measurement takes time, so it is desirable to use immediately available non-QoS related predictors to predict QoS. • Class imbalance. • Limited storage for historical data.

1.5 Organization of the Thesis

In chapter 2, we explain methods and algorithms used to solve QoS measurement, visualization and prediction related problems in MCCN. We introduce a passive one-way QoS measurement algorithm. We provide results of rigorous testing of our method. Next, we present details of our visualization platform. Then, we describe QoS prediction task in MCCNs and our prediction workflow. Experiments and results of our prediction framework are also provided. Finally, we introduce a new classification algorithm called "treed logistic regression" as a part of our prediction toolkit. This method is also tested and compared with other well known classification methods.

In chapter 3, we summarize the thesis, evaluate research question introduced in chapter 1 and point out future research directions.

Chapter 2

PROPOSED METHODOLOGY AND COMPUTATIONAL EXPERIMENTS

2.1 Application Context

Section 1 discusses common application areas, importance and features of MCCNs. MCCNs may have many application areas. In this study, we focus on the military use of MCCNs. This thesis covers our research efforts in a collaborative project with U.S. Army. The network we study consists of electronic communication devices, similar to handheld radios. We call each of these devices a node in the network. All of these devices are connected wirelessly to form an electronic communication network.

The network under study uses multicast routing scheme. Multicast routing is the transmission of a network packet to zero or more destination nodes identified by a single IP destination address. A multicast network packet is delivered to all members of its destination node group by a special network unit, called multicast router, which knows the membership of all or some groups in the network [41]. This implies that a packet may have more than one destination nodes and as many copies as the members of multicast group. Multicast doesn't guarantee that a network packet will be delivered to all the members of the destination group or in the same order relative to other packets. The membership of a multicast group is dynamic. Nodes may join and leave groups at any time. A node may be a member of many groups at a time. A node can send packets to a group whether or not it is a member as long as it knows the group IP address. When a node sends a packet to a multicast group, it may not be aware which nodes will actually receive that packet and the packet is not aware of its final destination.

Network data we study are provided by an official electronic test office of U.S.

Army. The data are collected during a device testing stage. Tests are conducted on the field using the actual devices and mimic the real use cases. The purpose of the tests is to understand and estimate the performance and behavior of these devices in a real warfare. The data are collected at each node in the network. Then data from all the nodes are brought to a central location and stored in a common repository. In a typical setting, data are collected during the day and dumped to the repository at the end of each day. Then, the data are analyzed in order to evaluate the day and help command and control for the next day. QoS measurement, visualization and prediction are three main aspects of the network data analysis and performance evaluation. These operations are very important for command and control.

2.2 A Comprehensive QoS Analysis Framework

We propose a comprehensive QoS measurement, visualization and prediction framework for MCCNs. Data collection in a typical MCCN setting is explained in section 2.1. Once the data are collected in the field, they are moved to a central or distributed location for storage. Initial data are not structured. Therefore pre-processing of the data is required. During the pre-processing stage data are segmented according to their collection sites and/or collection date. We store the data coming from each network node on a different flat file. These files have standard packet capture format [4]. Section 2.3 discusses data preparation, processing for QoS measurement purposes and our proposed measurement algorithm.

After pre-processing, data are stored in a structured manner and the size of the data is reduced significantly. Pre-processed data are stored in flow format. Two data storage options are explored for this format; binary flat files and proprietary database. To understand performance differences between the different data storage options and data formats, we performed several tests. Given that we could experience

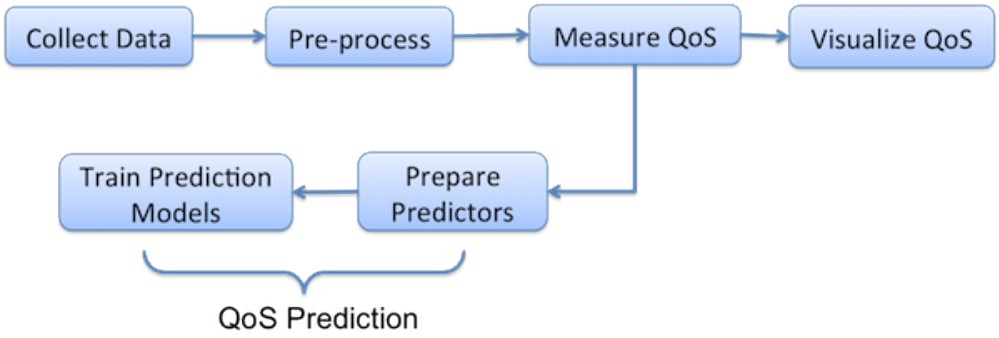

Figure 2.1: Proposed QoS Analysis Framework Data Flow Diagram.

performance bottlenecks when harvesting data, populating/moving database tables, and performing complex queries, we identified the following metrics for performance comparison; conversion time between formats, storage space and basic querying times. The conversion test is done for the data repository which consists of 36 packet capture files of total size 3.55 GB. Firstly, packet capture files are used to generate the flow files. This takes around 40 seconds in our configuration and it provides a reduction in file size from 3.55 GB to 610 KB. When the traffic flow information is stored in databases, it requires 2.5 MB storage space. Inserting the flow records to the database takes around 30 seconds. Table 2.1 summarizes the storage space and preparation time required for each data storage option mentioned.

Table 2.1: Evaluation of Different Data Storage Options

	Data Storage		
	packet capture	flow	flow database
Preparation Time	0 sec.	40 sec.	70 sec.
Storage Size	3.55 GB	610 KB	2.5 MB

We also evaluate the time required to extract information from the data for each storage option. We consider three basic queries that summarize certain aspects of the network traffic. These queries provide the following information:

– Q1: Traffic (number of bytes and number of packets) sent per IP address.

– Q2: Traffic (number of bytes and number of packets) between all different source and destination node pairs.

– Q3: Traffic (number of bytes) over different transport protocols

Table 2.2 summarizes the results of our experiments.

Table 2.2: Time to Query Data for Different Storage Options.

	Querying times in milliseconds(ms)		
	packet capture	flow	flow database
Q1	56595	54	809
Q2	63591	136	478
Q3	18863	51	330

We can conclude that flat flow files outperforms other two approaches in terms of storage size and querying times. Therefore, we decided to store the data as flat flow files.

Once the data are stored in a proper format, they are ready for visualization and prediction. Visualization algorithms query the data to extract the fragments of interest. Then, these data are used to create web-based visualizations. Details of proposed visualization platform are discussed in section 2.4. On the other hand, structured and QoS measured data are fed to multiple machine learning and data analysis algorithms and a complete QoS prediction and analysis platform is built. Section 2.5 discusses the details of QoS prediction platform.

Efficiency and scalability of the methods are considered developing each stage of the mentioned QoS analysis framework because of the time and life critical nature of

the MCCN. We propose a complete and effective solution to all QoS related questions in MCCNs.

2.3 Passive One-Way QoS Measurements

Section 1.1 discusses the necessity of measuring QoS metrics to understand the perceived quality by network users. To characterize the perceived QoS, we calculate the one-way delay which is the amount of time passed from the packet generation until its reception at the destination [42] and one-way loss which is whether a packet is delivered to its intended destination or lost in the way. We call these two metrics as one-way QoS metrics. One-way metrics are directional metrics calculated between two end-points in the network. Reasonable values for these metrics vary for different applications.

As discussed in section 1.2.1, passive measurement is preferred in MCCN. In passive measurement method, QoS is measured after the data are collected and put into a central or distributed storage. In our application, QoS is measured using the packet data which are collected by the data collectors also referred as data collection sites (DCSs). This process is illustrated in Figure 2.2 for two DCSs. The overall goal is to measure the QoS between each pair of DCS to characterize the network performance. This is achieved by the correlation of packets collected at the both ends of a link. What we mean by correlation of packets is simply tracking a packet at the source DCS and finding the same packet at the destination DCS. This way one-way delay can be calculated by comparing the collection times of the same packet at the source DCS and destination DCS. If the a packet sent from a DCS cannot be found at its destined DCS then it is marked as a packet loss.

Accurate measurements of QoS require all DCS nodes to be time synchronized. In our current implementation, the clocks of the DCS nodes are not assumed to be tightly

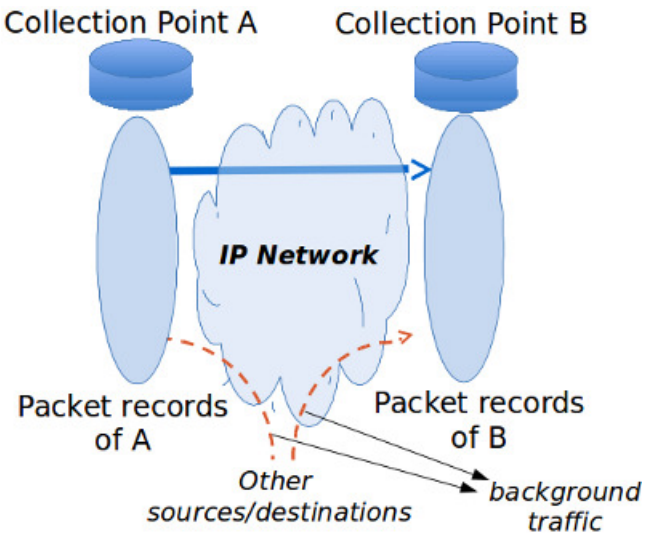

Figure 2.2: Illustration of the Communication Between DCS Nodes A and B.

synchronized. However, certain modifications might be required if the misalignment of the timestamps across different DCS nodes are significantly large. Also, external routing information is known in a typical setting. In other words, prior knowledge about the addresses connected to DCS nodes A and B is known. The background traffic can be excluded by matching the unique source and destination pairs observed at both DCS nodes.

2.3.1 QoS Measurement Using Packet Traces

As discussed in section 1.2.1 using packet traces is the most common and standard way of measuring QoS. It allows tracking and correlation of the packets easily by looking at the user data of each packet. To correlate packets between two DCS nodes, we compare the user data of the packets in addition to transport information. Our packet matching algorithm uses the packet traces from two DCS nodes for which we are interested in calculating the one-way delay. A Berkeley Packet Filter (bpf) expression can also be provided so that the calculations focus on the traffic of interest. For example, when the addresses associated with DCS nodes are known, it is better

to focus only on the packets between these addresses. After the packet traces are filtered, the packets in DCS node A is correlated with the packets in DCS node B. Correlated packets are used to measure the one-way delay by simply differencing the destination time and source time. If a packet from DCS node A is found to be lost, then there is no packet in DCS node B that is correlated.

This method allows exact matching of each packet. Therefore, we can calculate accurate delay and packet loss statistics.

2.3.2 QoS Measurement Using Flow Data

Similar to the correlation of the packets, we correlate flows using the five-tuples and the timestamps. Since the packet content is not available in flow format, this correlation is prone to matching wrong flows. In section 1.2.1, the rationale behind using network flow data for QoS measurements and the limitations are discussed. To enable exact correlation of packets, the packet content should be integrated in the flow data. Although fairly accurate approximations to QoS metrics can be made with network flow data, a higher confidence level is desired for MCCN. In order to achieve accurate QoS calculations, we propose an efficient approach to integrate the packet content into the network flow data. Here, we exploit the extensibility of the standard network flow data formats. Being able to access the content through packet traces, we add external information to predefined fields of network flows. Since copying all the packet content to a network flow is not a reasonable choice, we hash the content of the packets in order to characterize each packet with a near-unique identifier.

Benefiting from the ideas of hash-based packet selection [43], we obtained a unique 16-bit identifier for every packet content and saved this information along with the control information of the corresponding flow. This identifier is used to successfully correlate network flows collected at different DCS nodes since only same packets

would generate the same hash values in the ideal situation. Use of a proper hash function that can uniformly distribute inputs while minimizing the time required to calculate the hash value is essential for the success and performance of our approach. After generation of this unique identifier for each flow record, delay and packet loss are calculated by correlating the flows across different collection DCSs. The only difference compared to correlation using the packet traces is that hash values are now used for correlation instead of the packet contents.

The approach for flow correlation is illustrated in Figure 2.3. Let λ be the deterministic upper bound to the one-way packet delay between DCS nodes A and B. In practice, the value of λ will be set to a conservative value, e.g. $\lambda = 5$ sec. For each flow x started at time t_x from F_A, we find a flow record y (starting at t_y) from F_B that shares the same five-tuple of the flow x where $|t_y - t_x| < \lambda$. If the flow x can not be correlated to any flow from F_B, the packet corresponding to flow x is assumed to be lost. Our algorithm requires a loose synchronization between the clocks of two DCS node. If the flows are severely misaligned, a preliminary realignment procedure should be applied to reduce the clock error down to a value comparable with λ.

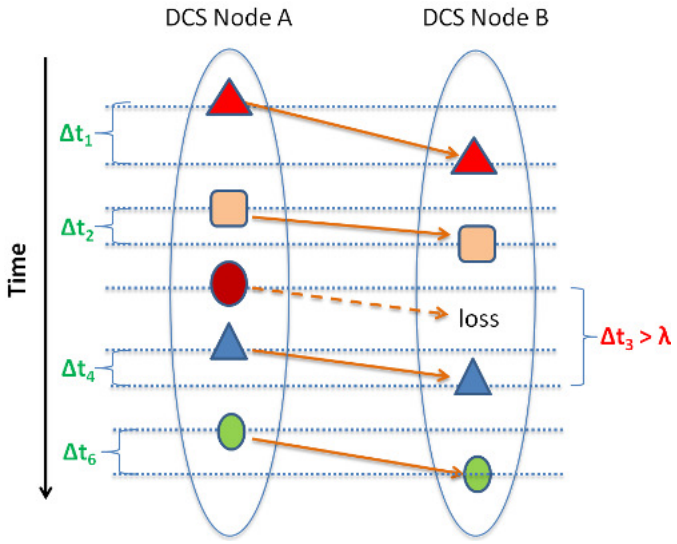

Figure 2.3: Illustration of the Flow Correlation Between DCS Nodes A and B.

2.3.3 Computational Experiments and Results

Our experiments use an Ubuntu 12.04 system with 32 GB RAM, six core CPU (XEON E52630, 2.3GHz). We use libtrace [44] for filtering packets, jnetpcap [45] java library that uses libpcap [4] to process packets. SilK tools [46] are used to convert packet traces to flow format. A custom java application is implemented to estimate the QoS from the flow records. SilK tools [46] also work on compressed flow files. The flow files are compressed using gzip utility [47] to illustrate the benefit of reducing the file sizes.

As mentioned earlier, disk I/O is a bounding factor on the computation times. One may consider compressing the flow files if HDD storage is used. SilK tools can work with compressed files without a need for changing the proposed algorithms. We show that significant gains in terms of computation time are achieved with the compression on HDD. However, one should keep in mind that there is a trade off between the use of disk I/O and processing. The compressed files need to be decoded to be used in the algorithms. Since we use SSD storage disk I/O bandwidth is not our biggest concern. We tested our algorithm with both compressed and uncompressed flow files. We have observed that it runs almost as twice as fast on uncompressed flow files. Therefore we concluded that decision of using compressed or uncompressed flow files should be made based on the hardware used.

The dataset contained 43 DCS nodes containing a total of 43 active node pairs with traffic on which we can calculate QoS. Table 2.3 summarizes the characteristics of the collected data over the DCS nodes. We also run a simple select query on each format to illustrate the time required to retrieve certain packet/flow information.

Table 2.3: Data Characteristics

	Data Size		
	min	max	avg
Packet trace	5.4 KB	149.4 GB	8.8 GB
Flow (compressed)	509 B	1.2 GB	64.3 MB
Flow (uncompressed)	2.8 KB	12.7 GB	675.8 MB

2.3.3.1 Hashing strategies

An evaluation of the well-known hash functions specifically for the network measurement domain is provided by [43]. Based on the discussion due to [43], we evaluated five of the best performing Hash functions (Hseih, OAAT, BOB, CRC32, FNV) for our application. The nonlinearity of each hash function is tested since our main purpose is to generate a unique identifier for each packet. Our tests concluded that all five hash functions can provide the desired nonlinearity. Among all, BOB hash function performs the best in terms of computation time. The performance results of each hash function is provided in Table 2.4. These results are obtained from conversion of approximately 380.2 GB of packet trace data to network flow data using different hash functions to hash the packet content and add it to network flows. Resulting network flow data contains approximately 544 million individual flows.

2.3.3.2 Estimation accuracy

An important metric for the usage of flow data to calculate QoS is its estimation accuracy. Since flow-based packet matching has potential to mismatch certain packets, the delay results might not be the same as exact matching results. However, we found that the error was small and the estimation was good. The mean square error

Table 2.4: Time to Convert All Packet Traces to Network Flow Format with Different Hashing Strategies

Hash Function	Time (minutes)
Hseih	31.6
OAAT	58.7
BOB	24.5
FNV	35.8
CRC32	56.6

between the delay calculations using flow data (compressed or uncompressed) and the packet capture data was 3.31×10^{-5} showing a close correlation. Figure 2.4 provides a scatter plot of estimated (flow based) and actual (packet based) delay. Our approach provides accurate estimation of the one-way delay with the flow files.

2.3.3.3 Computation time

The time required to query the flow files are shown in Figure 2.5 for different file storage options. X axis corresponds to the number of flows in the flow file to be queried. Although not provided here, the file sizes increase almost linearly with the increase in the number of flows in the file. This comparison is done on two different drive types, solid state drive (SSD) and hard disk drive (HDD). We also investigated if compression is improving the query times for different drive types.

The benefit of using SSD is clear as computation times are compared. However query times are larger when flow files are stored as compressed in SSD. This behavior is due to the trade-off between the disk I/O bandwidth and CPU usage. Although faster disk reads are enabled by the reduction in file sizes with compression, an additional overhead is introduced to CPU for decompression. Since we use SSD, faster disk read

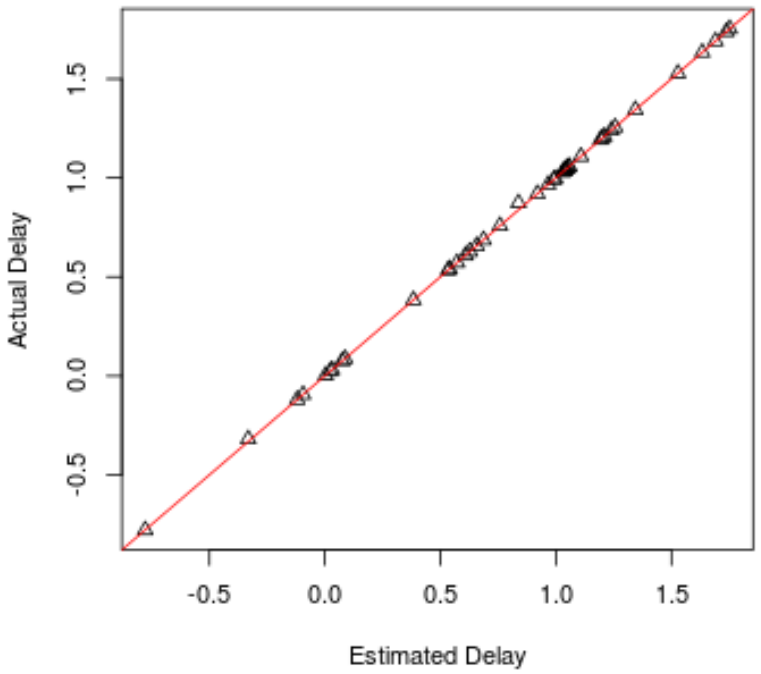

Figure 2.4: Scatter Plot of Estimated Delay and Actual Delay

does not provide a significant gain when flow files are compressed. This is just the opposite when HDD is used for storage. Compression provides significant gains in query times since disk read times are taking more time compared to decompression on HDD.

Once the data are in the appropriate formats, QoS calculations can be performed. Table 2.5 summarizes the computation time for QoS calculations for each DCS pair using the different formats on SSD drives. The maximum computation time is the smallest when the uncompressed flow files are used. Figure 2.5 explains such behavior clearly. Uncompressed flow files enable faster querying as the file sizes gets larger compared to compressed ones. This behavior is due to the trade-off between the CPU usage and disk I/O. As shown in Table 2.3, the average file size is around 675.8 MB and the average computation time is the smallest for the uncompressed flow files.

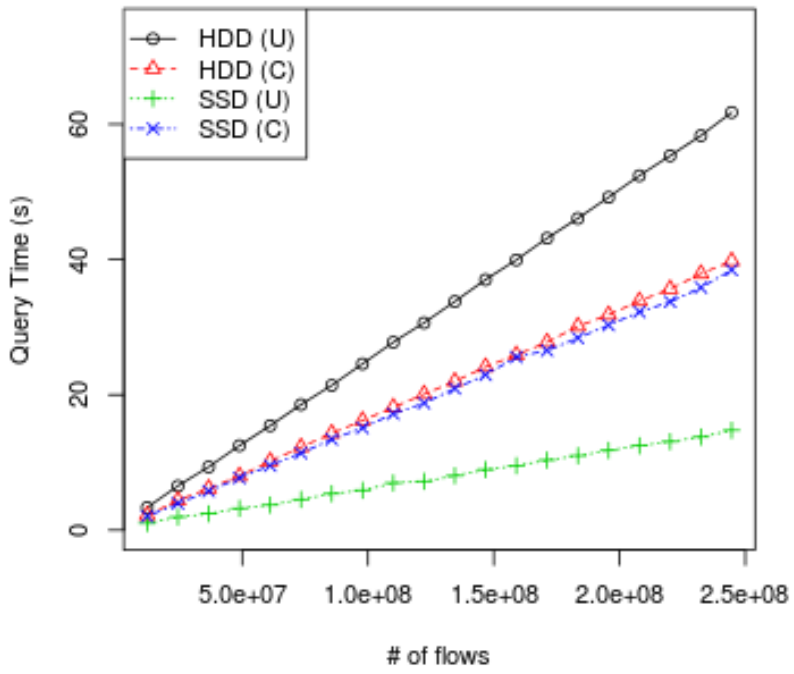

Figure 2.5: The Query Times Across Different File Storage Options. X Axis Corresponds to the Number of Flows in the Flow File.

The theoretical read speed of SSD drives used in this study is 1 GB per second which explains the better results with SSD drives. This is 200 MB per second for HDD. The choice of using compressed or uncompressed flow files depends on the characteristics of the data and the hardware. Compressed flow files can also be chosen when there is lack of disk space.

Table 2.5: Delay Computation Times Over Each DCS Pair

	min	max	avg
Packet trace	2.02	1123.90	130.36
Flow (compressed)	0.84	112.36	14.20
Flow (uncompressed)	0.38	103.10	9.63

2.4 QoS Visualization

Visualization of network, measured QoS and network health is very important to detect and understand any network issue fast and efficiently. Necessity and important capabilities of a good visualization platform are discussed in section 1.3.1. To satisfy aforementioned needs, we developed a web-based multifunctional and interactive visualization platform.

We build the visualization platform on top of Ozone Widget Framework (OWF) [48]. OWF is an open-source web application that allows users to easily build online tools called widgets and provides configuration, customization options and communication capabilities between widgets. Each widget acts as a web-page. They are developed independently and may serve a different purpose or multiple widgets can work collaboratively to accomplish a common goal. We develop widgets using java and javascript programming languages. For visualization purposes open-source D3 [49] javascript library is used. To provide on-demand visualization, MCCN data are stored in flow format. This allows faster processing and querying of the data.

Our visualization platform has the following capabilities:

1. Topology and Connectivity

   Figure 2.6 shows the topology and connectivity view of our visualization platform. Users can easily see the connectivity of the nodes in the network using this view. Nodes are colored based on their type. Size of a node represents the amount of traffic data going through that node. IP addresses of the nodes can be seen by hovering the mouse over a node. Links between nodes represent topology (physical links) or connectivity (whether or not there is traffic between the nodes) at choice. Users can filter traffic and nodes based on transport protocol to focus on the portion of interest.

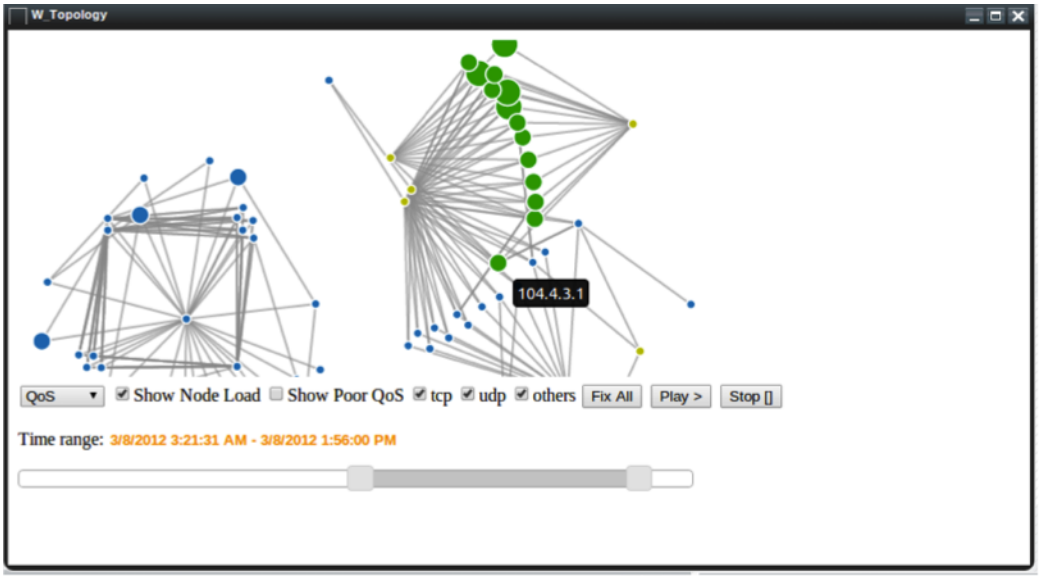

Figure 2.6: Topology and Connectivity View of the Visualization Platform.

This view also has playback capabilities. A sliding bar allows users to select a start and end time. Then, aggregated connectivity of the selected time period can be investigated. This function allows users to visualize the network at any time in the past. By hitting the 'play' button, topology shows changes in the connectivity over time at any granularity or aggregation level. This capability is particularly important in order to pinpoint the time and possible reason of any network issue.

2. Drill-down Charts

Figure 2.7 shows the detailed topology view widget. In the standard topology and connectivity view, users can click on any link and open up drill-down charts. There are three types of drill-down charts. First chart shows traffic load of the selected link over time as a bar chart. Second chart shows the transport protocol distribution of the traffic on the selected link. Third chart gives general information about the link. It reports source and destination IP addresses of the end-points and the total amount of traffic within the selected time window

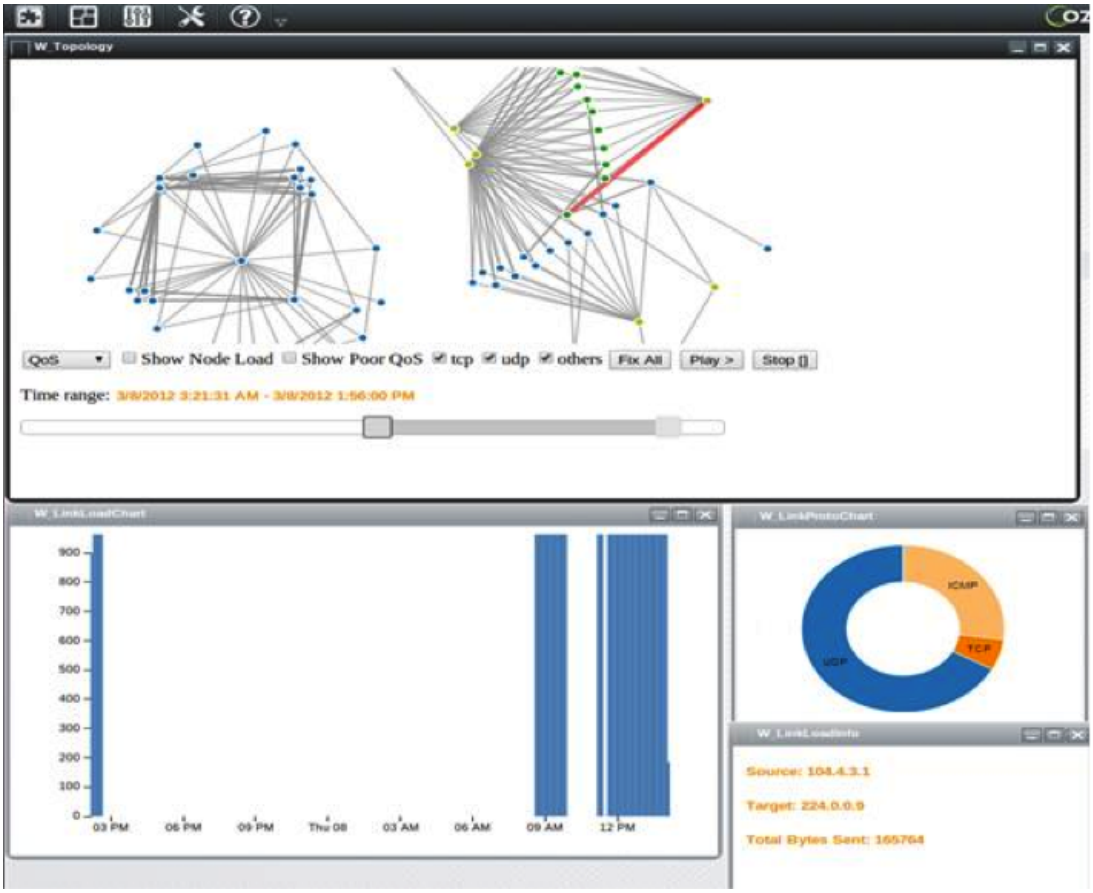

Figure 2.7: Drill-down Charts. Selected Link is Highlighted in Red.

of the selected link.

These charts helps to diagnose issues in the MCCN. Amount and type of traffic over time have a significant effect on the network performance. Therefore, it is important to monitor traffic profile of the links on the network.

3. Performance Charts

Figure 2.8 shows an example performance chart. This chart shows average latency and total message completion rate of a selected link. Message completion rate is defined as the proportion of successfully delivered packets to all sent packets from a sender to a destination in the network. This metric is derived from measured packet loss metric. In Figure 2.8, chart is color coded. Green

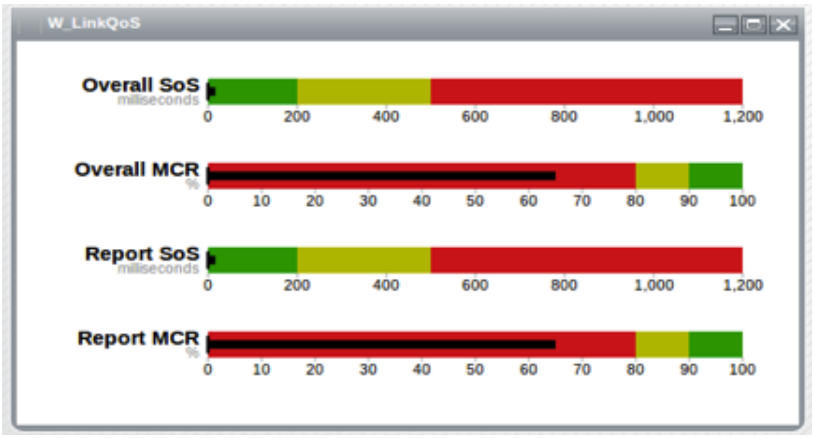

Figure 2.8: Performance Visualization of a Link.

zone indicates good performance, yellow zone indicates average or alarming performance and red zone shows poor performance. Levels of these zones can be defined by user and it depends on the application. This view allows users to easily see good and bad performing links in the network and act accordingly.

4. Geographical Map

   Figure 2.9 shows the geographical map view of our visualization platform. As mentioned in section 1, a typical MCCN consists of mobile nodes and usually these nodes are aware of their geographical location with the help of Global Positioning System (GPS). Yellow pins on the map represents nodes of the network. These nodes are placed on to the map based on their reported altitude, longitude and latitude values. Figure 2.9 shows red regions on the map. These regions represent bad performance locations within the selected time window. If a node has poor performance at anytime within selected time window, its location is marked red. This allows users to easily identify geographical location of a possible network issue. This view also has playback capabilities. Users can track movements of nodes over time.

Our visualization platform provides all the critical capabilities to monitor network

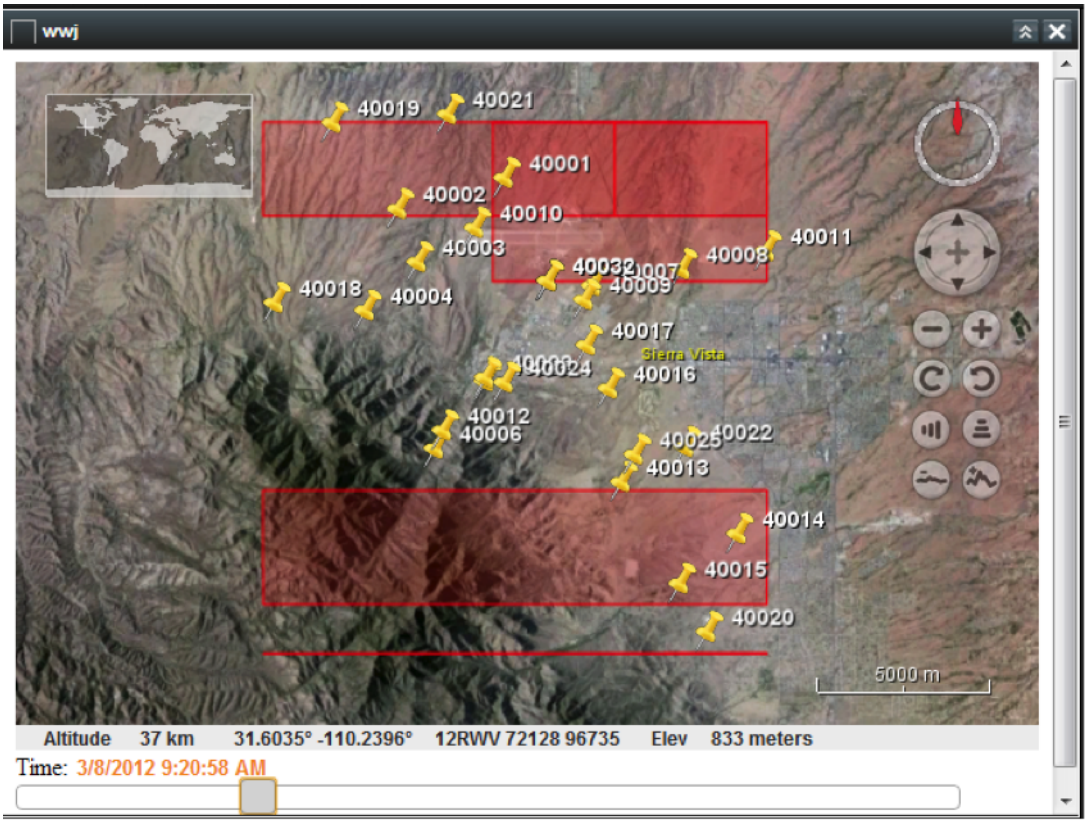

Figure 2.9: Geographical Map Visualization. Yellow Pins Represent Nodes.

situation and performance in MCCNs. Our aforementioned efficient data storage and processing techniques allow on-demand and interactive visualization. Visual analytics plays a very important role in QoS analysis in MCCNs.

2.5 QoS Prediction

As discussed in section 1.3.2, QoS prediction is very important for mission planning and failure prevention in MCCNs. An accurate prediction of QoS metrics would allow network administrators to identify vulnerabilities in the network before they cause any problems. It also helps network planning and efficient resource allocation. However, accurate prediction alone is not sufficient to help network planning. Our ultimate goal is to help network administrators to better understand the network and its performance. For that reason, a good QoS prediction framework should explain

relationships between performance predictors and the performance itself. Showing the impact of events and components in the network on the performance will give network administrators a valuable insight.

We employ several statistical analysis and machine learning methods in order to provide a complete and detailed QoS prediction and analysis framework.

Details of the data used in this study are provided in section 2.1. For QoS prediction study we analyze one day of test data. Data collection starts around 9 AM and stops around 4 PM. During that time, test continues without any interruption. Traffic activity in the network is not stable the whole day. We observed that some time periods and/or some nodes are more active than the others. At the end of the day 225004 network packets are captured and stored. These packets are collected from 15 nodes. All the nodes in the network are mobile. They regularly report their current geographical location using a special pre-determined format. This format includes latitude, longitude and altitude information. These reports are also captured and stored. Thus, geographical locations of the sender of all packets are known.

2.5.1 The Classification Task

Classification is learning how to assign a new observation to one of a pre-defined set of categories or classes on the basis of a set of observations (training data) whose category memberships are already known. QoS prediction is usually a classification task where packets are observations and the goal is to classify new packets as either loss or delivered or high delay or low delay. The classification task described here is referred to as supervised learning. The goal in supervised learning is to derive a mathematical function or formulation that maps input data to a pre-defined category. This function is derived by identifying and generalizing core characteristic of objects of each category and finding a representation for it. Success of supervised

learning heavily depends on the quality of the training data. Training data should be representative of the main concepts of the data and not dominated by noise.

In a typical classification problem, data are represented as a table of observations. Each observation is described by a set of features (attributes / predictors). This set of features is called feature vector. These features are the columns of the table. Typically, features are one of two types; numeric (continuous real numbers), or nominal (a set of unordered values). Some learning algorithms may only work with one of these type of features. Along with features, each observation in training data has a label that denotes its class. If class label can only be one of two distinct values, this type of classification is called binary classification. Our problem is a binary classification problem.

2.5.2 Identifying Predictors

Identifying the potential predictors (features) is the first step of building our prediction framework. As discussed in section 1.3.2, historical QoS may not be available in the MCCN even though it is potentially a good predictor of the future QoS. That's why we turned our attention to traffic patterns. [24] and [37] shows that congestion, exceptionally high traffic on a link between two nodes in the network, implies packet loss and high latency. It is reasonable to believe that high traffic or sudden changes in the traffic of a node may cause performance degradation. Therefore, we claim that observed traffic patterns of nodes and links in a network can be used to predict QoS metrics.

Traffic pattern of a node can be represented as time series data where each observation is the volume (in bytes) of a network packet that is captured by this node at a given time. On the other hand, traffic pattern of a link can be represented as time series data where each observation is the volume (in bytes) of a network packet that

is sent from one end of the link to the other at a given time.

Our goal is the make packet level QoS predictions that is to predict the delay of a packet and the probability of a packet being lost. Our idea is to use traffic patterns of nodes and links that are closely related to a packet to predict QoS metrics of that packet. We define closely related nodes and links of a packet as follows; sender node of the packet, receiver node of the packet, link between the sender and receiver nodes of the packet. In addition to these items, traffic of the multicast group may impact the packet QoS. As stated in section 2.1, the network we study uses multicast routing scheme. If a packet is routed through a multicast router, traffic of that router may have an effect on QoS. Therefore, we declare multicast group traffic pattern as a potential QoS predictor. Multicast group traffic pattern can be represented as time series data, where each observation is the volume (in bytes) of a network packet that is sent to or received from that multicast group IP address by any node.

Besides traffic related information, packet related information may be used for prediction. Network packet header contains the following information [3]:

- Source address
- Destination address
- Transport layer protocol
- Packet length

These items are good candidates to be QoS predictors since they describe the packet and the route it takes in the network. However, our preliminary analysis showed that all the packets in our data share the same transport layer protocol that is UDP [50] and same packet length. Hence, transport layer protocol and packet length cannot be used as a predictor and they are discarded.

Finally, we consider geographical location related predictors. As mentioned above, geographical locations of the nodes are known by means of latitude, longitude and altitude. It is required to incorporate this information to our prediction framework. Intuitively, we claim that geographical distance between two wireless nodes in a network might affect communication performance between these nodes. Thus, we introduce a new predictor, geographical distance between sender and receiver nodes of the packet. We calculate the geographical distance using the Haversine formula [51].

Consequently, we have the following list of potential predictors:

- Traffic pattern of the sender node of the packet
- Traffic pattern of the receiver node of the packet
- Traffic pattern of the link between the sender and the receiver nodes of the packet
- Traffic pattern of the multicast group through which the packet travels
- Geographical distance between the sender and the receiver nodes of the packet

The question remains is how to represent traffic patterns in a proper format to use them with standard classification algorithms. Traffic patterns are defined as time series data. We cannot directly use a time series as a predictor. Our goal is to represent our data as rows and columns where each column is a predictor and each row is a packet whose QoS values are to be predicted. We need to discretize traffic pattern time series data. We use the idea of data binning for discretization. Time range of all observations are divided into equal intervals of time called bins. Original data values which fall into a bin are replaced by a value representative of that bin such as average or total of the observations. This way each traffic pattern related predictor, sender node, receiver node, link and group patterns, will be represented as

one feature of a bin. If we predict the QoS values of a packet captured at time t, let the length of a bin be t_b and the time range of n^{th} bin be T_n, then:

$$t - n * t_b \ > \ T_n \ > \ t - (n+1) * t_b, \qquad n = 0, 1, 2, 3, 4, ... \tag{2.1}$$

One packet may be associated with as many bins as we need as long as the times of the bins do not exceed the test start time. It should be noted that the range of the time series data mentioned here is between the time when the packet is captured and the start time of the test. In other words, we may use the traffic pattern values before t and after t_0 where t_0 is the time when the test starts.

Figure 2.10 illustrates a row of our data where each field is a predictor

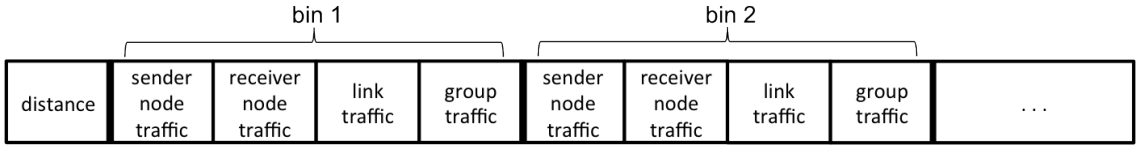

Figure 2.10: Illustration of Predictors of a Packet. Each Packet Might Have $4*N+1$ Many Predictors Where N is the Number of Bins.

We also need to specify QoS metrics to be predicted. For demonstration purposes, we predict packet loss in this study. Same or similar methods can be applied to predict delay or other QoS metrics as well. Our response variable (class variable) will have value of 1 ($Y = 1$) if a packet is lost and value of 0 ($Y = 0$) if a packet is successfully delivered to its intended destination. From now on, we may use "$Y = 1$" to represent a lost packet and "$Y = 0$" to represent a delivered packet.

2.5.3 Prediction through Packet Classification

In order to find the best approach, we tried some of the widely used classification algorithms.

2.5.3.1 Logistic Regression

Logistic regression is a probabilistic classification model [52]. It is commonly used to obtain linear models for binary classification problems. Binary categorical dependent variable is predicted based on one or more predictor features. Logistic regression usually works only with numeric predictors. General linear model has the following form:

$$y_i = \beta_0 + \beta_1 * X_i1 + \beta_2 * X_i2 + ... + \beta_n * X_in \tag{2.2}$$

Logistic regression uses logistic function to classify binary response variables. Logistic function always takes on values between zero and one and has the following form:

$$F(y) = \frac{e^y}{e^y + 1} \tag{2.3}$$

When logistic function is integrated with general linear model we have the following function:

$$\pi(x) = \frac{e^{\beta_0 + \beta X}}{1 + e^{\beta_0 + \beta X}} \tag{2.4}$$

where X is the feature vector and β is called the coefficient vector.

Please observe that $0 < \pi(x) < 1$, which is appropriate because $\pi(x_i)$ is the probability that observation x_i is either 1 or 0. Logistic regression uses Equation 2.4 to predict class assignments of new observations. Parameters of this equation, β_0 and β vector, are estimated using maximum likelihood estimation [52]. Details of this method are out of the scope of this document. Provided references might be referred for further detail.

2.5.3.2 Classification Tree

Classification tree or decision tree is a supervised learning algorithm that recursively partitions the data set using one predictor at a time and yields multiple rectangular regions. Then each region is denoted by a class label, often by majority class label of training data points that falls into that region. When a new observation is seen, algorithm first determines which region this observation falls into, then its class label becomes that region's label. Classification tree method, introduced by Breiman et al. in 1984, is one of the most popular classification methods in data mining due to its simple yet powerful algorithm and easy to interpret model [53].

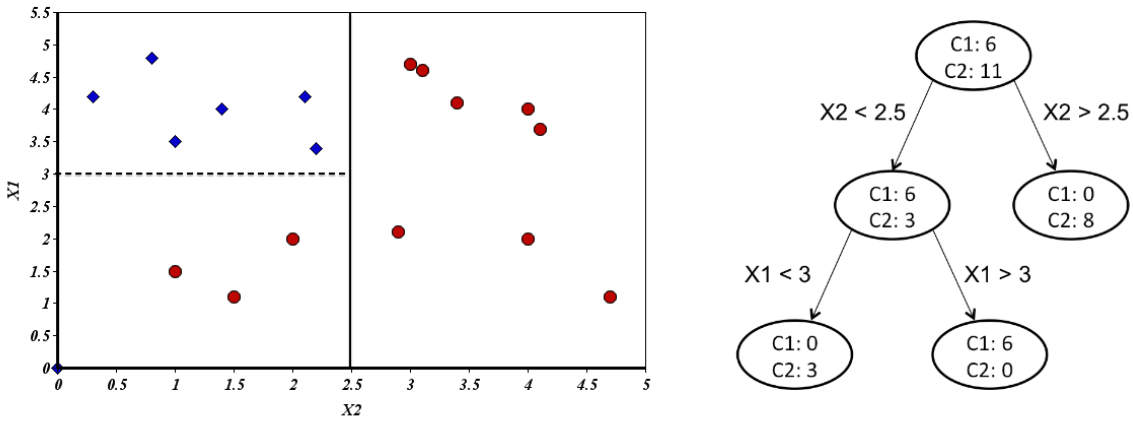

Figure 2.11: Visualization of a Sample Two-class Classification Tree. Partitioning of the Feature Space (left) and Tree Structure (right).

Classification tree algorithm starts with one node that contains all the training data. This node is called root node. The initial data set is split into two or more new nodes using a function which tries to find locally best way to split the observations based on a criterion. Splitting recursively continues until a stopping condition is reached, often when the node contains observation only of one class. After construction, tree is usually pruned in order to reduce its complexity and increase gen-

eralization, hence prevent overfitting. Classification tree method has the following main components:

- A way to find the best split.
- A rule to determine when to stop further splitting of a node.
- A method to prune the grown tree (optional).
- A rule to assign a value to each terminal node.

Details of splitting and pruning methods are out of the scope this document. Provided references might be referred for further detail.

2.5.3.3 Random Forest

Random forest is a supervised learning algorithm that grows many classification trees [54]. In standard classification trees, each node is split using the best split among all predictors, however, in random forest the best split is chosen among a randomly selected subset of all predictors. Each tree in random forest is grown using a randomly (with replacement) selected subset, often two-third, of the training data set to the largest extent possible. There is no pruning. Although this process seems counterintuitive, random forest is shown to outperform many well-known classification algorithms in terms of classification accuracy [54]. In addition, it is fast with big data sets and simple to use in the sense that it has only few parameters. It is also robust against overfitting. It is shown that performance of the random forest depends on two things; the correlation between any two trees in the forest and the strength of each tree.

Prediction using random forest is performed as follows:

- The observation to be predicted is put down each of the trees in the forest. Each tree gives a classification result.

– Random forest chooses the classification having the most votes over all the trees.

Random forest is capable of reporting feature importances by identifying which features are used the most in the trees that has lower error rate.

2.5.4 Imbalanced Data Classification

Once we have identified potential predictors and classification methods, we applied those methods to our data set. We have immediately seen that the prediction accuracy of one class is significantly lower than the other. This occurs usually when number of observations of one class is much larger than another. It is an expected situation for a network data set, since the number of lost packets in the network is expected to be significantly lower than the number of delivered packets. In our data set, 40326 of all 225004 packets are lost packets. Although our data set is not highly imbalanced, it still affects the performance of classification algorithms. Standard classification algorithms usually have bias towards the larger class, hence the accuracy of smaller class suffers significantly [40] .

For example; when we apply logistic regression to our data set we get the following results:

Classification accuracy of lost packets:	99%
Classification accuracy of delivered packets:	13%

Although we can correctly classify 99% of the delivered packets, 87% of the lost packets are misclassified. This is not a desirable result. We need to solve imbalanced data problem.

There are many well-known methods to tackle this problem. It is shown several times that a balanced data set provides improvements on overall classification accuracy [55], [56], [57]. These results suggest that balancing the data set through sampling methods might be a good way to solve the imbalanced data problem.

There are two main simple sampling methods suggested: random oversampling and undersampling. Random oversampling achieves a certain level of class balance through adding randomly selected samples of smaller class data to original data set. Final data set may have a lot of duplicates from smaller class observations. On the other hand, random undersampling randomly removes larger class observations from the data set until a certain level of class balance is reached. Both methods are very simple and easily provide a balanced data set, however, each method introduces its own problems. When oversampling is used, adding replicated data to the data set might lead to overfitting [58]. The problem is more obvious in the case of undersampling. Removing observations from the original data set might cause learning algorithm to miss some important concepts. Because of these issues more intelligent sampling methods are proposed.

Synthetic minority oversampling technique (SMOTE) improves the simple oversampling method by creating synthetic observations instead of replicating smaller class observations. Synthetic observations are created around the neighborhood of smaller class observations. This method is shown to achieve better classification performance than simple over or undersampling [59].

Another noteworthy sampling method is repeated random sub-sampling. This method divides the training data into sub-samples where each of these sub-samples contains equal number of observations from each class. Sub-samples are formed by sampling N observations from each class without replacement where N is the number of smaller class observations. Randomly sampled larger class observations are removed from the original data set after each sub-sample is formed. Sub-sampling continues until original data set doesn't contain any larger class observations. For example, if our data set contains N observations from one class and $4N$ observations from other class, then we will have 4 sub-samples each containing same smaller class observations

but different larger class observations. Then, a classifier is trained on every sub-sample and the final classification result is determined by voting. This method was shown to be very successful handling class imbalance classification problem [60].

Besides sampling methods, cost-sensitive learning methods can be used to handle class imbalance. These methods assign distinct costs to misclassifying observations [61]. Typically, smaller class observations are given higher cost of misclassification. This way classifier will try harder to learn smaller class concepts that eliminates initial disadvantage of those observations.

Our empirical studies suggested that repeated random sub-sampling method performs the best among the methods described above. Therefore, we will use this method to overcome imbalanced data problem in this thesis.

2.5.5 Feature Selection

Identification of the potential predictors is discussed in section 2.5.2. We suggest using one geographical location predictor that is the distance between the sender and the receiver nodes and four traffic related predictors for each bin. We extracted 60 bins of traffic predictors where one bin is 100 seconds long. These predictors reflect 100 minutes of historical traffic activity related to one packet. These values are derived empirically. 60 bins yields 240 features in total. Therefore, we have very high-dimensional data. It is shown that having large number of features decreases running time of a classification algorithm significantly [62]. Our feature set may contain irrelevant or redundant features. Removing these features often improves predictive accuracy and running time performance of the classifier and comprehensibility and interpretability of the results [63], [64].

We need to identify and remove redundant and irrelevant features. Irrelevant features do not affect the target concept. Response variable has no relation with an

irrelevant feature in any way. Redundant features, on the other hand, do not add anything new to the target concept. Another feature already captures the target concept in the same way as a redundant feature.

We also have the idea of representing the past traffic patterns with general statistics instead of using all the bin information. This way we can reduce the number of features significantly. This idea is basically taking the mean and standard deviation of four aforementioned traffic pattern predictors over all the time bins. It is a way of representing traffic patterns with less predictors, however we lose some information by doing that. We will test the trade-off between using individual bin data and averaged statistics derived from bins.

Section 2.5.7 discusses the details of experiments and tests we conducted in order to detect redundant and irrelevant features.

2.5.6 Evaluation of Classification Performance

Receiver operating characteristic (ROC) is a widely used, standard method to asses the performance of a classifier [65]. We use area under the ROC curve (AUC) metric for performance assessment. This is an accepted performance metric for a ROC curve [66]. The greater the AUC value is, the better the classifier performs. However, in imbalanced data classification, AUC is not enough. For the case, when one class outnumbers the other class significantly, high classification accuracy only on larger class may result in a very good AUC value. We want both of the class accuracies to be high. Therefore, we also report the classification accuracies of both classes separately and use this information to evaluate a classifier's performance.

In summary, we use following metrics to evaluate the performance of a classifier:

1. AUC

2. Classification accuracy of delivered packets ($Y = 0$)

3. Classification accuracy of lost packets ($Y = 1$)

2.5.7 Computational Experiments and Results

Our experiments use a 64-bit Ubuntu 12.04 system with 16 GB RAM and 4-core CPU (Intel i7, 2.7 GHz). All the experiments are implemented with python [67] programming language. We use "numpy" python package [68] for data processing and "scikit-learn" python package [69] implementations of classification algorithms with default settings. Repeated random sub-sampling method is implemented in python by the author of this thesis.

2.5.7.1 Feature Selection

Since the number of rows and predictors are very large in our data set, our initial trials of running logistic regression failed. Our empirical studies showed that logistic regression performs poorly on large and high-dimensional data sets in terms of computational time. Therefore, we decided not to use logistic regression unless we reduce the size of feature vector by removing redundant and irrelevant predictors.

In order to detect redundant and irrelevant predictors, we introduce the following claim; traffic activity that is too far away from the time of the packet may be irrelevant to the QoS of the packet. Therefore, we designed the following experiment:

- First, run the classifier with full feature set, using 60 bins.
- Start removing features of bins one by one starting from the furthest bin from the packet (sixtieth bin).
- Compare the classification accuracies.

Random forest and classification tree methods are employed for his test. For

each run, repeated random sub-sampling is used. In order to estimate the classifier performance, we divide the data into training and test data sets. We use randomly selected two-thirds of the original data set for training and the remaining observations for testing. For each run, we use the predictors identified in section 2.5.2. Results of this experiment are summarized in Figure 2.12. It is clearly seen that adding more bins after the eighth bin doesn't improve the classification accuracy. Therefore, we conclude that using traffic pattern predictors of first eight bins (i.e 800 seconds before the packet capture time), is sufficient to make a good prediction. Earlier traffic patterns has little or no effect on packet level QoS or they are redundant.

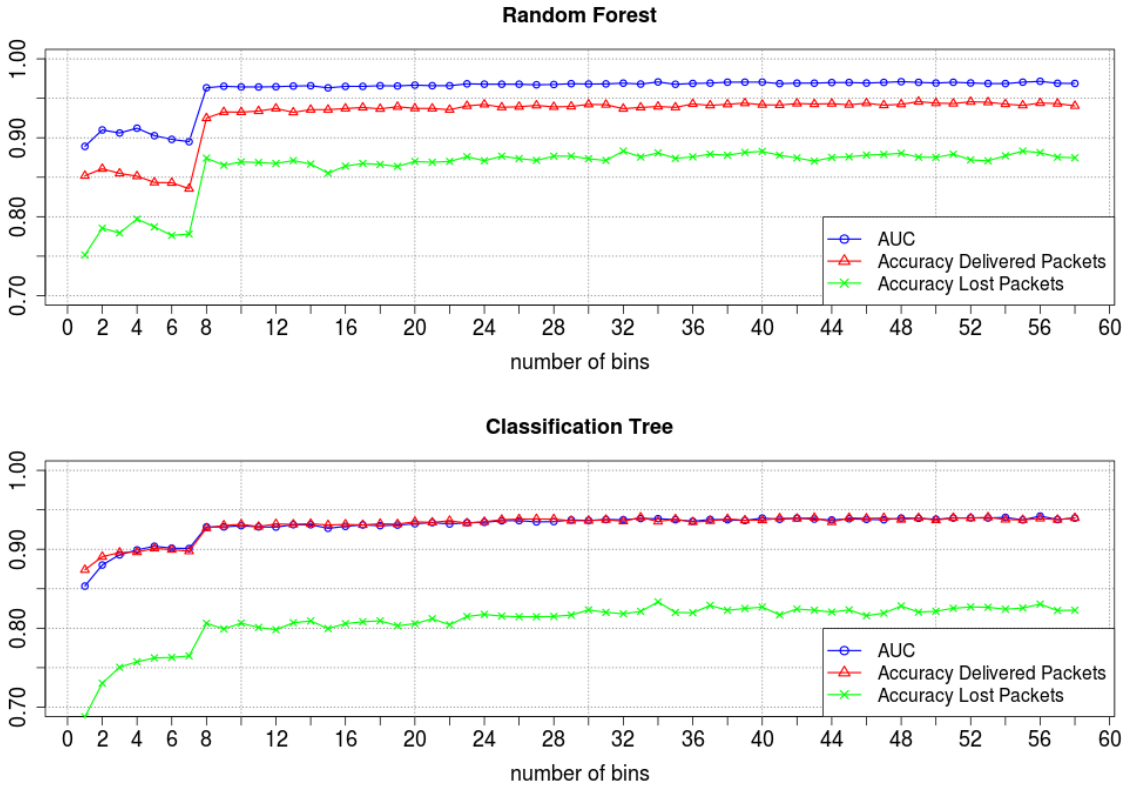

Figure 2.12: Plots Showing the Effect of Number of Bins on Classification Accuracy.

Idea of representing bins with averaged statistics is introduced in section 2.5.5. We expand the above experiment as follows to evaluate this idea:

- First, run the classifier with full feature set, using 60 bins.
- Start removing features of bins one by one starting from the furthest bin from the packet (sixtieth bin).
- Compute mean and standard deviation for each of four aforementioned traffic pattern predictors over the removed bins. This will add 8 new predictors replacing removed bin predictors.
- Compare the classification accuracies.

Other steps of the experiment remain the same. Results are shown in Figure 2.13. We see a very similar pattern to Figure 2.12. More importantly, we want to see if adding mean and standard deviation statistics of removed bins will increase classification accuracy. When less than eight bins are used, this process helps improving accuracy. However, adding derived statistical values of bins earlier than eighth bin doesn't improve accuracy. We have strengthen our earlier conclusion, which was earlier traffic patterns has little or no effect on packet level QoS when first eight bin of traffic pattern predictors are used. Moreover, we can conclude that if we want to decrease the size of our feature set even further, replacing all or most of individual bin values with averaged statistics may be a good option. It still provides a reasonable accuracy.

2.5.7.2 Evaluation of Classification Methods

Finally, we will test all the classification algorithms introduced in this thesis, including logistic regression, to evaluate their performance with the optimal feature set. Each algorithm is run ten times to construct confidence intervals for performance metric estimations. For each run, repeated random sub-sampling is used with 8 bins and total 33 predictors. In order to estimate classifiers' performances, we divide the data into training and test data sets. We use randomly selected two-thirds of the

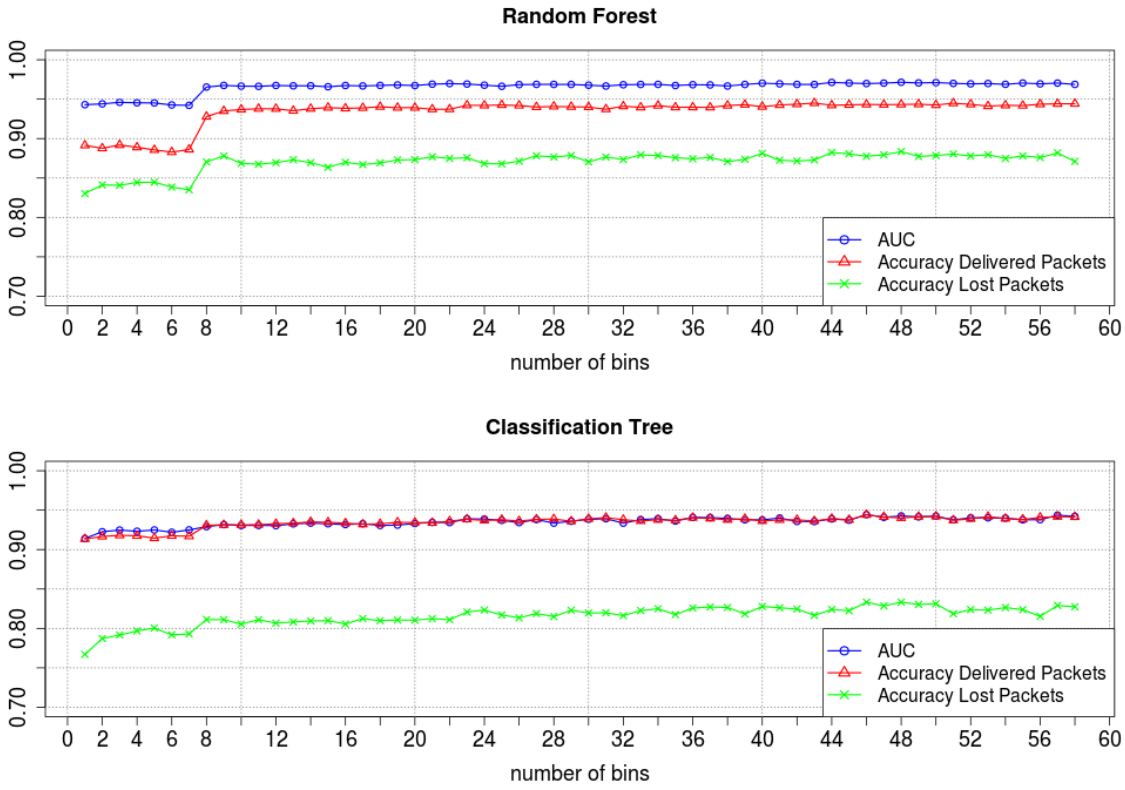

Figure 2.13: Plots Showing the Effect of Number of Bins on Classification Accuracy. Predictors for Bins That Are Not Used Directly Are Replaced by the Mean and Standard Deviation Statistics Computed over All of Them.

original data set for training and the remaining observations for testing. Results are summarized in Table 2.6. Random forest clearly outperforms other methods in overall performance, both in terms of accuracy and runtime.

2.5.7.3 Predictor Importances

We will also need to report importances of predictors to help network administrators in their decision making process. Importance of a predictor may be considered as an indication of its impact on packet level QoS. Random forest can report importances of predictors. Table 2.7 shows the relative importances of predictors we identified as relevant. Results suggest that traffic pattern of the link between the

Table 2.6: Evaluation of the Classifiers. 32 Traffic Predictors and 1 Geographical Location Predictor Are Used. Result Are Presented Using 99% Confidence Intervals.

	Logistic Regression	Classification Tree	Random Forest
AUC	0.72 ± 0.00	0.93 ± 0.00	0.96 ± 0.00
Classification accuracy (Y=0)	0.74 ± 0.01	0.93 ± 0.00	0.92 ± 0.00
Classification accuracy (Y=1)	0.58 ± 0.01	0.81 ± 0.00	0.87 ± 0.00
Runtime (seconds)	45.15 ± 2.33	19.44 ± 0.25	16.13 ± 0.21

sender and the receiver nodes has the biggest impact on QoS. Other predictors are also important for QoS prediction.

Table 2.7: Relative Importances of Relevant Predictors. Values Are Averaged over 10 Identical Runs of the Algorithm. Variations Between Runs Are Insignificant, Thus Not Reported.

	Importances
Traffic pattern of the sender node	0.19
Traffic pattern of the receiver node	0.16
Traffic pattern of the link between the sender and the receiver nodes	0.28
Traffic pattern of the multicast group	0.25
Geographical distance between the sender and the receiver nodes	0.12

2.5.7.4 Future QoS Prediction

There still remains one question to answer. How much in the future can the QoS be predicted with the desired accuracy? To answer that question, we design the following experiment:

- First, use traffic pattern predictors that covers the time between t and $t-(n \times t_{bin})$, where t is the packet capture time, n is the number of bins used and t_{bin} is the time span of one bin. Evaluate the classification performance using only these predictors.
- Shift the time by t_{bin} to cover traffic patterns between $t-t_{bin}$ and $t-(n \times t_{bin})-t_{bin}$. Evaluate the classification performance using only these predictors.
- Continue shifting the traffic pattern predictors' starting and ending time until we hit the end of stored traffic pattern data time.
- Compare classification performances.
- Repeat the above process for different n values.

In this experiment, we don't use geographical distance predictor because we don't know where the nodes will be in the future. We increase the number of bins used from 1 to 60 one by one. We repeat the steps described above for all the number of bins. Data set is divided into training and the test sets using the same logic as previous experiments. Repeated random sub-sampling is used for each run. Results of this experiment are presented in Figure 2.14. AUC values are obtained for all the possible number of bins and time shift combinations. In the figure, blue color represents high AUC and red color represents low AUC. The range of colors between blue and red correspond to the AUC values between the best and worst values. Horizontal axis of the figure shows how long in the future the prediction is made for. For example, first

column in the figure shows the AUC values of the prediction for 100 seconds in the future with different number of bins used. Figure shows that, when a longer term prediction is made, usually more number of bins required for a good prediction. If enough number of traffic pattern predictors are used, we can successfully predict QoS of a packet 3500 seconds (approx. 58 minutes) ahead of time. This will allow network administrators to foresee potential QoS issues in the network approximately one hour before they happen so that they will have enough time to prepare and prevent those issues.

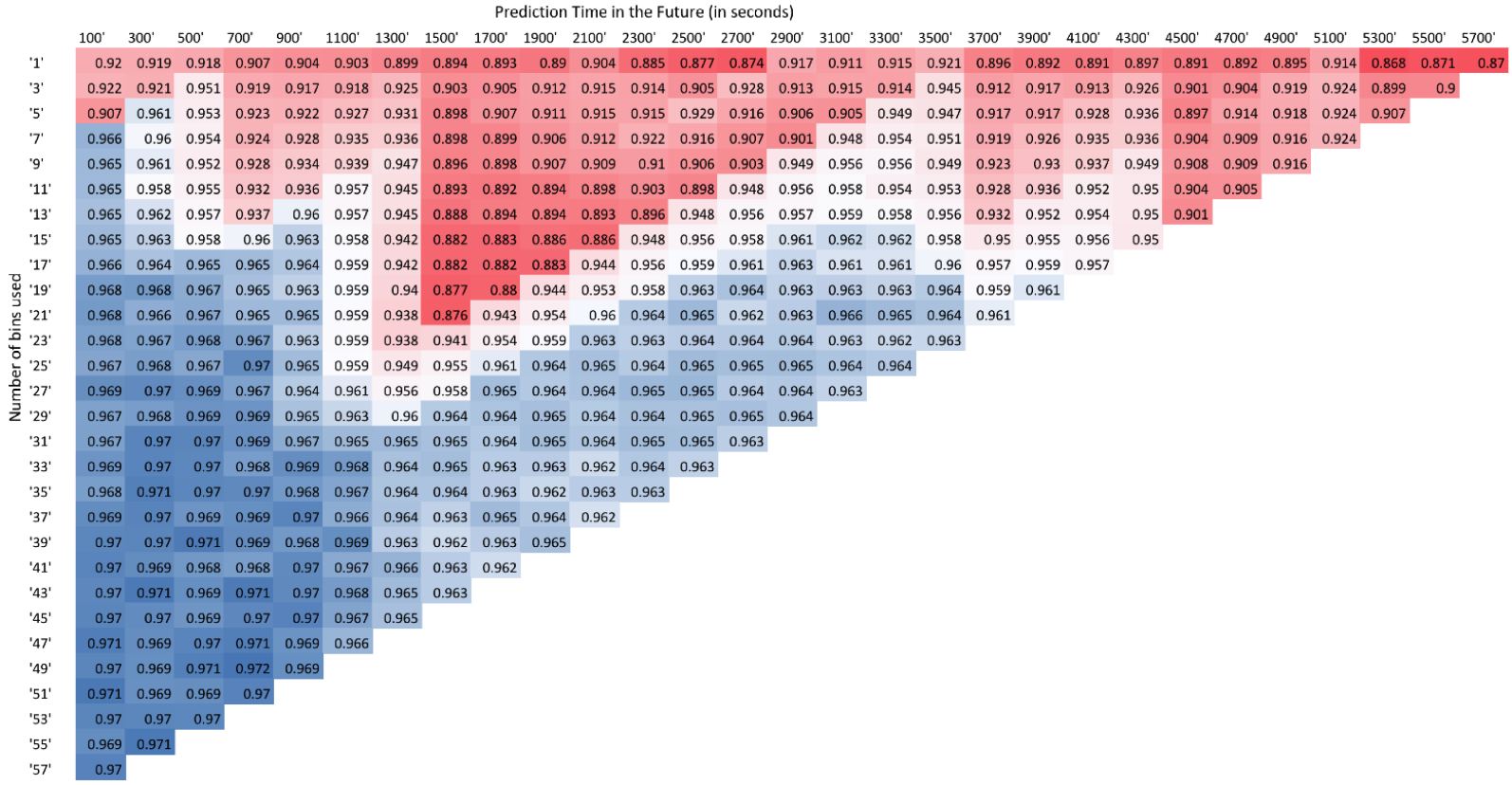

Figure 2.14: Results of the Analysis on the Effect of Bins' Time Proximity to the Packet and Number of Bins Used on Classification Accuracy.

2.6 Knowledge Discovery through Treed Logistic Regression (TLR)

Our experiments show that random forest provides a very good prediction accuracy within our prediction framework. If the task at hand is QoS prediction with high accuracy, random forest and our prediction framework whose steps laid out above provides a good solution. However, it is very hard to interpret the results of random forest. Ultimately, we want to learn the effects of predictors on QoS. Logistic regression, described in section 2.5.3.1, is a linear model that defines the response variable as a function of predictors. Coefficients of predictors in a logistic regression model are used to understand the effect of one unit change in any predictor on the response variable. In other words, a logistic regression model can be used to measure how much one unit increase in sender, receiver, link or group volume or distance between the sender and the receiver nodes will increase the probability of a packet being lost. Logistic regression provides invaluable information for network planning. Therefore, we want to use logistic regression models for knowledge discovery. However, it is so naive to believe that predictors and response variable are linearly related in a very dynamic and complex environment such as communication networks. Our experiments discussed in section 2.5.7 showed that logistic regression is not very successful at predicting QoS on our data set. However, there may be subsets of data on which a good linear model can be fitted. There may be time periods where traffic patterns described in section 2.5.2 follows a uniform trend or sub-networks presenting different patterns than the overall network. It is intuitive to believe that different time frames or node groups show different traffic patterns. Therefore, one linear model may not be fitted to all data but subsets of data defined by some features of the network, such as subnetworks or time of the day, may be represented by a linear model. This would allow us to reveal linear relations between predictors and the response variable

with multiple linear models. In order to find those subsets we propose a method that incorporates the basic principles of classification tree and logistic regression models.

Combining classification tree methods with linear models is not a new idea. The term "treed regression" is first coined in 1996 [70], where authors describe a tree method with simple linear regression in each terminal node. In [71] a variation of "treed regression" method is introduced. Original splitting criterion of regression trees described by [53] is adopted. Standard cost-complexity pruning is then used to prune the tree. A uni-variate linear regression model is fitted to the observations in each terminal node to assign a label to that node. This method is used to estimate a continuous response variable. On the other hand, [72] introduces "logistic model trees (LMT)" for classification task, which uses a similar idea as "treed regression". A classification tree is build using the C4.5 splitting criterion [73]. A logistic regression model is fitted at each node of the tree on the data associated with that node using LogitBoost algorithm [74] during tree growing step. Then, tree is pruned using the CART pruning algorithm [53].

Derived from the ideas presented above, we propose a new algorithm that combines tree structure and logistic regression model. Our algorithm differs from the existing work in three ways: (i) it uses AUC model evaluation metric as a part of splitting criterion; (ii) it uses separate feature sets for splitting and logistic regression fitting; (iii) it supports multiway grouped nominal feature splits.

Communication network data sets present some properties that lead us to use some features for splitting the data and some other features for fitting linear models to predict QoS metrics. For example, data coming from different nodes in the network may show different patterns. This may have an impact on QoS of a packet but it is not appropriate to use sender or receiver node identifier in a linear model. Moreover, the effect of packet sending time might not be linearly related with QoS metrics.

However, different times in a day may show different QoS patterns. Our empirical studies showed that these claims are viable. Different times of the day and different nodes or node groups show different QoS and traffic patterns. Hence, we will use following features of a packet for splitting the data set:

- sending time
- sender node identifier
- receiver node identifier

Features described in section 2.5.2 will be used to fit logistic regression at the nodes of the tree.

Our algorithm works as follows:

- Tree growing starts with building a logistic regression model on all training data set observations. The node contains all the observations is called the root node.
- All splitting features are tested to find the best split at the root node and the data are split into two or more children nodes.
- Logistic regression models are built at new children nodes on the data associated with them.
- Splitting continues recursively until the stopping criteria are met.
- No pruning performed.

Once the tree is grown, classification of a new observation is performed by simply finding the terminal node assignment of the observation and using the logistic regression model associated with that terminal node.

2.6.1 Splitting Criterion

We use AUC binary model evaluation metric for splitting. AUC is an indication of how well a logistic regression model performs on unseen data. When a model is tested with the training data set, AUC metric can be interpreted as the measurement of how well the model fits to the data. We will use this idea to decide which split is the best split. The steps of testing a split are listed below:

- Fit a logistic regression on each newly formed children node data.
- For each children, obtain an AUC score by testing the model with the same data as the model is trained.
- Calculate weighted average of children AUC scores.
- Take the difference between parent AUC score and averaged children AUC scores.

The steps listed above are repeated for all possible splits of all splitting features and a split score is calculated for each. Split with the greatest split score is selected as the best split and the data of the parent node is split accordingly.

2.6.2 Handling Nominal Splitting Features

Handling nominal features is not a trivial task in tree methods. Two of the most well known tree algorithms handle nominal features differently. C4.5 algorithm makes a multi-way split (one branch for each value of the nominal feature), on the other hand, CART tries all the possible two-way splits and selects the best one. CART forces binary splits whereas C4.5 allows multi-way splits. However, neither method is preferred in our algorithm. We have two nominal features; sender and receiver node identifiers. We don't want to have one branch for each value of these features

since nodes usually form groups in networks and identifying those groups is more interesting. Trying all the possible two-way splits is more desirable. However, its computational complexity limits us. This method requires $2^{n-1} - 1$ tests per each feature, where n is the number of unique values of the nominal feature.

Thus, we propose a new method to handle nominal features. Our proposed algorithm first makes a multi-way split as in C4.5. Then, it identifies two children nodes whose merging gives the best improvement on AUC score. Merge score is calculated by obtaining two AUC scores for two nodes and taking weighted average of those and comparing this average with the AUC score calculated on merged data. All the possible two-node merges are tested and the one with the highest merge score is selected. Merging continues until there is no merge with positive merging score or split is binary. This way, the tree can have multi-way splits where each branch of a split represents a group of nodes. This algorithm requires $\binom{n}{2} + \binom{n-1}{2} + ... + \binom{3}{2}$ tests per each nominal feature. For instance, when $n = 14$, trying all two-way splits requires 8191 tests whereas our proposed method requires 454.

2.6.3 Stopping Criteria

In order to make a split, all the children nodes resulting from that split must meet the following conditions. Otherwise the split is discarded.

- Node should contain at least a predefined number of observations (usually 15-20).
- Node should contain at least two observations from each response variable class.

Moreover, a node is not split further if the predefined maximum depth of the tree is reached or splitting doesn't provide any improvement on AUC. Maximum depth is a parameter to our algorithm.

2.6.4 Computational Experiments and Results

Experiments in this section use the computer introduced in section 2.5.7. Our treed logistic regression algorithm (TLR) is implemented in python programming language. Logistic regression implementation of "scikit-learn" python package is used with the default settings. Instead of repeated random sub-sampling method, simple over and under sampling methods are used to overcome the class imbalance issue due to computational constraints.

TLR requires fitting many logistic regression models at different levels of the tree. Therefore, running time of one logistic regression algorithm is very crucial determining the computational cost of TLR. Our experiments show that running time of the logistic regression algorithm increases as the number of features increases. We want to use as less features as possible for logistic regression fitting. Hence, we decided to use only one bin of traffic pattern related predictors along with one geographical location related feature with logistic regression. Our tree structure tolerates fitting simpler models at terminal nodes. In order to determine which bin to use, we constructed a similar table to Figure 2.14 for logistic regression. It showed that eighth bin gives the highest classification accuracy when used alone. Thus, we will use predictors of eighth bin in our experiments.

Maximum depth of TLR trees can be specified by users. If a big enough maximum depth parameter is set, tree grows to its maximal size until stopping criteria are reached. In order to understand how big the tree grows, we run the algorithm several times on randomly selected two thirds of all the observations and let the tree grow to its maximal size. At each run maximum depth of grown tree might differ due to sampling variations. Results are presented in Figure 2.15. As we see, maximum depth ranges from 14 to 22. It tends to take values between 16 to 18 most of the time.

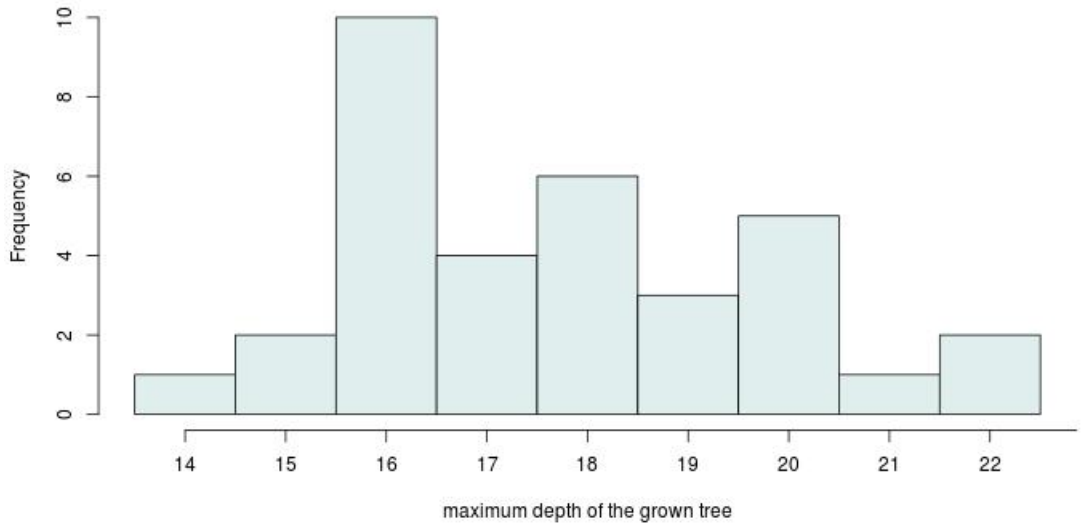

Figure 2.15: Histogram of Maximum Depths of Fully Grown Trees. Sampling Variations Result in Different Maximum Depths at Different Runs of the TLR Algorithm.

However, we might not need to grow the tree to its maximal size in order to obtain best classification accuracy. In fact, we want as small trees as possible with satisfactory accuracy to achieve interpretability. Therefore, we experimented with various maximum depth settings to study the effect of depth of the tree on TLR's classification accuracy. Results are summarized in Figure 2.16. Results suggest that classification accuracy of TLR increases as the tree depth increases until a certain threshold. Beyond that threshold, accuracy stabilizes.

Based on these results, we decide on a maximum depth setting that will provide a desired classification accuracy while keeping the tree small enough for easy interpretation. Keeping the tree small by setting a maximum depth that will stop tree growth early will also prevent overfitting [53]. Figure 2.16 shows that tree with maximum depth 9 gives the highest classification accuracy on test data. However, this tree will produce at least $2^9 = 512$ terminal nodes, which is hard to interpret. Based on heuristic evaluation, we will set maximum depth parameter to 4, which pro-

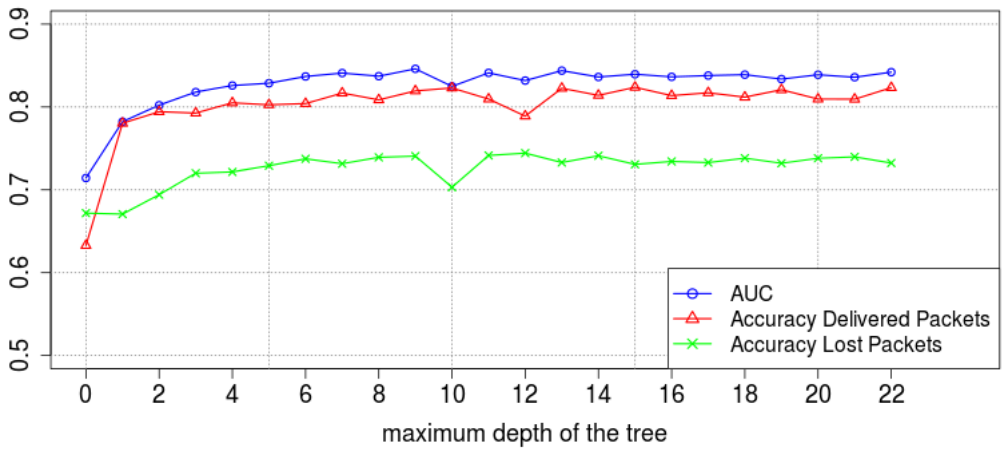

Figure 2.16: Plot Showing the Effect of Maximum Tree Depth on Accuracy of TLR.

vides a reasonably good classification accuracy and relatively easy to interpret tree structures.

In order to evaluate classification accuracy of TLR algorithm, we will compare it with other well known classification methods introduced in section 2.5.3. We run the algorithm for 10 times to obtain mean and standard deviation of evaluation metrics. At each run, algorithm is trained using randomly selected two thirds of all observations and tested with the remaining observations that are not used for training. To make a fair comparison, splitting features of TLR which are sending time of the packet, sender node identifier of the packet and receiver node identifier of the packet, are added to the list of the predictors used in other classification algorithms.

Table 2.8, which is extended from Table 2.6, summarizes the results. TLR provides better classification accuracy than logistic regression. Although its accuracy is worse than classification tree and random forest, TLR method has the advantage of interpretability. Linear models at the terminal nodes of TLR can be used to explain relations between traffic patterns / geographical location and packet level QoS. In addition, structure of the resulting tree reveals node clusterings and times of the day

when relation between network traffic and packet level QoS differs. This will help network planning and root cause analysis.

Table 2.8: Performance Comparison of the TLR Method with Other Well Known Classification Methods. Result Are Presented Using 99% Confidence Intervals.

	TLR	Logistic Regression	Classification Tree	Random Forest
AUC	0.82 ± 0.00	0.72 ± 0.00	0.94 ± 0.00	0.97 ± 0.00
Classification accuracy (Y=0)	0.80 ± 0.01	0.69 ± 0.01	0.94 ± 0.00	0.95 ± 0.00
Classification accuracy (Y=1)	0.72 ± 0.01	0.59 ± 0.01	0.84 ± 0.00	0.89 ± 0.00
Runtime (seconds)	764.5 ± 62.7	67.91 ± 7.32	25.41 ± 1.28	18.29 ± 0.66

Our experiments with TLR revealed that on the day whose data are analyzed, network traffic and QoS patterns show different behavior in the afternoon than in the morning. The reason of this may be a change of personnel or hardware occurred in the afternoon. We have also seen from the TLR tree structure that packet loss rate increases towards the end of the test day. Moreover, TLR helps to discover groups of nodes whose traffic and QoS patterns are different than the general trends. These findings are reported to the network administrators for further investigation since an abnormal pattern might often indicate a problem.

Chapter 3

CONCLUSION AND FUTURE WORK

3.1 Summary

In this thesis, we outlined the details of a comprehensive QoS analysis in mission critical communication networks (MCCNs). In chapter 1, we described the QoS analysis task in MCCNs and our research objectives. We explained why MCCNs present unique challenges for QoS analysis. Three main components of QoS analysis in MCCNs, which are QoS measurement, visualization and prediction, are covered. Goals and challenges of each component are explained. We provided a detailed literature review of QoS research. Expected original contributions of the thesis are listed to conclude chapter 1.

Chapter 2 covers the details of our proposed solutions to the QoS related problems in MCCN introduced in previous chapter. First, we explained the data studied in this thesis and environment where the data are collected. Then, we laid out the details of our QoS analysis framework. Within this framework we have the following components; QoS measurement, QoS visualization, QoS prediction and knowledge discovery.

We proposed an efficient, passive one-way QoS measurement algorithm in section 2.3.2. We measure the QoS in the network using flow formatted network data. We demonstrated the efficiency and accuracy of our measurement algorithm and compared it with the existing methods.

Next, we listed the details and capabilities of our QoS visualization platform in section 2.4. We provided screenshots to demonstrate the capabilities.

Finally, we covered our QoS prediction research in section 2.5. We defined QoS

prediction task in MCCNs. Well known classification algorithms are introduced. Next, we identified which pieces of data collected in the network should be used for prediction. Imbalanced data classification problem is defined and possible solutions to this problem are explained. We designed numerous experiments to evaluate existing classification algorithms in terms of their prediction performances. Then, we introduced a new method which combines tree based algorithms with logistic regression in order to provide a better knowledge discovery and prediction solution. This method is rigorously tested and its performance is compared with other well known classification algorithms.

In summary, this thesis addresses all QoS related research issues in MCCNs. We provided a comprehensive roadmap to QoS analysis in MCCN where problems are defined and solutions are suggested at each stage.

3.2 Assessment of the Research Questions and Expected Original Contributions

Research goals and expected original contributions of this thesis are discussed in section 1.4. Throughout the rest of the thesis we attempted to answer those research questions.

We proposed a new QoS measurement method in section 2.3.2 that uses flow formatted data and hashed packet content instead of packet data. This method enables much faster QoS measurement while not compromising the accuracy.

Section 2.5 explains our QoS prediction study. We laid out the steps of our QoS prediction algorithm with which very accurate prediction is achieved. We concluded that traffic patterns of the nodes and links in the network and geographical distance between the nodes are good predictors of packet level QoS. Hence, this information should be collected and stored for prediction. Based on our computational experi-

ments explained in section 2.5.7, we concluded that accurate QoS prediction might be achieved as much as one hour beforehand. We also showed that approximately 13 minutes of historical traffic data are sufficient to make immediate QoS prediction with high accuracy.

Within our QoS prediction toolkit, we suggested a novel classification method called TLR in section 2.6, which improves accuracy of simple linear models with added advantage of tree based algorithms. TLR is able to explain relations between predictors and response variable as well as clustering structures in the data while making QoS predictions, which makes it a powerful tool for QoS analysis in MCCNs. As a part of TLR algorithm, we also proposed a new way to handle categorical splitting variables in tree based methods.

We demonstrated our QoS visualization platform in section 2.4, which delivers all the required capabilities.

Our another important contribution is explaining the goals and challenges of the QoS research and outlining the detailed steps of developing a computational QoS analysis framework for MCCNs.

Finally, all these components listed above comes together and forms a complete computational QoS analysis framework for MCCNs, which was our main objective.

3.3 Future Work

There are four main aspects of research presented in this thesis. We may point out several interesting directions for future work for each of those aspects.

3.3.1 QoS Measurement

We have demonstrated an approach for calculating QoS metrics using network flow data that is both scalable (because of the significant size reduction from packet

capture to flow) and accurate (because our flow data ensures that each packet with its hashed content is accounted for). In order to reduce the amount of time required to present QoS calculations to users, several areas of the system can be optimized. The size/accuracy trade off could be further explored by slightly increasing the aggregation of flows (e.g. two, three, or N packets per flow). Data collection and conversion points can be optimized, and data storage devices can be improved for better performance in disk IO. However, as the amount of network traffic data increases, techniques for minimizing the amount of data required to produce accurate QoS measurements become more important.

3.3.2 QoS Visualization

Our visualization platform covers our essential needs. However, several more capabilities can be added. Topology view may be simplified by grouping nodes to provide clearer visualization when number of nodes on the network grows larger. Filtering options may be expanded to support more protocol types, port numbers or IP addresses. Drill-down view may be added to geographical map view. More importantly, QoS prediction tools can be integrated into the visualization platform. Prediction results or prediction confidence levels by node may be visualized. Moreover, interactive visualization of TLR trees may be a good extension to the current platform.

3.3.3 QoS Prediction through Packet Classification

We have outlined the steps of an accurate QoS prediction framework for MCCNs. In order to improve computational performance, several improvements might be suggested. A distributed implementation can speed up the prediction algorithms significantly. Prediction may be performed at each node or a group of nodes in the network. This will eliminate the requirement of collecting all the data from all the

nodes to a central location. It will also require processing smaller amounts of data locally, thus decrease computation times. We can also experiment with aggregated prediction, where packet loss rate aggregated over a period of time is predicted rather than individual packet losses. This would allow faster predictions while sacrificing precision. This trade-off should be studied. Moreover, we can further explore the effect of bin size over prediction accuracy and running time of the algorithms. Increasing the size of the bin may reduce the number of features, hence improve computational performance. However, it might decrease the prediction accuracy.

3.3.4 Knowledge Discovery through TLR

We proposed a new classification method called TLR, which is built upon existing treed regression methods. There are several aspects of this method left unexplored. This thesis will provide basis for future research in several areas regarding TLR algorithm. Stopping criteria of the TLR may be optimized by experimenting with different "minimum number of observations" settings. Maximum depth of the tree parameter may be discarded to grow the tree to full extent and then pruning should be performed. Effect of pruning on tree size and classification accuracy can be explored. Computational performance of the algorithm may be improved by fitting simpler linear models at the terminal nodes (by selecting a subset of features) but trade-off between classification accuracy and computational performance should be studied. Furthermore, a parallel implementation of the algorithm (by parallelizing the testing of features to find the best split) might improve the computational performance.

Tree based models are known for their instability [75]. Stability of a classification algorithm can be defined as its ability to generate repeatable results, given different subsets of data drawn from the same population [76]. To extend our work in this thesis, stability of the TLR algorithm can be analyzed and improved.

REFERENCES

[1] ETSI, "General aspects of quality of service and network performance in digital networks," European Telecommunications Standards Institute (ETSI), Tech. Rep. ETR 003, Aug. 1990.

[2] A. Campbel, G. Coulson, and D. Hutchison, "A quality of service architecture," *SIGCOMM Computer Communication Review*, vol. 24, no. 2, pp. 6–27, Apr. 1994.

[3] J. Postel, "Internet protocol," Information Sciences Institute, University of Southern California, Tech. Rep. RFC 791, Sep. 1981.

[4] "tcpdump/libpcap." [Online]. Available: http://www.tcpdump.org/

[5] G. Sadasivan, V. Valluri, M. Djernaes, and J. Quittek, "Cisco systems netflow export version 9," in *RFC 3954*, Oct. 2004.

[6] G. Sadasivan, N. Brownlee, B. Claise, and J. Quittek, "Architecture for IP flow information export," in *RFC 5470*, Mar. 2009.

[7] F. Ricciato, F. Strohmeier, P. Dorfinger, and A. Coluccia, "One-way loss measurements from IPFIX records," in *Measurements and Networking Proceedings (MN), 2011 IEEE International Workshop on*, Oct. 2011, pp. 158 –163.

[8] T. Zseby, S. Zander, and G. Carle, "Evaluation of building blocks for passive one-way-delay measurements," in *Proceedings of Passive and Active Measurement Workshop (PAM 2001)*, 2001.

[9] W. Jiang and H. G. Schulzrinne, "Qos measurement of real time multimedia services on the internet," Department of Computer Science, Columbia University, Tech. Rep. CUCS-015-99, 1999.

[10] I. Cozzani and S. Giordano, "A passive test and measurement system: traffic sampling for qos evaluation," in *Proceedings from the Global Telecommunications Conference*, vol. 2, 1998, pp. 1236–1241.

[11] A. E. Conway, "A passive method for monitoring voice-over-IP call quality with ITU-T objective speech quality measurement methods," in *Proc. IEEE Int. Conf. Communications*, vol. 4, 2002, pp. 2583–2586.

[12] Y. Gu, L. Breslau, N. Duffield, and S. Sen, "On passive one-way loss measurements using sampled flow statistics," in *INFOCOM 2009, IEEE*, Apr. 2009, pp. 2946 –2950.

[13] F. Ricciato, F. Strohmeier, P. Dorfinger, and A. Coluccia, "One-way loss measurements from ipfix records," in *Measurements and Networking Proceedings (M N), 2011 IEEE International Workshop on*, Oct. 2011, pp. 158 –163.

[14] M. Lee, N. Duffield, and R. R. Kompella, "Opportunistic flow-level latency estimation using consistent netflow," *IEEE/ACM Trans. Netw.*, vol. 20, no. 1, pp. 139–152, Feb. 2012.

[15] "ntop." [Online]. Available: http://www.ntop.org/

[16] "Argus." [Online]. Available: http://www.qosient.com/argus/

[17] "Wireshark." [Online]. Available: http://www.wireshark.org/

[18] "Nfsen." [Online]. Available: http://nfsen.sourceforge.net/

[19] L. Zeng, B. Benatallah, A. H. H. Ngu, M. Dumas, J. Kalagnanam, and H. Chang, "Qos-aware middleware for web services composition," *IEEE Trans. Software Eng*, vol. 30, pp. 311–327, 2004.

[20] L. Yang, Y. Dai, and B. Zhang, "Performance prediction based ex-qos driven approach for adaptive service composition," *Journal of Information Science and Engineering*, vol. 25, pp. 345–362, 2009.

[21] J. Zhu, Z. Z. Yu Kan and, and M. R. Lyu, "A clustering-based qos prediction approach for web service recommendation," in *2012 IEEE Ninth International Conference on Services Computing (SCC)*, Jun. 2012, pp. 138–145.

[22] L. Shao, J. Zhang, Y. Wei, J. Zhao, B. Xie, and H. Mei, "Personalized qos prediction for web services via collaborative filtering," in *Proc. of the IEEE ICWS'07*, 2007, pp. 439–446.

[23] H. Madhyastha, T. Isdal, M. Piatek, C. Dixon, T. Anderson, A. Krishnamurthy, and A. Venkataramani, "iPlane: An information plane for distributed services," in *Proc. OSDI*, 2006.

[24] L. Roychoudhuri and E. S. Al-Shaer, "Real-time packet loss prediction based on end-to-end delay variation," in *IEEE Transactions on Network and Service Management*, vol. 2, no. 1, 2005, pp. 29–38.

[25] A. F. Atiya, S. G. Yoo, K. T. Chong, and H. Kim, "Packet loss rate prediction using the sparse basis prediction model," in *IEEE Transactions on Neural NetworksS*, vol. 18, no. 3, May 2007, pp. 950–954.

[26] W. Jiang and H. Schulzrinne, "Modeling of packet loss and delay and their effect on real-time multimedia service quality," in *Proc. NOSSDAV*, Jun. 2000.

[27] C.-S. Yang, Y.-C. Su, and C.-W. Lee, "The analysis of packet loss prediction for gilbert-model with loss rate uplink," in *Proc. 23rd International IEEE Conference on Distributed Systems Workshop*, May 2003, pp. 526–530.

[28] H. Sun, D. D. Perkins, and H. D. Hughes, "Packet loss rate prediction using a universal indicator of traffic," in *IEEE 58th Vehicular Technology Conference*, vol. 5, Oct. 2003, pp. 3541–3543.

[29] A. Pescape, F. Palmieri, G. Iannello, and P. S. Rossi, "End-to-end packet-channel bayesian model applied to heterogeneous wireless networks," in *IEEE Global Telecommunications Conference*, Dec. 2005, pp. 484–489.

[30] P. S. Rossi, G. Romano, F. Palmieri, and G. Iannello, "Joint end-to-end loss-delay hidden markov model for periodic UDP traffic over the internet," in *IEEE Transactions on Signal Processing*, vol. 54, no. 2, Feb. 2006, pp. 530–541.

[31] L. S. Brakmo, S. W. O'Malley, and L. L. Peterson, "Tcp vegas: New techniques for congestion detection and avoidance," in *Proc. ACM SIGCOMM*, 1994, pp. 24–35.

[32] F. Dabek, R. Cox, F. Kaashoek, and R. Morris, "Vivaldi: a decentralized network coordinate system," in *Proc. ACM SIGCOMM*, vol. 34, no. 4. ACM New York, Oct. 2004, pp. 15–26.

[33] P. S. Rossi, G. Romano, F. Palmieri, , and G. Iannello, "Bayesian modelling for packet channels," in *Italian Workshop on Neural Nets*. Springer Verlag, 2003, pp. 285–292.

[34] A. Bremler-Barr, E. Cohen, H. Kaplan, and Y. Mansour, "Predicting and bypassing end-to-end internet service degradations," *IEEE Journal on Selected Areas in Communications*, vol. 21, no. 6, pp. 961–978, Aug. 2003.

[35] E. N. Gilbert, "Capacity of a burst-noise channel," *Bell System Technical Journal*, vol. 39, pp. 1253–1265, Sep. 1960.

[36] X. T. Xunqi Yu, James W. Modestino, "The accuracy of gilbert models in predicting packet-loss statistics for a single-multiplexer network model," in *Proc. IEEE INFOCOM*, vol. 4, Mar. 2005, pp. 2602–2612.

[37] H. R. Mehrvar and M. R. Soleymani, "Packet loss rate prediction using a universal indicator of traffic." in *ICC*. IEEE, 2001, pp. 647–653.

[38] T. H. Szymanski and D. Gilbert, "Provisioning mission-critical telerobotic control systems over internet backbone networks with essentially-perfect qos," *IEEE Journal on selected Areas in Comm.*, vol. 28, no. May, pp. 630–643, Jun. 2010.

[39] E. Rey, D. Mende, H. Schmidt, and M. Luft, "Attacking high speed ethernet links-practical attacks against unencrypted high speed ethernet links," in *ERNW Newsletter 39*, Mar. 2012.

[40] N. Japkowicz, "The class imbalance problem: Significance and strategies," in *Proc. of the International Conf. on Artificial Intelligence*. Citeseer, 2000.

[41] S. Deering, "Host extensions for ip multicasting," Stanford University, Tech. Rep. RFC 1112, Aug. 1989.

[42] M. Crovella and B. Krishnamurthy, *Internet Measurement: Infrastructure, Traffic and Applications*. John Wiley & Sons, 2006.

[43] C. Henke, C. Schmoll, and T. Zseby, "Empirical evaluation of hash functions for multipoint measurements," *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 3, pp. 39–50, Jul. 2008.

[44] S. Alcock, P. Lorier, and R. Nelson, "Libtrace: a packet capture and analysis library," *SIGCOMM Comput. Commun. Rev.*, vol. 42, no. 2, pp. 42–48, Mar. 2012.

[45] "jnetpcap." [Online]. Available: http://jnetpcap.com/

[46] "Silk." [Online]. Available: http://tools.netsa.cert.org/silk/

[47] "gzip." [Online]. Available: http://www.gzip.org/

[48] "Ozone widget framework (owf)." [Online]. Available: http://www.ozoneplatform.org/

[49] "Data-driven documents (d3)." [Online]. Available: http://www.d3js.org/

[50] J. Postel, "User datagram protocol," Information Sciences Institute, Tech. Rep. RFC 768, Aug. 1980.

[51] R. W. Sinnott, "Virtues of the haversine," *Sky and telescope*, vol. 68, p. 158, 1984.

[52] G. Casella and R. L. Berger, *Statistical Inference*, 2nd ed. Cengage Learning, 2001.

[53] L. Breiman, J. Friedman, C. J. Stone, and R. Olshen, *Classification and Regression Trees*, 1st ed. Chapman and Hall, 1984.

[54] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[55] J. Laurikkala, *Improving identification of difficult small classes by balancing class distribution*. Springer, 2001.

[56] G. M. Weiss and F. Provost, "The effect of class distribution on classifier learning: an empirical study," *Rutgers Univ*, 2001.

[57] A. Estabrooks, T. Jo, and N. Japkowicz, "A multiple resampling method for learning from imbalanced data sets," *Computational Intelligence*, vol. 20, no. 1, pp. 18–36, 2004.

[58] D. Mease, A. J. Wyner, and A. Buja, "Boosted classification trees and class probability/quantile estimation," *The Journal of Machine Learning Research*, vol. 8, pp. 409–439, 2007.

[59] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "Smote: synthetic minority over-sampling technique," *Journal of Artificial Intelligence Research*, vol. 16, no. 1, pp. 321–357, 2002.

[60] M. Khalilia, S. Chakraborty, and M. Popescu, "Predicting disease risks from highly imbalanced data using random forest," *BMC medical informatics and decision making*, vol. 11, no. 1, p. 51, 2011.

[61] C. Elkan, "The foundations of cost-sensitive learning," in *International joint conference on artificial intelligence*, vol. 17, no. 1. Citeseer, 2001, pp. 973–978.

[62] M. Dash and H. Liu, "Feature selection for classification," *Intelligent data analysis*, vol. 1, no. 3, pp. 131–156, 1997.

[63] A. L. Blum and P. Langley, "Selection of relevant features and examples in machine learning," *Artificial intelligence*, vol. 97, no. 1, pp. 245–271, 1997.

[64] R. Kohavi and G. H. John, "Wrappers for feature subset selection," *Artificial intelligence*, vol. 97, no. 1, pp. 273–324, 1997.

[65] J. A. Swets, "Measuring the accuracy of diagnostic systems," *Science*, vol. 240, no. 4857, pp. 1285–1293, 1988.

[66] A. P. Bradley, "The use of the area under the roc curve in the evaluation of machine learning algorithms," *Pattern recognition*, vol. 30, no. 7, pp. 1145–1159, 1997.

[67] "Python - version 2.7." [Online]. Available: https://www.python.org/download/releases/2.7

[68] "numpy." [Online]. Available: http://www.numpy.org/

[69] "scikit-learn." [Online]. Available: http://scikit-learn.org/

[70] W. P. Alexander and S. D. Grimshaw, "Treed regression," *Journal of Computational and Graphical Statistics*, vol. 5, no. 2, pp. 156–175, 1996.

[71] M. A. P. William D. Shannon, Maciej Faifer and D. C. Rao, "Tree-based models for fitting stratified linear regression. models," *Journal of Classification*, vol. 19, no. 1, pp. 113–130, 2002.

[72] N. Landwehr, M. Hall, and E. Frank, "Logistic model trees," *Machine Learning*, vol. 59, no. 1-2, pp. 161–205, 2005.

[73] J. R. Quinlan, *C4. 5: programs for machine learning*. Morgan kaufmann, 1993, vol. 1.

[74] J. Friedman, T. Hastie, and R. Tibshirani, "Additive logistic regression: a statistical view of boosting," *The Annals of Statistics*, vol. 28, no. 2, pp. 337–407, 2000.

[75] L. Breiman *et al.*, "Heuristics of instability and stabilization in model selection," *The annals of statistics*, vol. 24, no. 6, pp. 2350–2383, 1996.

[76] P. Turney, "Technical note: Bias and the quantification of stability," *Journal of Machine Learning*, vol. 20, 1995.