BLIND NETWORK TOMOGRAPHY

by

Muhammad Hassan Raza

Submitted in partial fulf llment of the
requirements for the degree of
Doctor of Philosophy

at

Dalhousie University
Halifax, Nova Scotia
July 2011

DALHOUSIE UNIVERSITY

DEPARTMENT OF ENGINEERING MATHEMATICS

The undersigned hereby certify that they have read and recommend to the Faculty of Graduate Studies for acceptance a thesis entitled “BLIND NETWORK TOMOGRAPHY” by Muhammad Hassan Raza in partial fulf llment of the requirements for the degree of Doctor of Philosophy.

Dated: July 18, 2011

External Examiner: _____

Research Supervisor: _____

Examining Committee: _____

_____

_____

_____

Departmental Representative: _____

DALHOUSIE UNIVERSITY

DATE: July 18, 2011

AUTHOR: Muhammad Hassan Raza

TITLE: BLIND NETWORK TOMOGRAPHY

DEPARTMENT OR SCHOOL: Department of Engineering Mathematics

DEGREE: Ph.D. CONVOCATION: October YEAR: 2011

Signature of Author

Table of Contents

List of Tables

List of Figures

Abstract

The parameters required for network monitoring are not directly measurable and could be estimated indirectly by network tomography. Some important aspects of network tomography appear to be unaddressed and left open for further research. These aspects include the assumption of a known routing matrix, errors in the measurement of parameters used as input to network tomography, multi-metric network tomography, and distributed network tomography. These important research issues, related to network tomography, motivated the research in this dissertation.

The f rst contribution of this research presents the application of a sparse matrix-based blind technique such as Non Negative Matrix Factorization (NNMF) to eliminate the unrealistic assumption of a known routing matrix in the network tomography model. The statistical ability of NNMF (with the feature of sparsity) is used to estimate link delays from path delays without *a priori* knowledge of the routing matrix.

As the second contribution, a modif ed version of Sparse Code Shrinkage (SCS) is applied to denoise the network tomography model with errors. These errors in the measurement of various parameters for network tomography are introduced by various factors such as Simple Network Management Protocol (SNMP) operation, NetFlow measurements, etc. The estimated denoised link delays are compared with the original (error free) link delays.

The third contribution introduces a novel concept of multiple metric network tomography. By using two directly observed parameters (Packet Loss Rate (PLR) and path delay), a single parameter (link delay) is inferred indirectly, based on the evidence from the literature that there is a correlation between link delay and PLR parameters. A variation of the Non-negative Tensor Factorization (NTF), the NTF1 model, is applied for this purpose to estimate link delays.

The fourth contribution introduces a novel technique for implementing distributed network tomography. In conventional network tomography, processing is carried out by a single node in a centralized manner. A distributed system of network tomography estimation is proposed where more than one node is responsible to carry out network tomography on a distributed pattern.

List of Abbreviations and Symbols Used

A	Routing Matrix
$X_{N \times T}$	Matrix of Unknown Parameters (size $N \times T$)
$Y_{P \times T}$	Measured Parameters Matrix (size $P \times T$)
ε	Measurement Errors
ARIMA	Auto-Regressive Integrated Moving Average
BD	Blind Deconvolution
BSS	Blind Source Separation
CSLA	Cisco Service Level Agreement
EEG	Electroencephalography
EM	Expected Maximization
EO	Extended Options
ICA	Independent Component Analysis

ICMP Internet Control Message Protocol

ISP Internet Service Provider

LLS Link-local Signaling

MEG Magnetoencephalography

MLE Maximum Likelihood Estimation

MRTG Multi Router Traff c Grapher

MSE Mean Squared Error

NNMF Non Negative Matrix Factorization

NTF Non-negative Tensor Factorization

OD Origin-Destination

OSPF Open Shortest Path First

PARAFAC Parallel Factorization

PCA	Principal Component Analysis
PLR	Packet Loss Rate
QoS	Quality of service
RTT	Round Trip Time
SCS	Sparse Code Shrinkage
SLAs	Service Level Agreements
SNMP	Simple Network Management Protocol
SVD	Singular Value Decomposition
TLV	Type/Length/Value
UDP	User Datagram Protocol
VOIP	Voice Over IP

Acknowledgements

Thanks to the supervisory committee for their guidance, my family for the support, and colleagues for their help.

Chapter 1

Introduction

Computer networks form the backbone of the modern communication setup. Over the last decade, computer networks have grown exponentially in terms of the number of users, the amount of traffic, and the complexity of the applications. An important feature of modern computer networks is the absence of centralized control. This enables service providers to develop and offer a rich variety of applications and services at different quality-of-service levels.

The distributed, unregulated and heterogeneous structure of communication networks deals with various challenging tasks such as dynamic routing, service optimization, service-level verification, and detection of anomalous behavior [1]. At present, the competitive network services marketplace is aware of the critical need to detect poor quality of service.

Numerous user applications, which can be broadly classified into two categories: time-sensitive and time-insensitive applications, have been developed for modern networks. Time-sensitive applications, such as Voice Over IP (VOIP), require extremely high quality links, while time-insensitive applications, such as email, allow for the retransmission of corrupted messages. Different levels of service quality are required by different user applications. Hence performance assessment and network monitoring are critical to support the vast variety of user applications and for the service providers to meet service level agreements [2, 3, 4].

Accurately characterizing the performance of a network is essential for successful network management. Network management requires some important performance parameters such as Packet Loss Rate (PLR) and link delays to monitor, predict, and diagnose the state of a network.

Because access to measurements are restricted, and there are many users and a high volume of traffic [2, 4], the goals of assessing network performance, detection of anomalous behavior, capacity planning and efficient routing become hard to meet when networks are decentralized and multilayered.

The tools have been developed to discover the network connectivity structure, the available bandwidth of links, and other performance characteristics. Despite these efforts, quantitative network performance assessment is still very diff cult, and the expectation of full cooperation of the routing equipment is unrealistic in most situations. Collection of the required performance parameters is challenging due to a number of reasons as follows [2, 3, 4]:

1. Network measurement tasks depend on the cooperation of individual servers and routers, but router based direct measurement monitoring software such as traceroute are usually disabled to avoid computation and communication overheads.

2. Although internal monitoring is performed on select links, privacy and proprietary concerns prevent that data from being shared. Internal network information is usually considered conf dential and is not shared with outsiders. The owner of a network domain has information available about a self owned network and little or no knowledge about the properties of other domains. Thus, the decentralized nature of the Internet makes quantitative assessment of internal network performance from within the networks very diff cult.

3. Service providers cannot depend on internal network elements to freely transmit vital network statistics such as traff c rates, link delays, and packet loss rates. Routers already bear the burden of managing large amounts of incoming traff c across multiple outgoing links at very high data rates and any increased burden is inadvisable.

4. Even if the internal link-level characteristics are assumed to be collectable, collecting such statistics on various hosts may result in a considerable increase in the computational overhead, the communication cost, and required hardware. The added cost of processing and communicating performance-related statistics on demand may make network monitoring an impractical approach.

5. Traditional queuing and traff c models are not designed to capture the characteristics and complexity of network behavior.

These difficulties and challenges call for efficient data collection and analysis techniques. This is the motivation for the current research on network tomography. Network tomography research attempts to develop methods for the indirect inference of network characteristics (such as PLR, link delays, link utilizations, and routing topologies) using network measurements collected either by actively sending probe packets into the network or by passively monitoring packets in the ongoing traffic.

Network tomography presents a good means to measure the statistics of interest that may not be measured directly. Network tomography measures a parameter (that is usually not required for network management) actively or passively, and the desired parameter is indirectly measured by applying statistical techniques using an inverse modeling.

Each host does not have the visibility of the rest of the network(s) on the Internet due to the lack of fine grain measurement/analysis. For example, routers maintain no fine-grained state such as 'per user' or 'per flow' information about the traffic. Only aggregate performance parameters such as loss or utilization statistics are measured at the router interfaces. With the wide range of applications, the need of differential measurements is hard to meet with direct measurements. For example, trouble shooting PLR requires performance measurement at each link. This is unavailable while the composite performance PLR along a path of multiple links is available. The required link level performance parameter must then be estimated from the path level measurement.

Network tomography bears a strong resemblance to other inverse problems in which the key aspects of a system (such as computerized image tomography, system identification, and array processing) are not directly observable. Computerized image tomography refers to the cross-sectional imaging of an object from either transmission or reflection of data collected by illuminating the object from different directions. Tomographic imaging originated with Hounsfeld's [5] invention of the x-ray computed tomographic scanner for which he received a Nobel prize in 1972. More recently, however, medical imaging has also been successfully accomplished with radioisotopes, ultrasound, and magnetic resonance [5].

Network tomography has several practical applications:

1. Fault detection: from the inference of internal link characteristics, network administrators can identify degradations of network performance and locate such failures from path diagnosis [6].

2. Congestion control: from the inference of internal link parameters, network administrators can identify the drop rate in other f ows. This can be used to improve congestion control algorithms, or to capture violating f ows that mean to attack the network by injecting excessive traff c [2, 7].

3. Service verif cation: with the collected information about the internal parameters of a network such as link delays, loss rate and throughput, Internet Service Providers (ISP) can detect potential service violations, bandwidth theft, denial of service attacks, and then consider the need to reorganize their network or limit their users [7, 8] to reduce potential threats.

4. Network security: network tomography can also be applied in the f eld of network security. As described in [9], many types of network problems cause abnormal patterns to appear in the network traff c. Such traff c anomalies may be caused by problems ranging from security threats such as Distributed Denial of Service attacks and network worms, to unusual traff c events such as f ash crowds, to vendor implementation bugs, and to network misconf gurations. The work in [9] refers to the problem of inferring anomalies from indirect measurement as network anomography (combining anomalous with tomography, a general approach to such inference problems).

The simplest model of network tomography is represented by the following equation,

$$Y_{P\times T} = AX_{N\times T}, \tag{1.1}$$

linking the measured parameters matrix ($Y_{P\times T}$) with the matrix of unknown parameters ($X_{N\times T}$) with dependence on the routing matrix (A) of the network. If Y has P rows and X has N rows, then the size of the routing matrix (A) is $P \times N$. The rows of A (A_i) correspond to paths from the sender to the receivers and the columns (A_j) correspond to individual links in those paths. An element (A_{ij}) of the routing matrix is 1 if the link *j* is included in the path *i* and 0 otherwise.

Tomography research literature makes two common assumptions. To explain these assumptions, let us consider X_l, $l = 1, \cdot, \cdot, \cdot, \cdot, N$ to be the packet delays, and Y_i, $i = 1, \cdot, \cdot, \cdot, \cdot, P$ to be the end-to-end delay of a packet along the path destined to receiver i. The following independence and stationarity assumptions are made [2]:

- Spatial independence: packet delays at different links are statistically independent, i.e., X_i and X_j are independent for $i \neq j$.

- Temporal independence and stationarity: for a given link, the delays encountered by different packets at that link are statistically independent.

After having discussion about the importance of network parameters for effective measurement, the diff culties in collecting the desired parameters, the importance of network tomography, and the mathematical model of network tomography, some unresolved key issues related to network tomography are mentioned in the next section. These issues are the basis of the motivation for this dissertation.

1.1 Issues with Conventional Network Tomography

The f eld of network tomography has always focussed on applying statistical techniques for solving the inverse model of network tomography. Some important aspects of network tomography appear to have received little or no attention and have been left open for further investigation. This dissertation has focussed on such research issues and investigates the assumption of a known routing matrix, the presence of errors in the measurement of parameters for network tomography, multimetric network tomography, and distributed network tomography. These important and unaddressed issues related to network tomography motivated the research in this dissertation. The research motivation is described in the following enumerations:

1. **Routing Matrix Assumption:** Although network tomography offers an effective alternative solution to the problem of f nding network performance parameters, network tomography research assumes that the routing matrix is known. For example, Verdi [7] assumes a routing matrix as f xed (all entries in the matrix either 1 or 0) and random (Markovian) routing. This assumption of a known routing matrix in network tomography modeling was the motivation to research for more appropriate methods for the inverse problem solution where this assumption about the routing matrix could be eliminated.

2. **Errors in Measurements:** In practice, networks have potential errors that should be

ref ected in the network tomographic model as shown in the equation (1.2),

$$Y = AX + \varepsilon, \tag{1.2}$$

where ε represents the error in the model. There are various sources that contribute towards the error term (ε) such as SNMP operation and NetFlow measurements. The heterogeneity of the network components in terms of vendors and hardware/software platforms, that are used by various types of networking technologies, is also a contributing factor toward the error term, ε. The parameters represented by Y are used for the estimation of parameter X; obviously the values of the parameter X will not be reliable if the factor of errors (ε) is ignored as in the conventional model ($Y = AX$). The decision made for network management based on such parameters may adversely affect the performance of a network.

3. **Multimetric Matrices:** In contrast to the conventional tomography model as discussed above, the direct measurements of multiple metrics to estimate indirectly a single parameter with the expectation of getting a better estimate as compared to using a single directly measured parameter to estimate a parameter indirectly has never been looked at in network tomography research. This motivated the need to model network tomography with multi metrics like the model represented by the equation (1.3), where Y_1 and Y_2 are directly observed in order to estimate X indirectly by solving the following inverse equation.

$$Y_1 Y_2 = AX \tag{1.3}$$

For example, instead of recovering link delays only from end to end path delays, link delays can be estimated from a combination of path delays (Y_1) and PLR (Y_2). The idea behind this innovation is that a better input in terms of two interdependent metrics should produce better estimation than using only one parameter such as path level link delays. This correlation of two network parameters has been discussed in the literature. For example, the authors of [10] report on the correlation between delay and loss observed by a continuous-media traff c source. This study [10] determines the extent to which one performance measure could be used as a predictor of the future behavior of the other (for example, whether observed increasing delay is a good predictor of future loss) so that an adaptive continuous media application might take anticipatory action based on observed performance.

4. **Distributed Network Tomography**: Network tomography, in general, works in a central manner. There is a single data repository that is the node where network tomography is being performed. A single data repository brings in challenges such as susceptibility to redundancy, and computation and communication complexities. These problems can be taken care of by multiple data repositories in the form of a distributed system. As an alternative, various entities (nodes performing tomography) in a distributed system can operate concurrently and possibly autonomously. Tomographic tasks can be carried out independently and actions are co-ordinated at well-def ned stages by exchanging messages. Also, the nodes performing tomography can be heterogeneous, and failures are independent. There is no single node that performs network tomography and has the knowledge of the entire state of the system, but the workload is divided and the ref ned information is exchanged among the participating nodes that perform network tomography. This has motivated the need to research on a novel technique to implement distributed network tomography.

1.2 Contributions

The research in this dissertation consists of various unaddressed research aspects of network tomography. These issues were identif ed as open research problems in the previous section. The main philosophy behind the contributions of this dissertation has grown out of blind network tomography techniques, where the indirect estimation of a parameter (link delay) was achieved with the knowledge of only one measured parameter (path delays). In this dissertation, the estimation of the routing matrix in the inverse model of network tomography has been taken care of by the statistical ability of the blind technique applied to solve the research problems identif ed in the previous section.

This dissertation has used various blind techniques in the same way that many other f elds of science have used similar approaches. Blind deconvolution (BD) appears in various applications related to acoustics, optics, geophysics, communications, and control. In communications, the term blind channel equalization is more common, as the main interest lies in retrieving the data transmitted over a dispersive communication channel [11, 12]. In control, BD is usually known as blind identif cation, since the main goal is to obtain a model of the system [13, 14], whereas in acoustics, optics and geophysics the term blind

deconvolution is more adequate, since the goal is to undo the inf uence of a system by f nding its stable inverse.

The dissertation has considered active tomography for implementation of the blind network tomography techniques. However, the techniques developed in this dissertation can be applied to all types of network tomography. This is an additional advantage of the contributions of this dissertation as compared to the other tomography techniques where statistical techniques are typically limited to only one type of network tomography. The contributions of this thesis are summarized below.

1. The f rst contribution, published as [15], presents the application of a sparse matrix-based blind technique to eliminate the assumption of a known routing matrix in network tomography. Most of the network tomography research unrealistically assumes that the routing matrix is known and models network tomography as an inverse problem. The elimination of this assumption motivated the need to discover more appropriate methods for the inverse problem solution where the routing matrix is accommodated by the statistical ability of such methods as Non Negative Matrix Factorization (NNMF). NNMF (with the feature of sparsity) is used to factorize a matrix into two factors (matrices). The simulation results verify that NNMF performs network tomography accurately without *a priori* knowledge of the routing matrix.

2. In the second contribution, published as [16, 17], a modif ed version of sparse code shrinkage (SCS), a blind technique, is applied to denoise network tomography model with errors. SCS is used in the f eld of image recognition for denoising of the image data and here, for the f rst time, has been utilized for estimating error free link delays from erroneous link delay data. To make SCS properly adoptable in network tomography, some changes have been made in the SCS technique such as the use of NNMF instead of Independent Component Analysis (ICA) for the purpose of estimating sparsifying transformations. The estimated error free link delays are compared with the original (error free) link delays based on the data obtained from a laboratory test bed. The simulation results reveal that denoising of the tomography data has been carried out successfully.

3. The third contribution, published as [18], introduces a novel concept of multiple metric network tomography. Conventional network tomography is mono metric based

meaning that it estimates a single parameter from the observation of a single parameter. Observation of two parameters directly has been considered and both of these parameters have been used to infer a single parameter indirectly. Nonnegative Tensor Factorization (NTF) is a blind technique where a parameter in the form of a matrix is estimated from more than one directly measured parameter in the form of matrices. A variation of Nonnegative Tensor Factorization NTF (NTF1 model) has been applied for this purpose to estimate link delays from path delays and PLR. Simulation results show a better correlation between the estimated and measured link delays when a duplex of metrics is used rather than using only the path level link delays for estimating the link delays on a test bed.

4. The fourth contribution introduces a novel technique for implementing distributed network tomography. In conventional network tomography research, the processing is carried out by a single node in a centralized manner. A distributed system of network tomography estimation is proposed, where more than one node is responsible to carry out network tomography by using a blind network tomography technique such as NNMF in a distributed pattern. Implementation of a modif ed version of Open Shortest Path First (OSPF) updates with Link-Local Signaling (LLS) in NS-2 simulation is an integrated part of the fourth contribution. OSPF updates with LLS is responsible for the mutual exchange of the network tomography estimations among the special nodes called '*tomographing-nodes*' in the distributed network tomography. The estimated parameters (for example link delays) in various sections of a network are exchanged among the nodes that carry out network tomography. The simulation results verify that estimated link delays from various *tomographing-nodes* are mutually exchanged and f nally are appended at each *tomographing-node* to give link delays for the whole test bed.

The four contributions can also be combined in the form of a comprehensive commercial application that is capable of implementing the various blind network tomography initiatives discussed above. Such an application can provide a turn-key solution to network administrators for implementing various kinds of network tomography with the additional feature of multi-metric and distributed blind network tomography. In addition, the errors in the network tomography estimations can also be removed. This way the system administrators manage their networks to better solve the issues related to fault detection, congestion

control, service verification, capacity planning, traffic engineering, and network infrastructure development. This will reflect as a better market share for a company by providing better services and products to have an increased number of satisfied customers.

1.3 Thesis Outline

The following six chapters cover the background work and the details of the contributions listed in the previous section. The thesis is organized in the following structure.

Chapter 2 provides an overview of the various ways of categorizing network tomography. In the literature review, the most popular categories of network tomography have been described as passive network tomography, active network tomography, and topology identification. Major contributions and respective mathematical techniques are reviewed in each of the network tomography categories.

Chapter 3 presents the application of a sparse matrix-based blind technique to eliminate the unrealistic assumption of a known routing matrix in network tomography. Routing matrix challenges have been discussed. NNMF has been explained as a matrix-based blind method to solve inverse problems. The relationship between the network tomography and NNMF has been established. For validation of the application of NNMF for network tomography, laboratory test beds were used. A description of the test bed, the types of traffic, and the detail of ping traffic is given. The steps required in the processing of the data obtained from the test bed are explained. Finally, the results have been interpreted to show that the estimated link delays are strongly correlated to the measured link delays.

Chapter 4 highlights the issue of errors in the estimation from network tomography and provides the solution through a modified version of SCS to denoise the network tomography model with errors by using NNMF in place of ICA. The sources that cause the introduction of the errors in the network measurements, to be used as input to NNMF, have been examined. The related work in the field of error modeling has been reviewed to show the significance of this contribution. SCS has been explained and the rationale for using SCS has been described. A description of the test bed used, the types of traffic, and the details of ping traffic is given. The steps required in the processing of the data obtained from the test bed are explained. Finally, the results have been shown as graphs giving a comparison of the measured, errored, and denoised link delays based on the data obtained from a laboratory test bed. The simulation results reveal that denoising of the tomography data has

been carried out successfully by applying SCS.

Chapter 5 introduces a novel concept of multiple metric network tomography. By using two directly observed parameters (PLR and path delay); a single parameter (link delay) is inferred indirectly. In related work, it has been described that the use of multi metrics is a different concept from the concept of additive metrics and the novelty of the multi metric has been highlighted. The correlation of the link delays and PLR has also been reviewed. The simulation arrangements in the form of a test bed that includes multimedia and ping traff c has been described. Two sets of results have been presented that are the estimation of link delays from path delays and the estimation of link delays from a combination of path delays and PLR. Simulation results show a better correlation between the estimated and measured link delays when a duplex of metrics is used, as compared to using only the path level link delays for estimating the link delays on a test bed.

Chapter 6 proposes a novel technique for implementing distributed network tomography. In conventional tomography research, the network tomography processing is carried out by a single node in a centralized manner. A distributed system of network tomography estimation is proposed where multiple nodes are responsible to carry out network tomography on a distributed pattern. The related work reviews the only research effort on the topic of distributed tomography and gives a critique. The steps involved in the distributed network tomography are described. The estimated parameters (for example link delays) are exchanged among multiple data repositories by implementing LLS capable version of OSPF in NS-2 network simulator. This method uses the benef ts of distributed computing and avoids the disadvantages of centralized computing. The simulation results verify that the link delays from various sections of a test bed were estimated by the respective *tomographing-nodes*. These estimations were exchanged among the *tomographing-nodes*, and f nally these sectional estimates were appended at each *tomographing-node* to provide the link delays of the entire test bed.

Chapter 7 summarizes the main topics and contributions presented in this dissertation and outlines the future research directions that are heralded by this dissertation.

Chapter 2

Network Tomography

2.1 Introduction

This chapter explains various aspects of network tomography in detail including classif cation methods, the role of parameters in the model, and the contribution of previous research work in network tomography.

After having discussed the needs, the motivation, and the basic model of network tomography in the previous chapter, this chapter is organized to provide a detailed in-depth analysis of the various types of network tomography so that the readers of this dissertation are enabled to judge the simplicity, superiority, and affectiveness of the contributions of this dissertation over the work done previously in the f eld of network tomography.

As per the network tomography model discussed in Chapter 1, the parameters represented by X and Y depend on the type of network tomography, such as passive, active, and topology identif cation. The following is a brief description of these types of network tomography.

End-to-end measurement techniques can be classif ed according to the manner in which measurements are acquired. Passive techniques observe already-existing network traff c, and thus consume no extra bandwidth. They look for naturally occurring packet sequences which provide information about the state of the network. The main drawback of using passive techniques is the lack of measurement f exibility, since the patterns of interest may only occur rarely. On the other hand, active techniques make measurements by sending packet probes between the participating hosts. These techniques center on designing probing algorithms to gauge a specif c network property. Although most probing algorithms aim to be economical, active measurements still expend network resources. Thus, the tradeoff between using active and passive methods is a matter of f exibility versus resource consumption.

Knowledge of the routing matrix (A) is crucial for most of the network tomography problems; such knowledge, however, is not always readily available, and this need shapes the

third type of network tomography, topology identif cation [2, 4]. A summarized comparison of the types of network tomography is given in Table 2.1. The key idea in most of the existing topology identif cation methods is based on the collection of measurements at pairs of receivers. A simple example is the case of delay covariance. These types of network tomography (passive, active, and topology identif cation) are compared in terms of traff c type, measured and inferred parameters, and advantages and disadvantages in the following sections.

Table 2.1: Comparison of Network Tomography Types

Tomography Type	Traff c Type	Parameters Measured	Parameter Infered	Pros.	Cons.
Active	Unicast Multicast Probing	End-to-end loss rate Delay	Link Delay, Packet loss Bandwidth	Traverses through others networks	Multicast disabled, unicast causes extra traff c synchronized clocks assumed
Passive	Routine Traff c	Delay Packet count	Traff c Matrix	Good for private networks	Cooperation of external routers
Topology Identif cation	Multicast Probing	Relative loss rate Delay	Logical Topology	Beyond owned network	External cooperation

2.2 Literature Review

Network tomography has been looked at in the literature from various angles; as active (link-level), passive (path level), or topology identif cation (as mentioned before), by the type of probing (unicast and multicast), and by the type of statistical technique used (Maximum Likelihood Estimation (MLE), Auto-Regressive Integrated Moving Average (ARIMA), Markov Chain, Expected Maximization (EM), Clustering, and Bayesian techniques).

In general, network tomography has two forms:

1. Path-level [19, 20]: measurements are taken at each internal node, and inference

is made for origin-destination (OD) characteristics, for example, obtaining a traffic matrix. For a traffic matrix, the number of packets that pass through each internal node is counted. Then an estimation is made for the amount of traffic that originates from a specified source node and is destined for a specified receiver node. Traffic intensities of all these OD pairs are combined to form the traffic matrix.

2. Link-level [4, 21, 22]: Measurements are taken at a source node and inference is made for internal link characteristics, such as PLR, delay distribution and logical topology. Logical topology identification is also mentioned as the third type of network tomography, but from a technical point of view it is a special case of link level tomography. For logical topology, the degree of correlation between receiver nodes is computed, for example, the correlation of counts of losses, and delay covariance.

Network tomography is carried out in a number of ways that are classified by several factors:

1. Active or passive [2, 23]: as mentioned before, active tomography sends probes across a network and keeps track of end-to-end information for those probes. This information is then used to estimate internal link-level parameters. Passive tomography captures packets of normal network traffic as they are sent across links and collects information about them. This information is usually collected at the link-level and used to estimate path-level characteristics. Despite the fact that the probes sent by active monitoring may disturb the normal network traffic, they are more controllable during the measurement process than passive network topology and can give more reliable information about link behavior than passive monitoring.

2. Multicast or Unicast probes [3, 21]: active monitoring has two probing schemes. One is the multicast scheme, in which each probe is replicated at internal nodes for every branching path. The packet is effectively sent from the root node to all the receiver nodes. The key to multicast probing for network tomography is that it introduces dependency between end-to-end measurements on different receivers, which in turn enables inferences about network internal link characteristics. The other scheme is unicast probing, which refers to sending separate packets from the source node, one at a time, to the receiver nodes. Unicast traffic is easy to generate and is widely supported by computer networks. But this scheme suffers from identifiability problems;

therefore modif cations such as back-to-back unicast have been proposed in the literature [3, 21]. The key idea behind back-to-back unicast probing is to mimic the behavior of the multicast probing scheme, and the closer the resemblance, the more accurate the probing result.

3. Fixed or random routing [2, 7]: in f xed routing networks, the directed path between any two nodes is known and remains the same during the measurement time. In random routing networks, the directed path between any two nodes is determined by a known Markov Chain and changes with time.

4. Single or multiple source [8]: most of the existing techniques for network tomography are based on single source topologies. However, multiple sources are also used in the context of network tomography. It can measure some of the internal links that are inaccessible by single source measurement. Moreover, it is possible to get a more accurate and ref ned network characterization with the information gleaned from multiple sources.

However, the most common categorization of network tomography is passive, active, and topology identif cation. Further explanation of these types and the research work related to each type of network tomography is given in the following subsections.

2.2.1 Passive Network Tomography

The goal of the traff c matrix estimation literature is to estimate the intensity of point-to-point, or Origin-Destination (OD) traff c between nodes in a network based upon total traff c counts observed at each link. This approach is called passive network tomography as the measurements consist of observations made on traff c already in the network.
To understand the terms involved in passive (also called path level) tomography, here is an explanation. Suppose one wants to transfer a f le from a remote network location to the local host. The contents of a f le are broken into pieces, called packets, that also contain OD information, reassembly instructions (such as sequence numbers), and error-correction features. The collection of packets comprising the entire f le is called a f ow. The OD information is used by the network elements (routers and switches) to deliver the packets to the intended recipient. One can think of the routers as the intersections in a road network. Packets belonging to different f ows are queued at routers, awaiting their transmission to

the next router according to some protocol (f rst-in-f rst-out is common, but there are other methods as well). Physically, a queue consists of a block of memory that temporarily stores the packets. If the queue (memory) is full when a packet arrives, it is discarded and depending on the transmission protocol, the sender may or may not be alerted. Otherwise, the packet waits until it reaches the front of the queue and is forwarded to the next router on the way to its destination. This queuing mechanism is responsible for observed packet losses and, to a large extent, for packet delays.

Figure 2.1 shows such an example and is taken from [7]. There are 4 nodes in the network. The links are either unidirectional or bidirectional. There are 7 directed links (a to b, c to a, a to c, c to b, b to c, c to d, d to c), hence 7 measurements, and 12 possible OD pairs (ab, ac, ad, ba, bc, bd, ca, cb, cd, da, db, dc). In the following discussion, the directed links, OD pairs, and the routing table that connects the OD pairs are shown as matrices in the network tomography model settings. Suppose the network traff c is stationary and the routing is f xed (if multiple paths are possible between a source-destination pair, only one

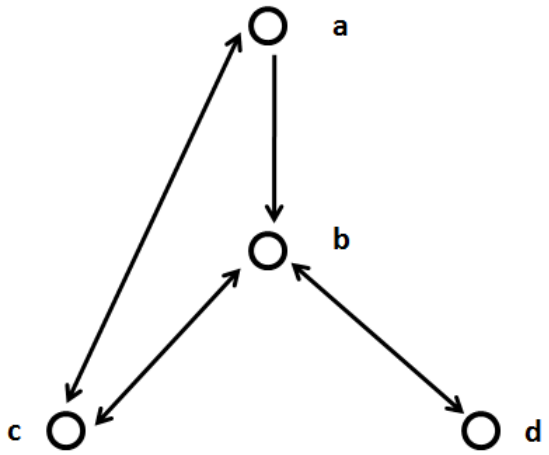

Figure 2.1: A simple topology of passive network tomography

of them is used and it is known which one). The problem then becomes a linear inverse problem, $Y = AX$, where $Y = [Y1, ..., Yp]^T$ is the vector of the aggregated traff c counts (the measurements), $X = [X1, ..., Xn]^T$ is the vector of the source-destination counts and A is the routing matrix, which is determined by the routing tables held at every node. Every element of A is either 1 (the link is on the path of a certain OD pair) or 0 (the link is not on the path). All the three components in the network tomography model, $Y = AX$, are

shown below as three matrices.

$$\begin{pmatrix} ab \\ ac \\ ca \\ bc \\ cb \\ bd \\ db \end{pmatrix} = \begin{pmatrix} 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} ab \\ ac \\ ad \\ ba \\ bc \\ bd \\ ca \\ cb \\ cd \\ da \\ db \\ dc \end{pmatrix}$$

The inverse problem arises from the fact that Y (the left-most matrix above) is measured, but the traff c matrix X or its distribution (the right most matrix above) is of considerable interest to service providers and network engineers. This is a highly ill-posed inverse problem as the number of observations of nodes (Y) is much smaller than the number of OD pairs (X). So additional assumptions are required or some type of regularization must be used before all the parameters are estimated. A number of approaches have been proposed to solve this statistically ill-posed inverse problem. Some of the prominent contributions are brief y discussed in the following paragraphs.

Vardi [7] assumed that the OD pair counts (X) follow Poisson distributions with different parameters. Since the mean and variances are the same for the Poisson distribution, this yields additional information (in the form of variances) to estimate the parameters.

Vardi's paper [7] developed the maximum likelihood approach using the EM algorithm but points out certain diff culties:

1. The problem becomes computationally intractable for large networks.

2. It is possible to construct simple networks for which the Expectation Maximization (EM) algorithm will not converge to the Maximum Likelihood Maximization (MLE).

The EM-algorithm for obtaining the MLEs is computationally very intensive, so moment-based methods were also proposed in [21].

In [21], a slightly different framework is introduced for the passive or path level network tomography problem. They model the OD packet counts as a normal random variable in which the variance is proportional to the mean raised to a power. The authors specify an EM algorithm for computing the MLE. The model is extended in the situation in which the traffic intensities vary over time. This estimation is accomplished by using a moving window. Thus, all the observations within a certain time period are used to estimate the parameters. The window of time is shifted one unit and the estimation is repeated independently of the previous windows (despite sharing most of their observations). The result is a smoothly changing set of estimates over time.

A Bayesian approach for the same framework was considered in [24]. The goal here is slightly different as the authors of [24] seek to estimate the actual OD traffic counts instead of the distribution of the counts. Again, they assume that these counts follow a Poisson process which is estimated as a means to get the counts. The authors of [24] present a Markov Chain Monte Carlo (MCMC) algorithm for estimation.

Zhang et al. [25] present an information-theoretic approach to the traffic matrix problem. The main idea is to estimate a model that is consistent with the data while remaining as close as possible to an independent model. The closeness is measured in terms of entropy. In this case, the purely independent model is based on a gravity model in which traffic between the source and the destination is a scaled product of the total traffic originating at the source and the total traffic ending at the destination.

All the types of passive network tomography techniques, reviewed above, have a common assumption that the routing matrix is known. For example, the routing matrix is assumed as fixed (1s or 0s) and random (Markovian) routing. In practical networks, the routing matrix is unknown. One important contribution of this disseration is that this assumption of a known matrix in the network tomography model is eliminated by the use of the blind technique of NNMF where routing matrix estimation is taken care of by the statistical ability of NNMF.

There are various sources such as SNMP operation and NetFlow measurements that introduce errors in network tomography estimates. The decision made for network management based on such parameters may adversely affect the performance of a network. None of the passive network tomography techniques considers these errors while modeling network tomography. This dissertation has considered this factor of errors and has introduced a

technique to denoise the tomography data.

The passive tomography models use a single directly measured parameter to estimate a parameter indirectly. This dissertation presents an approach for improved indirect estimation of a single parameter by directly measuring multiple metrics. The idea behind this innovation is that a better input in terms of more than one interdependent metric should produce better estimation than using only one parameter such as path level link delay.

All the passive network tomography techniques, in general, work in a central manner. There is a single data repository that is the node where network tomography is being performed. A single data repository brings in challenges such as susceptibility to redundancy, and computation and communication complexities. These problems can be taken care of by multiple data repositories in the form of a distributed system. As an alternative, distributed network tomography is proposed in this dissertation. Tomographic tasks can be carried out independently and actions are co-ordinated at well-def ned stages by exchanging messages. Also, the nodes performing tomography can be heterogeneous, and failures are independent.

This subsection covered the salient related work in the f eld of passive or path level network tomography. One is left with the impression that there has always been a great emphasis on f nding better statistical alternatives for the solution of the inverse problem of network tomography including EM algorithms, Bayseian approach, information theoretic approach, and others. The novel contributions of this dissertation in addressing many issues not addressed by the previous techniques have also been mentioned in the above paragraph.

The operational limitations of passive tomography, such as full access to a network, makes active network tomography an alternative. Active network tomography is discussed in the next subsection.

2.2.2 Active Network Tomography

The application of passive network tomography requires full control of the network. With no control over other networks, the link level information cannot be obtained and thus active tomography is required as an alternative method for performing network tomography. This section brief y analyzes some signif cant contributions in the area of active network tomography.

The goal of active network tomography is to estimate the performance characteristics of

individual links based upon measurements of injected traffic sent across a network from one accessible edge node to another node (or to a group of nodes) as described in [26]. Link-level network tomography is the estimation of link-level network parameters (PLR, delay distributions) from path-level measurements. Link-level parameters can be estimated from direct measurements when all nodes in a network are cooperative. Many promising tools such as pathchar (pchar), traceroute, clink, and pipechar use Internet Control Message Protocol (ICMP) packets (control packets that request information from routers) to estimate link-level loss, latencies, and bandwidths. However, many routers do not respond to or generate ICMP packets or treat them with very low priority, motivating the development of large-scale link-level network inference methods that do not rely on cooperation (or minimize cooperation requirements). The challenge for active network tomography is to deconvolve this path-level information into link-level information. Although it can be viewed as the reverse of the situation encountered in passive tomography, the active tomography also gives rise to an inverse problem of the form $Y = AX$, where A is the routing matrix giving the end-to-end paths, X contains the unknown link processes, and Y gives the observed path-level data. Here also the routing matrix has a row for each path and a column for each link with $A(i,j) = 1$ when link j is in the path i.

The probing activity has three variations in terms of the receiving nodes; a probe can be sent from one source to a single destination, a multicast probe can be sent from a source to multiple sources, and wherever multicast is not permitted, a variation of the unicast that is back-to-back unicast can be used to mimic mutlicast. Back-to-back unicast probes are sent to more than one node with the inter-probes distance so small that this repeated unicast appears to be a multicast. The following paragraphs give a description of the research contributions related to these variations of probing.

Multicast Probing

Let us consider the network scenario shown in Figure 2.2 to get a better understanding of the loss rate network tomography through multicast probing as discussed in [3]. If a multicast packet is sent by the sender (node 0) and received by node 2 but not by node 3, then it can be immediately determined that loss occurred on link 3 (successful reception at node 2 implies that the multicast packet reached the internal node 1). By performing

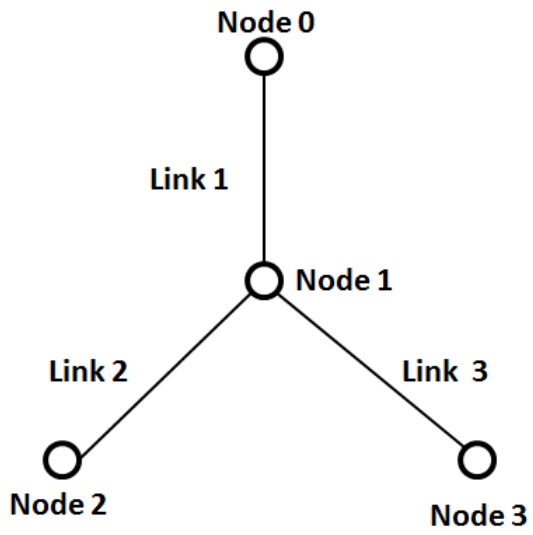

Figure 2.2: A simple topology of four nodes

such measurements repeatedly, the rate of loss on the two links 2 and 3 can be estimated; these estimates and the measurements enable the computation of an estimate for the loss rate on link 1. To illustrate further, let θ_1 , θ_2 , and θ_3 denote the log success probabilities of the three links in the network, where the subscript denotes the lower node attached to the link. Let $p_{2|3}$ denote the ratio of the number of multicast packet probes simultaneously received at both nodes 2 and 3, relative to the total number received at node 3. Thus, $p_{2|3}$ is the empirical probability of success on link 2 conditional upon success on link 3, which provides a simple estimate of θ_2 . Def ne $p_{3|2}$ in a similar fashion, and also let p_i , i=2,3 denote the ratio of the total number of packets received at node i over the total number of multicast probes sent to node i. We can then write,

$$\begin{pmatrix} \log(\widehat{p}_2) \\ \log(\widehat{p}_3) \\ \log(\widehat{p}_{2|3}) \\ \log(\widehat{p}_{3|2}) \end{pmatrix} = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \end{pmatrix}$$

A least squares estimate of θ_i is easily computed for this overdetermined system of equations.

The authors of [26] are concerned with estimating link loss rates based upon multicast probing. [26] describes a framework common in this area: treeshaped topologies, spatially

independent links, and temporally independent probes. Consider the network in Figure 2.3. A single multicast probe sent from node 0 will try to reach all of the receiver nodes: 4, 5, 6, and 7. It does so in the following manner. A packet is placed on the link from node 0 to node 1. At node 1, the packet is duplicated and sent on to the children of node 1. At each subsequent child node, the packet is duplicated further and sent. In this case, each probe generates a fourtuple observation. At each of the receivers, the packet is either received or lost. The authors assume that link losses are independent Bernoulli processes and derive an estimator based on the end-to-end measurements. The shared information resulting from the common paths and the multicast mechanism produces correlated end-to-end observations that can be used to deconvolve the path-level loss process into the link-level loss processes. This estimator involves solving a polynomial equation and the solution asymptotically corresponds to the maximum likelihood estimate.

In [27], it is postulated that a pseudo-likelihood method is appropriate for both link performance and traff c matrix estimation with multicast experiments. The basic focus is to consider pairwise projections of the high-dimensional observation. The result is an algorithm with improved computational eff ciency at the expense of some statistical eff ciency. Some important properties are preserved such as consistency. The authors of [28] also tried to use similar ideas to estimate link delay distributions, but the method does not extend easily to the delay problem.

Unicast Probing

Alternatively, one can tackle loss rate and delay distribution tomography using unicast measurements. Unicast measurements are more diff cult to work with than multicast, but since many networks do not support multicast, unicast-based tomography is of considerable practical interest. The diff culty of unicast-based tomography is that although the single unicast packet measurements allow the estimation of end-to-end path loss rates and delay distributions, there is no unique mapping of these path-level parameters to the corresponding individual link-by-link parameters. For example, referring again to Figure 2.2, if packets are sent from node 0 to nodes 2 and 3 and n_k and m_k denote the numbers of packets sent

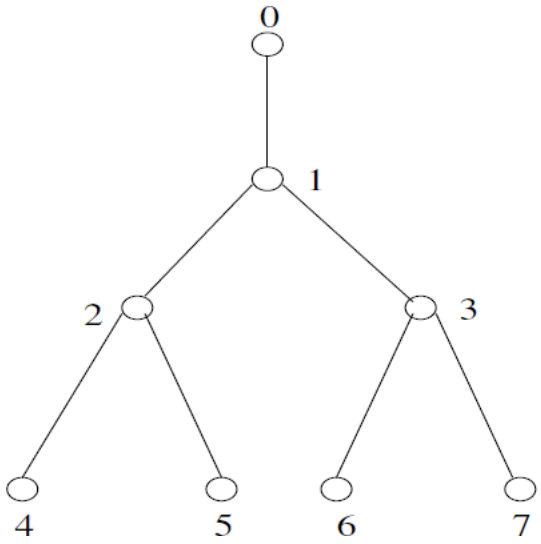

Figure 2.3: A tree topology for multicasting

to and received by receiver node k, k=2, 3, then,

$$\begin{pmatrix} \log(\widehat{p}_2) \\ \log(\widehat{p}_3) \end{pmatrix} = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix} \begin{pmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \end{pmatrix}$$

where $\widehat{p}_k = m_k/n_k$ and θ_j, j =1, 2, 3 denotes the log success probability associated with each link. Clearly, a unique solution for θ_i does not exist since A is not full rank. It suffers from identif ability problems: active probing experiments based on unicast schemes alone cannot identify all the internal link parameters of interest, such as loss rates and delay distributions. To address this problem, the authors of [21] introduced back-to-back unicast experiments for link loss estimation. Practically, this allows analysis of networks in which the multicast mechanism is not available. They developed an EM algorithm to maximize the likelihood and studied some of the properties of the resulting estimator.

The authors of [29] estimated link delay distributions based on back-to-back unicast measurements. They altered the traditional discrete model slightly and introduced an eff cient EM algorithm based on the Fast Fourier Transform (FFT).

The back-to-back unicast probing scheme proposed in [29] also imitates the multicast protocol. Figure 2.4 explains the back-to-back unicast probing scheme. At time t a unicast packet is sent to receiver i and at time t+δt a second unicast packet is sent to receiver j. If

the time difference δt is small enough, we would expect the two unicast probes to experience a similar network environment on the common path that they share on the network. The authors of [29] did not quantify the correlation on the common path; instead it was

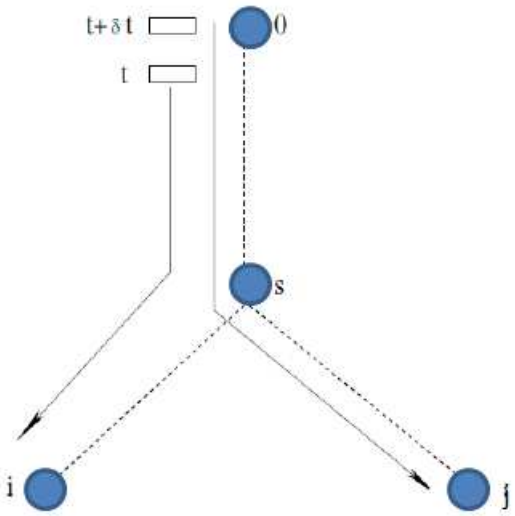

Figure 2.4: A back to back probing case

assumed that the loss experience and delays on the common path are identical. Under such an assumption the back-to- back unicast probing scheme in [29] is equivalent to the bicast (multicast to a pair of receivers) protocol. However, the assumption of identical performance will not hold in all cases and it is diff cult to verify this assumption.

The authors of [30] also modeled delay using back-to-back unicast probing. They modeled each link delay distribution as a f nite mixture of Gaussians with a point mass at zero. They used a penalized maximum likelihood approach to choose the number of mixing components and estimate the parameters.

The authors of [27] present a pseudo-likelihood method appropriate for both link performance and traff c matrix estimation. The basic focus is to consider pairwise projections of the high-dimensional observation. The result is an algorithm with improved computational eff ciency at the expense of some statistical eff ciency.

This dissertation has used active tomography, but is applicable to all other types of network tomography.

Like passive network tomography, there are various sources that introduce errors in tomography data into active network tomography such as SNMP operation and NetFlow

measurements. The errors in measurements for active network tomography have been addressed and a novel method to denoise network tomography data has been introduced.
All the conventional active tomography methods, as discussed in this section, use a single directly measured parameter to estimate a parameter indirectly. Based on the fact that a better input in terms of more than one interdependent metrics should produce better estimating than using only one parameter, this dissertation introduces a novel approach of estimation of a single network parameter from the input of multiple parameters.
This subsection has described various research contributions in the area of unicast probing and has explained the difference between unicast and back-to-back unicast probing.
The next section describes the third type of network tomography, topology identif cation.

2.3 Topology Identif cation

Most of the network tomography problems addressed in earlier sections dealt with the identif cation of network performance parameters, with full knowledge of the network (routing) topology. As mentioned before, there are deterministic tools like traceroute that report the network devices and their connectivity. These cooperative conditions are often not met in practice and may be increasingly uncommon as the network grows and privacy and proprietary concerns increase. Knowledge of A is crucial for network tomography problems; such knowledge, however, is not always readily available, and this need shapes the third type of network tomography, topology identif cation [2, 4]. Topology identif cation is concerned with discovering devices on a network.
For situations in which common tools such as traceroute are not applicable, a number of methods have been proposed for the identif cation of network (routing) topology based on end-to-end measurements by determining the degree of correlation between receivers [8, 21, 31]. Most of these approaches have concentrated on identifying the tree structured topology connecting a single sender to multiple receivers. It is assumed that the routes from the sender to the receiver are f xed. With only end-to-end measurements, it is only possible to identify the logical topology def ned by the branching points between paths to different receivers. The key idea in most of the existing topology identif cation methods is to collect measurements at pairs of receivers. A simple example is the case of delay covariance. If two receivers share some portion of their paths, then the covariance between the end-to-end delays to the two receivers is ref ective of the sum of the variances on the shared

links (assuming the delays are not correlated on unshared links). The more shared links (larger shared portion of their paths), the larger the covariance between the two. Metrics possessing this type of monotonicity property can be estimated from a number of different end-to-end measurements including counts of losses, counts of zero delay events (utilization), delay correlations, and delay differences [3, 26, 32].

Thus, using active probing experiments to determine the topology is an important area of research in network tomography. The methods used in the literature [8, 21, 31] include clustering techniques, maximum likelihood, and Bayesian approaches. There are also some clever ideas for probing experiments such as the sandwich probing scheme [21]. Figure 2.5

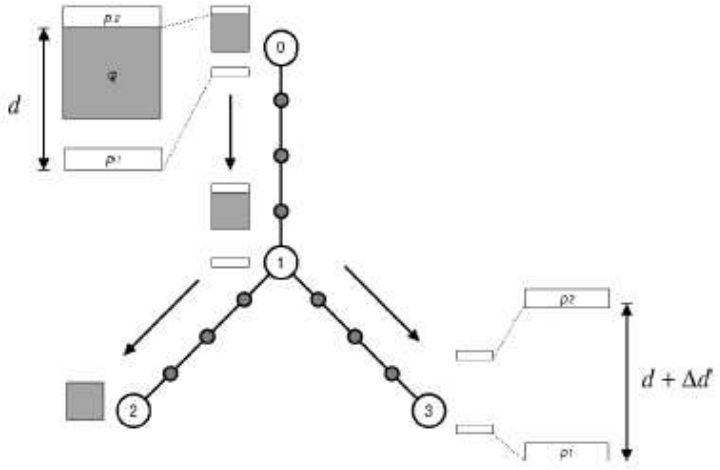

Figure 2.5: A back to back probing case

illustrates this idea. Two small unicast probe packets, packets 1 and 3, are sent to one receiver and the large unicast probe packet is sent to another receiver. But the large probe is sandwiched between the two small probes. Let us suppose that d denotes the time difference between probes 1 and 3 at the source node and d^* denote the observed time difference between the two probes at the receiver. The longer the common path in the pair of receivers, the longer is the delay. Thus, the delay differences can be used to estimate the topology. Since only the measurements of local delay differences are made, the clock synchronization among the nodes on the network is not an issue.

In single source active measurement techniques, the paths from the source to the receivers form a tree structure with the source at the root of the tree and receivers as the leaves.

Active probing techniques are designed to measure a specific network property. The multicast transport mechanism was identified early on as being well-suited for active probing [3, 26, 32]. Each multicast packet sent from the source is replicated whenever there is a new branch in the tree. Consequently, when a packet gets dropped or queued on a certain link, all receivers descended from that link will observe the effect of the loss or queueing behavior.

Ratnasamy and McCanne first demonstrated that these correlated drop observations could be used to reconstruct the multicast topology and to infer the link-level loss rates [3]. Duffield et al. [31] then rigorously established the correctness of this algorithm and developed a framework under which other metrics such as delay variance and Explicit Congestion Notification (ECN) marking rate could be used in place of loss rate for inferring the topology [27, 33, 34]. Duffield et al. proposed the use of stripes rather than packet pairs, where each packet in the stripe is sent to a different receiver. However, as the length of the stripe grows, correlation is weakened throughout the stripe. Long stripes are also much more intrusive and prone to disrupt other network traffic than packet pairs.

Motivated by the lack of support of multicast protocols over the entire Internet infrastructure, and because the majority of Internet traffic uses the unicast transport mechanism, researchers then developed a series of measurement tools using the unicast mechanism. Many of these techniques utilize packet pair measurements to infer loss [30, 31] and delay distributions [28]. In these measurements, packets are sent back-to-back and each packet is destined for a different receiver. Much like the correlation experienced by multicast packets, back-to-back packets are highly correlated on shared links before the paths to each receiver branch apart. Thus, the main difference between the techniques of using unicast and multicast probes is that the unicast measurements are made only to pairs of receivers at a time, whereas each multicast packet is transmitted to all of the receivers.

Researchers have also investigated the problem of identifying a single source unicast topology using special-purpose probes [2, 21, 35]. The probes in these algorithms use a collection of different sized packets sent to two receivers, noting that larger packets have a longer transmission time than smaller packets. They then employ a complexity reducing hierarchical statistical model to reconstruct the tree topology.

Whereas all previously mentioned techniques had only utilized a single source, the work in [36] combines measurements made independently from a collection of multicast sources.

Assuming the multiple source topology to be known, they establish conditions which guarantee that all links of interest are identif able from end-to-end measurements. Furthermore, they present two algorithms for estimating loss from the combined set of measurements. However, a closer look at the problem of identifying a multiple source topology from the end-to-end measurements reveals that this is no trivial task.

This dissertation focused on active tomography for estimating link delays from path level delays. The reason for reviewing topology identif cation is that topology identif cation, in general, applies active network tomography to guess the topology of a network. Therefore, this dissertation has a strong resemblance with the topology identif cation network tomography.

The next section summarizes the discussion on the existing def ciencies and research challenges related to network tomography in the context of the contributions of this dissertation.

2.4 Discussion on Existing Def ciencies and Research Challenges

The discussion in this chapter explains three types of network tomography; active (link-level), passive (path level), and topology identif cation. Explanation of the types of probing and the types of statistical technique for network tomography has also been given in this chapter. All the types of tomography have a common assumption that the routing matrix is known. For example, the routing matrix is assumed as f xed (1s or 0s) and random (Markovian) routing. In practical networks, the routing matrix is unknown. This assumption of a known matrix in the network tomography model has been addressed in this dissertation in Chapter 3. Chapter 3 applies a blind technique (NNMF) and eliminates the assumption of a known routing matrix.

Some of the conventional tomographic techniques such as [22, 37] have used statistical parameters such as mean, variance, and standard deviation to compare the recovered and true link matrices. The variance and standard deviation describe how spread out the data is. If all the data lies close to the mean, then the standard deviation will be small, while if the data is spread out over a large range of values, standard deviation will be large meaning that having outliers will increase the standard deviation. Considering these factors and the prominent role of correlation in validity testing as described in [38], the correlation coeff - cient has been preferred over mean and variance in Chapter 3.

The experiments in this dissertation use active tomography for implementation. However,

the techniques developed in this dissertation are not limited to this type of network tomography and can be used with all types of network tomography. This is a major advantage of the contributions of this dissertation as compared to the techniques presented in the literature review where statistical techniques are typically limited to only one type of network tomography.

In addition to the elimination of the assumption of a known routing matrix, some other important aspects of network tomography are also in the past research on network tomography. There has not been any effort to solve the unaddressed problems such as the errors in the measurement of parameters for network tomography, multi-metric tomography, and distributed network tomography. These important and unaddressed issues related to network tomography may improve the results of network tomography signif cantly.

In reality, all the practical networks have potential errors that should be ref ected in the network tomographic model as shown in the equation below,

$$Y = AX + \varepsilon, \tag{2.1}$$

where ε represents the error in the model. There are various sources that contribute towards the error term (ε) such as SNMP operation and NetFlow measurements. The heterogeneity of the network components in terms of vendors and hardware/software platforms, that are used by various types of networking technologies, is also a contributing factor toward the error term, ε. The parameters represented by Y are used for the estimation of parameter X; obviously the values of the parameter X will not be reliable if the factor of errors (ε) is ignored as in the conventional model ($Y = AX$). The decision made for network management based on such parameters may adversely affect the performance of a network. The errors in measurement have been tackled in Chapter 4.

The conventional tomography model, as discussed in the current chapter, uses a single directly measured parameter to estimate a parameter indirectly. In contrast to this approach, Chapter 5 presents an approach for improved indirect estimation of a single parameter by directly measuring multiple metrics. The idea behind this innovation is that a better input in terms of more than one interdependent metric should produce better estimation than using only one parameter such as path level link delay.

Network tomography, in general, works in a central manner as all the techniques reviewed in this chapter use a central approach. There is a single data repository that is the node

where network tomography is being performed. A single data repository brings in the challenges such as susceptibility to redundancy, and computation and communication complexities. These problems can be taken care of by multiple data repositories in the form of a distributed system. An alternative distributed network tomography is proposed in Chapter 6 where various entities (nodes performing tomography) in a distributed system can operate concurrently and possibly autonomously. Tomographic tasks can be carried out independently and actions are co-ordinated at well-def ned stages by exchanging messages. Also, the nodes performing tomography can be heterogeneous, and failures are independent.
To sum up, this section describes past research on topology identif cation network tomography. The main mathematical techniques applied in topology identif cation are clustering, maximum likelihood, delay distribution, and packet pair measurements.

2.5 Chapter Summary

This chapter has described various ways of categorizing network tomography. In the literature review, the most popular categories of network tomography have been described as passive network tomography, active network tomography, and topology identif cation. Major contributions and respective mathematical techniques are reviewed in each of the network tomography categories. This dissertation applies active tomography to implement the proposed blind network techniques. However, the techniques developed in this dissertation are not limited to one type of network tomography and can be used with all types. This is an additional advantage made possible by the contributions of this dissertation as compared to the techniques presented in the literature review where statistical techniques are typically limited to only one type of network tomography.

Chapter 3

Network Tomography By Non Negative Matrix Factorization

3.1 Introduction

Network tomography offers an effective alternative solution to the problem of f nding network performance parameters. Network tomography research assumes that the routing matrix is known. The need to eliminate this unrealistic assumption (as mentioned in Section 2.4) motivates the research for alternative solutions such as NNMF. NNMF is a matrix based technique for solving the inverse problem and it does not require knowledge of the routing matrix to perform network tomography. Sparsity is an important feature of NNMF that allows consideration of network links (in use) from a minimum to a maximum number. For this contribution, NNMF with the feature of sparsity has been applied to the data obtained from a laboratory test bed to perform delay tomography under various traff c conditions.

The rest of the chapter is organized as follows. Section 3.2 describes the challenges faced by the routing matrix in the network tomography model. Section 3.3 discusses NNMF. Section 3.4 explains application of NNMF in the context of network tomography. Section 3.5 presents and discusses results to show that NNMF performs network tomography accurately without *a priori* knowledge of the routing matrix. Section 3.6 is the chapter summary.

3.2 Routing Matrix Challenges

A routing matrix is usually assumed to be known and constant throughout the measurement (network tomography operation) period. For example, Vardi [7] (who is credited for introducing network tomography) divided networks into two categories: f xed routing (deterministic) and random (Markovian) routing networks. In f xed routing networks, the directed path between any two nodes is f xed and remains the same for all messages traveling between these two nodes. In random routing networks, the directed path taken by a

message traveling from the source node j_1 to the destination node j_2 is determined according to a fixed known Markov chain specific to a source destination (SD) pair (j_1, j_2). From a purely technical standpoint, fixed routing is a special case of random routing.

The large-scale network inference problem deals with a potentially very large dimension of A, which can range from half a dozen rows and columns for a small local area network, to thousands or tens of thousands of rows and columns for a large network or the Internet. The periodic routing table updates occur at intervals of several minutes. The dynamics of the routing matrix may restrict the amount of data that can be collected and used for inference. Most current methodologies usually assume that performance characteristics on each link are statistically independent of all other links; however this assumption is clearly violated due to common cross-traffic flowing through the links. Assumptions of temporal stationarity are also made in many cases [2, 4].

In general, A is not full rank and causes identifiability concerns. Statistical means are employed to introduce regularization and induce identifiability. Assuming independent and identically distributed (i.i.d.) Poisson distributions for the OD traffic byte counts on a general network topology, Vardi [7] demonstrated the identifiability of the Poisson model and developed an EM algorithm to estimate Poisson parameters in both deterministic and Markov routing schemes.

The above discussion motivates the need for more appropriate methods for the inverse problem solution where the statistical ability of such methods removes the need to make an unrealistic assumption about the routing matrix. Non Negative Matrix Factorization (NNMF) is one such method. The next section briefly describes NNMF.

3.3 Non Negative Matrix Factorization (NNMF)

NNMF is one of the implementations of Blind Source Separation (BSS). The BSS approach aims at reconstructing both unobserved input signals and the mixing-weights matrix from the combined signals received at a given sequence of time points. BSS in general has applications in the fields of power systems, telecommunications, speech processing, finance/econometrics, bio-medical science and digital image processing [39]. In the case of networks, as in similar applications, the individual components can be separated.

If a non-negative matrix V is given, then the NNMF finds non-negative matrix factors W

and H such that [40]:

$$V \approx WH \tag{3.1}$$

NNMF can be applied for statistical analysis of a given set of multivariate n-dimensional data vectors; the vectors are placed in the columns of an n $\times$ m matrix (V), where m is the number of samples in the data set. After choosing a value for r (the number of factors/components to be determined) that is usually smaller than both n and m, V is approximately factorized into a productof $n \times r$ (matrix, W) and an $r \times m$ matrix, H. Matrices W and H are smaller than the original matrix V. Matrices W and H are randomly initialized by the NNMF algorithm and the parameter (r) is user def ned is always matching with the real values in this dissertation. To f nd an approximate factorization, a cost function is def ned that quantif es the quality of the approximation. Such a cost function can be constructed using some measure of distance between two non negative matrices, A and B. One popular cost function is simply the square of the Euclidean distance between A and B,

$$\|A - B\|^2 = \sum_{ij} (A_{ij} - B_{ij})^2 \tag{3.2}$$

and another is based on divergence,

$$D(A\|B) = \sum_{ij} (A_{ij} \log \frac{CA_{ij}}{B_{ij}} - A_{ij} + B_{ij}) \tag{3.3}$$

Like the Euclidean distance, the divergence is also lower bounded by zero, and vanishes if and only if $A = B$, but is not called a distance, because it is not symmetric in A and B, therefore it is referred to as the divergence of A and B. It reduces to the Kullback-Leibler divergence [40], or relative entropy, when the sum of A_{ij} and B_{ij} is equal to unity, so that A and B can have normalized probability distributions.

A cost function is used for the formulation of the NNMF optimization problem to minimize $\|V - WH\|^2$ or $D(V\|WH)$ with respect to W and H, subject to the constraints W, $H \geq 0$.

For each cost function, there are rules for updating W and H after selecting initial values of W and H. At each iteration W and H are multiplied and $\|V - WH\|^2$ or $D(V\|WH)$ is calculated. The values of W and H are updated until $\|V - WH\|^2$ or $D(V\|WH)$ reach a minimum threshold. At this moment, the values of W and H represent the f nal estimate. A useful property of NNMF is its ability to produce a sparse representation (coding) of the data. Such a representation encodes much of the data using few active components, which

makes the encoding easy to interpret [40].

There are various other ways to implement BSS such as ICA and Principal Component Analysis (PCA)/Singular Value Decomposition (SVD) [39]. These BSS approaches may produce negative values in the estimated matrices whereas the link delays and entries of a routing matrix may not be negative values. NNMF is preferred over other BSS techniques, because it only deals with positive values.

In summary, NNMF is used to factorize a matrix into two factors (matrices) with the constraint that all three matrices must be non-negative, meaning that all elements must be equal to or greater than zero. The process involves matrices and optimizing the residue to obtain the best result with the feature of sparsity. The next section highlights the connection between network tomography and NNMF models.

3.4 Network Tomography and NNMF

There are a number of reasons for the consideration of NNMF in the context of network tomography:

1. Both models, Equation 1.1 and Equation 3.1, represent a system of equations in the form of three matrices.

2. Both models, Equation 1.1 and Equation 3.1, represent an inverse problem, where the parameter (X) in network tomography and sources (H) in NNMF are calculated from directly measured Y and V, respectively.

3. The mixing matrix (W) and routing matrix (A) are generally unknown. The mixing matrix (W) is recovered through statistical processing as a part of the NNMF operation. The routing matrix (A) is assumed to be known either as f xed routing, or as probabilistic routing. In reality, the routing matrix changes over time. Determining the routing matrix starts from simpler and ineffective methods like traceroute, but is a tedious and resource intensive task. Due to these reasons, most of the researchers assume it is f xed or probabilistic.

4. Sparsity is a feature of NNMF and is also present in a network represented by the tomography model. The level of sparsity in a network may depend on the routing scheme and network topology.

By considering NNMF for performing network tomography, there is no need to make assumptions about the routing matrix. The recovery of the routing matrix is taken care of by the NNMF operation as NNMF can estimate both the unknown parameters in the model, A and X in this case. NNMF offers an alternative to conventional statistical tomography techniques. A variety of traffic sources and disturbances such as Agilent Router Tester (N2X) traffic have been applied to verify the capability of NNMF as a better alternative to conventional tomography techniques. The reasons behind the behaviorial changes in link performances are also analyzed and interpreted as part of this contribution.

3.5 Validation of the Application of NNMF for Network Tomography

To validate the idea that NNMF can perform active network tomography, two types of tests were carried out: the first is based on known numerical data and the second set of tests is based on the real measurements from test beds.
In all types of tests, the end to end (path level) delay was input to the NNMF. The recovered link level delays were correlated with the true link level delays to examine the performance of NNMF for network tomography.
The next subsection describes two variations of a test bed that were used for data collection to obtain end to end delays for use as input to NNMF and link delays for bench-marking the estimations of link delays by NNMF.

3.5.1 Description of Networking Test Beds

A test bed was set up in the Advanced Internetworking Laboratory (AIL) that consisted of eight 3800 series Cisco routers, N2X, and a Multi Router Traffic Grapher (MRTG) capable workstation. OSPF routing was implemented on routers and N2X. The test bed is of smaller size than the practical networks and has a limited number of links, because a collection of the actual values of the link delays for bench-marking the accuracy of estimated link delays is required. The test bed experiments were used to prove that NNMF estimates are close to the actual link delays. In contrast to this test bed, the practical networks are larger in scale, but scalability is not an issue as NNMF can handle larger sizes of matrices [41].
The Echopath option of the Cisco Service Level Agreement (CSLA) was implemented to send six probes and collect the cumulative Round Trip Time (RTT) from source to each

hop on the path of a probe.

The CSLA is an application-aware synthetic operation agent that monitors network performance by measuring response time, network resource availability, application performance, jitter (inter-packet delay variance), connect time, throughput, and packet loss. Performance can be measured between any Cisco device that supports this feature and any remote IP host (server), Cisco routing device, or mainframe host. Performance measurement statistics provided by this feature can be used for troubleshooting, for problem analysis, and for designing network topologies.

In particular, the CSLA is a reliable mechanism for accurately monitoring the metrics in Service Level Agreements (SLAs). Response time and availability information is collected by operations that are conf gured on a router. Operations use synthetic packets specif cally placed in a network to collect data about the network. These packets simulate other forms of network traff c, as determined by the type of operation conf gured. Operations usually consist of multiple probe packets sent into the network; operations in general can be thought of as collections of probes.

CSLA operations are given specif c identif cation numbers so one can track the various conf gured operations. CSLA operations are conf gured in RTR conf guration mode.

For this contribution, all unicast probes (page 23) were grouped together. All probes in the group started at the same time and traveled from right to left in the test bed diagrams (Figure 3.1 and Figure 3.2). The group of probes was repeated 100 times with a time difference of 10 seconds between two consecutive repetitions. Various repetitions up to two thousands were used, however there was no discernible improvement over the results with one hundred repitions. Therefore, the results from the sample size of 100 are shown in this contribution, because getting estimation in shorter time is better for computer networks. The selected links were stressed by two sources: extended ping on selected links and traff c injected from the N2X.

Figure 3.1 shows a test bed with six probes (from right to left) and four selected links that were stressed by extended ping packets of two sizes: two of these links (Link 4 and Link 5) were stressed with an extended ping of the packet size 100 Bytes and the other two links (Link 1 and Link 6) were stressed with an extended ping of 200 Bytes. The size of the probes in CSLA was 40 Bytes. The condition of the network remained unchanged during the whole period of data collection.

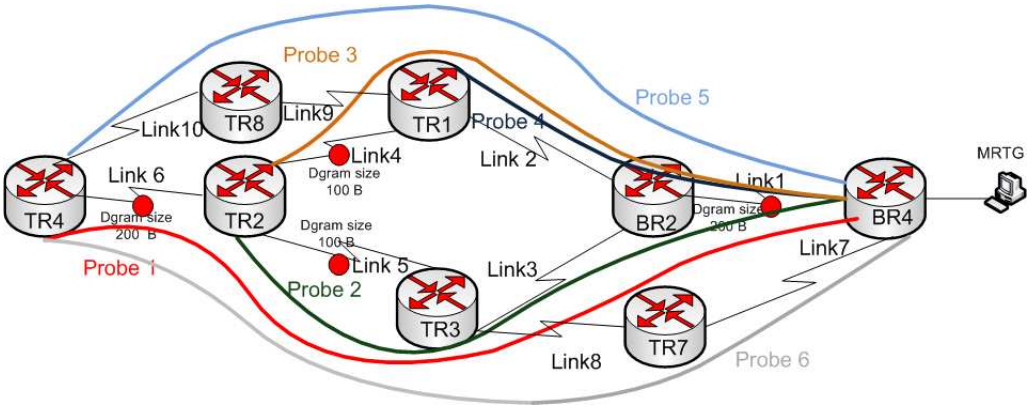

Figure 3.1: Testbed Setup with extended pings only

In Figure 3.1, the path of Probe 1 includes Link 1, Link 3, Link 5 and Link 6. The following matrix represents the routing matrix (A) for Figure 3.1 and Figure 3.2. The first row of this matrix corresponds to the path of Probe 1 and has entries of 1s for the elements of the first row that corresponds to Link 1, Link 3, Link 5 and Link 6. All other entries in the first row are 0s, because Probe 1 does not pass through these links. An element of sparsity is present as some links have values as 1s and some have values as 0s.

$$\begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \end{pmatrix}$$

Figure 3.2 shows a test bed with six probes (from right to left) and two of the links (1 and 6) were stressed with an extended ping of 200 Bytes. The other source of disturbance was the traffic from the N2X. The module 1 of N2X was generating a variable packet size from 1000 Bytes to 1500 Bytes. The size of the probes in CSLA was 10 Bytes for this scenario. In this case also, the condition of the network remains unchanged during the CSLA

operation.

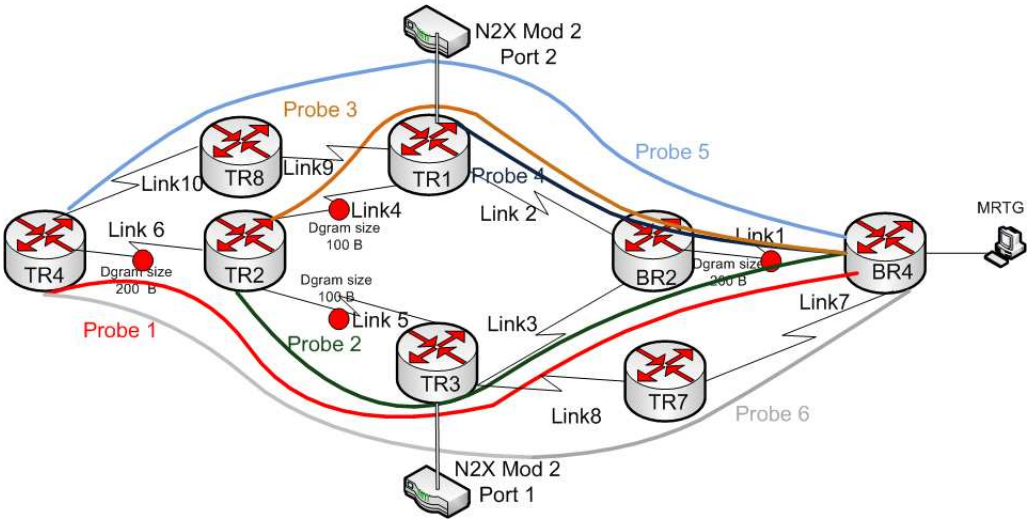

Figure 3.2: Testbed Setup with a mixture of extended pings and N2X traff c

3.5.2 Data Processing

The data obtained from the CSLA was in the form of accumulative hop-wise round trip time and a snapshot of a reading of CSLA is shown in Figure 3.3. The following steps were taken to process the data to obtain two matrices; a matrix of the end to end delays and a matrix of the link level delays:

1. As can be seen from Figure 3.3 that the output from the ‘collect statistics’ command of CSLA is in the form of text separated by tabs or blank spaces to give a tabular representation. The repetition of the ‘collect statistics’ command accumulates a list of such text patterns as in Figure 3.3 in the router command windows, which was then saved in a text f le. With path delays as an input to NNMF, the true link delays on the path of each probe were extracted. Parsing software was used to convert this information into the data required, that is the path and link delays.

   Three f elds are of special interest from the table shown in Figure 3.3; Entry, Hop, and Sumcmp. The f eld, Entry, basically identif es a probe. The f eld, Hop, gives the list of hops traversed by a probe in a round trip. The f eld, SumCmp, represents the accumulated RTT. The parsing software was written in Java and it extracts link delays

```
BR4#show ip sla monitor distribution-statistics
        Captured Statistics
Entry    = Entry number
StartT   = Start time of entry (hundredths of seconds)
Pth      = Path index
Hop      = Hop in path index
Dst      = Time distribution index
Comps    = Operations completed
OvrTh    = Operations completed over thresholds
SumCmp   = Sum of RTT (milliseconds)
SumCmp2L = Sum of RTT squared low 32 bits (milliseconds)
SumCmp2H = Sum of RTT squared high 32 bits (milliseconds)
TMax     = RTT maximum (milliseconds)
TMin     = RTT minimum (milliseconds)

Entry StartT     Pth Hop Dst Comps      OvrTh      SumCmp     SumCmp2L   SumCmp2H   TMax       TMin
1     16788174   1   1   1   2          0          3          5          0          2          1
1     16788174   1   2   1   2          0          4          8          0          2          2
1     16788174   1   3   1   2          0          7          25         0          4          3
1     16788174   1   4   1   2          0          9          31         0          5          4
2     16788372   1   1   1   3          0          3          3          0          1          1
2     16788372   1   2   1   3          0          8          22         0          3          2
2     16788372   1   3   1   3          0          9          22         0          3          2
3     16788572   1   1   1   3          0          3          3          0          1          1
3     16788572   1   2   1   3          0          8          22         0          3          2
3     16788572   1   3   1   3          0          9          22         0          3          2
4     16788772   1   1   1   3          0          5          9          0          2          1
4     16788772   1   2   1   3          0          6          12         0          2          2
5     16788174   1   1   1   2          0          4          6          0          2          1
5     16788174   1   2   1   2          0          5          10         0          2          2
5     16788174   1   3   1   2          0          8          28         0          4          3
5     16788174   1   4   1   2          0          9          40         0          5          4
6     16788174   1   1   1   2          0          4          6          0          2          1
6     16788174   1   2   1   2          0          5          10         0          2          2
6     16788174   1   3   1   2          0          8          28         0          4          3
6     16788174   1   4   1   2          0          9          40         0          5          4
```

Figure 3.3: A sample of data from the CSLA show command

and end to end (path) delays in the form of two matrices from the accumulative round trip time at each hop on the path of a probe.

If there are multiple paths possible for a single probe, that probe could have more than one entry in the column, Entry. The parsing software first checks for multiple paths for a probe and if there are any then it can make different sets of paths depending on the number of multiple entries of a probe and the number of probes that have multiple entries.

The hop-wise RTT is cumulative in nature meaning that RTT for the Hop 1 is the RTT from the source to the Hop 1 and RTT for the Hop 2 is the RTT from the source to the Hop 2. Subtracting RTT of the previous hop from the current hop gives the RTT from the previous to the current hop. Therefore the RTT for the last hop represented by the field, Hop, of a probe defines the RTT for that hop and is used to make a matrix of the path delays. The differential delays between the subsequent hops are used to get a matrix of true link delays.

2. Path level delays (V) were the input to NNMF. The Matlab tool NMFpack was used for NNMF factorization. This package is associated with [40]. The NMFpack Matlab

package implements and tests various versions of NNMF with the feature of sparsity. The NMFPack performs standard NNMF with divergence and Euclidean objectives with or without sparseness. NMPack is a comprehensive package with a directory of codes. As per the requirement of this dissertation, some components of the code have been borrowed and customized for the implementation of the NNMF in this dissertation. Both algorithms; divergence and Euclidean algorithms were tested and the results were similar. However, the results included in this dissertation were obtained using the divergence algorithm with the feature of sparsity. Based on the fact that the routing matrix for these test beds is of sparse nature as explained on page 37 in Section 3.5.1, the sparsity for the routing matrix is kept at 0.3 in all the tests and the sparsity of the link delays varies from 0.1 to 0.9 as per Appendix A.

3. The coefficient of correlation between the estimated link delays (H) and actual link delays (X) was determined by using a modified component of EEGLAB [42]. The EEGLAB is an interactive Matlab toolbox for processing continuous and event-related electroencephalography (EEG), magnetoencephalography (MEG) and other electrophysiological data.

   For finding the correlation coefficient, matching rows in two matrices (H and X) were found and their correlation was determined. As a result a column vector of correlation coefficients between the best-correlating rows of matrices H and X was obtained along with the other by-product (routing matrix).

3.5.3 Interpretation of Results

The results were expected to show a strong correlation between the estimated and measured link delays by using NNMF and the true link delays. Three types of results are discussed; the results from numerically simulated data, the results from the first version of the test bed being stressed with extended ping, and the results from the second version of the test bed using a mix of extended ping and traffic from N2X. The next subsection describes the reason for using correlation as a parameter for the comparison of the estimated link delays (H) and the measured link delays (X).

Correlation as a Parameter for Comparison

A statistical metric, correlation, has been considered as a measure for checking the closeness of the estimated link level delays by NNMF (H) to the actual link level delay (X). The correlation coeff cient is used to indicate the relationship of two random variables. It provides a measure of the strength and the direction of the correlation varying from -1 to +1. Positive values indicate that the two variables are positively correlated, meaning the two variables vary in the same direction. Negative values indicate that the two variables are negatively correlated, meaning the two variables vary in the contrary direction. According to the accepted guidelines for interpreting the correlation coeff cient in [38], values between 0.7 and 1.0 indicate a strong positive linear relationship.

Some of the conventional tomographic techniques such as [22, 37] have used statistical parameters such as mean, variance, and standard deviation to compare the recovered and true link matrices. The variance and standard deviation describe how spread out the data is. If all the data lies close to the mean, then the standard deviation will be small, while if the data is spread out over a large range of values, standard deviation will be large meaning that having outliers will increase the standard deviation. Considering these factors and the prominent role of correlation in validity testing as described in [38], the correlation coeff - cient has been preferred over mean and variance. The results in the subsections to follow prove that the correlation values produced by NNMF are close to 1 even without *a priori* knowledge of the routing matrix.

Results from Numerical Simulations

With MatLab simulation, end-to-end link delays were generated as Y (of the size of 4×100) from a known link matrix (X= 10+2$\times$randn(6,100)) and a routing matrix (A= [1 0 1 1 0 0;0 1 1 0 1 0; 1 1 0 0 0 0; 1 0 0 1 0 1]). The end-to-end matrix (Y) was input to NNMF and the estimated matrix (H) was correlated with the known link delay matrix (X). Figure 3.4 shows that the correlation between the estimated link delays (H) and the actual link delays (X) is close to 1 at low sparsity and as the network becomes sparse, the value of correlation decreases.

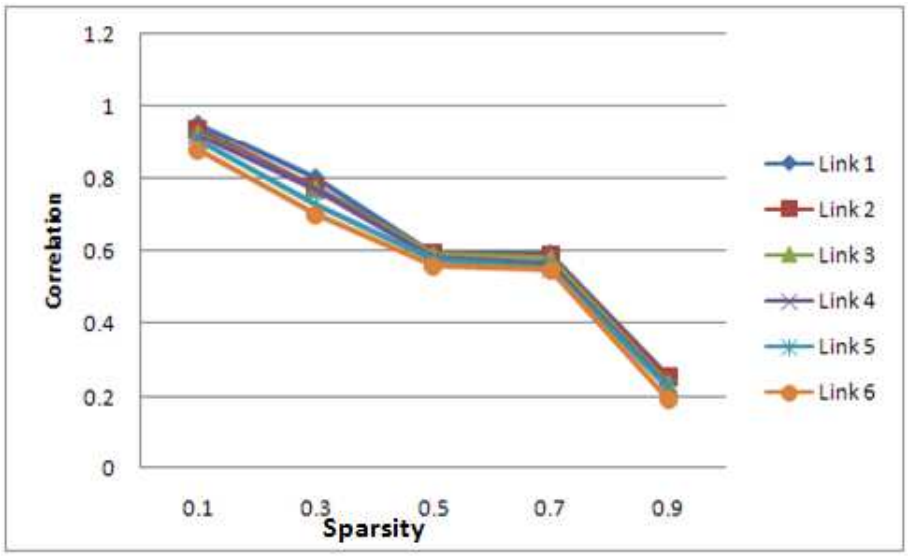

Figure 3.4: Correlation between H and X for numerical data

Results from the Test Bed Stressed with Extended Ping

Figure 3.5 makes it clear that when selected links were stressed with extended pings of 100 Bytes and 200 Bytes, the correlation between the estimated link delays (H) and the actual link delays (X) was close to 1 when this network was using more links (low sparsity). When the sparsity of this network increases, the degree of correlation decreases. It was observed that the correlation was high at low sparsity and this was because low sparsity resembled an actual network with all the links in use. In contrast, the high sparsity accounts for a subset of actual network links. As the links were being stressed with a predictable set pattern (constant extended pings of 100 Bytes and 200 Bytes during the entire time of data collection), the correlation lines for various links at different sparsity levels were spaced differently in Figure 3.5 as compared to the numerical simulation results (Figure 3.4). This depicts the behavior of a practical network due to the transmission delay of the CSLA packets and the delay added by the extended pings.

Results from the Test Bed Stressed with Extended Ping and N2X Traff c

One of the components of the link disturbance was introduced by N2X with a larger and variable packet size. The correlation between the estimated link delay (H) and actual link delay (X) is close to 1 in Figure 3.6 at low sparsity, which is similar to Figure 3.4

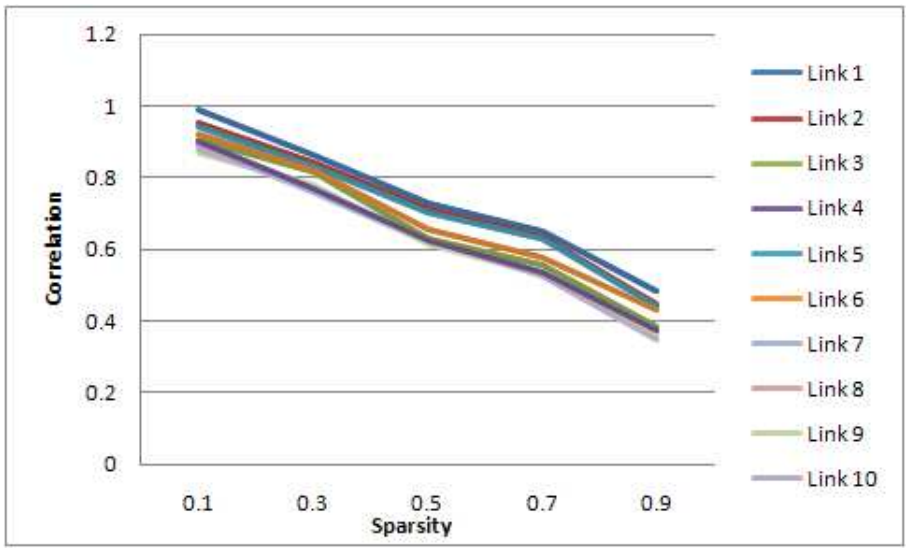

Figure 3.5: Correlation between H and X with extended pings only

and Figure 3.5. This shows that NNMF is able to successfully estimate the link delays under different traffic conditions. However, at high sparsity (right hand side of Figure 3.6), Figure 3.6 is different from Figure 3.4 and Figure 3.5. The reason for such behavior is the combination of the variable delay introduced by N2X and the other traffic patterns that are described as follows:

1. Four out of the six probes use Link 1 and three use Link 5. The link statistics collected for Link 1 and Link 5 have more reliable and verifiable information than the other links as comparatively more samples are collected in a given time at Link 1 and Link 5.

2. Though Link 2, Link 3, and Link 6 have two probes passing though them, Link 2 also has Probe 4 passing through it and it shows better correlation at high sparsity. This is because Probe 4 ends before entering Link 4 and Link 5 and because N2X traffic passes through Link4 and Link 5.

3. Link 2 has Probe 3 passing through it, and Probe 3 also passes through Link 4, which has additional delay due to N2X traffic passing through it. This is why the correlation for Link 3 shows less improvement at high sparsity than that for Link 2.

4. For implementing increased sparsity, NNMF models fewer links out of the available

links (ten links in our case). The result in Figure 3.6 reveals that Link 1, Link 2 and Link 5 show better correlation at all levels of sparsity based on the reasons described in items 1 and 2.

5. Link 4 and Link 5 have N2X traffic passing through them, and this traffic is faster than the CSLA traffic. Therefore, the correlation for Link 4 and Link 5 improves (as compared to Figure 3.5) less than Link 1 and Link 2 (in Figure 3.6) at high sparsity.

6. At Link 6, Link7, Link 8, Link 9 and Link 10, the traffic pattern remains the same in both the test beds; that is why the values of correlation for these links follow the same pattern in Figure 3.5 and Figure 3.6 at all levels of sparsity.

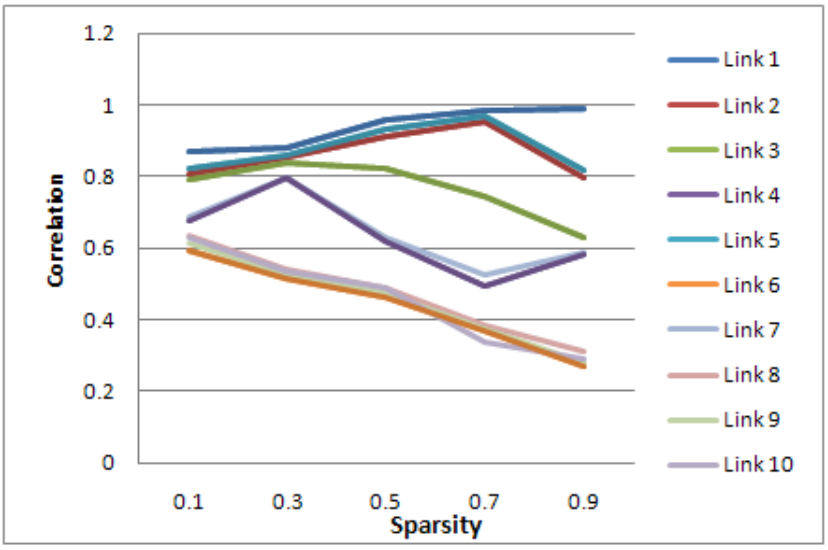

Figure 3.6: Correlation between H and X with a mixture of extended pings and N2X traffic

This discussion reveals an encouraging fact that if the pattern of stressing links is changed, NNMF takes this into consideration and this characteristic is clear in the results (Figure 3.6 in comparison to Figure 3.5).

Comparison with Conventional Tomography Methods

An examination of the results presented in conventional tomography methods such as [22, 37] shows that these authors have used statistics such as mean and variance (for comparing estimated and measured link delays) instead of correlation with the assumption of a known

routing matrix. As per the discussion in the beginning of this subsection, it is believed that correlation is a better statistical measure than mean and variance. In this analysis, NNMF has shown a correlation between the estimated link delay (H) and the actual link delay (X) close to 1 with no prior knowledge of the routing matrix.

3.6 Chapter Summary

In summary, the elimination of the assumption of a known routing matrix as a part of conventional tomography has been discussed in this chapter. One of the BSS methods, sparse NNMF, has been identified as a method which could recover the desired parameters without knowing the routing matrix. With the numerical simulations and the experiments to obtain data from laboratory test beds, it has been shown that NNMF can perform network tomography with accuracy without assuming a known routing matrix. The correlation between the estimated and actual link delays was close to 1. If the characteristics of the network change, as was done by the introduction of N2X in the test bed, the results obtained from NNMF reflect the changes in the network traffic by showing better correlation for some links even on high sparsity.

Chapter 4

Error Modeling in Network Tomography by Sparse Code Shrinkage Method

4.1 Introduction

All practical networks have the potential of errors that should be reflected in the basic network tomographic model ($Y = AX$) as shown in the equation below,

$$Y = AX + \varepsilon, \tag{4.1}$$

where ε represents the error in the model. There are various sources that contribute towards the error term (ε) such as Simple Network Management Protocol (SNMP) operation and NetFlow measurements. The heterogeneity of the network components in terms of vendors and hardware/software platforms, that are used by various types of networking technologies, is also a contributing factor toward the error term, ε.

For this contribution, Sparse Code Shrinkage (SCS) (a blind technique) has been applied to denoise the noisy link delay data. SCS exploits the statistical properties of the data to be denoised. A key idea that constitutes the rationale behind SCS is to use a basis that is more suitable for the data at hand. For denoising, it is required to transform data to a sparse code, apply MLE procedure component-wise, and transform back to the original variables. Originally, SCS is an image denoising technique; this contribution has employed SCS for the first time for estimating the error free link delays from the erroneous link delay data. The simulation results show that the SCS based denoising technique successfully denoises the noisy data and recovers almost noise free (original) data. The rest of this chapter is organized as follows. Section 4.2 briefly describes network tomography and various sources of error that may be present in tomography data. Section 4.3 presents related work. Section 4.4 discusses SCS and the rationale for using SCS. Section 4.5 explains the application of NNMF in the context of network tomography and sparsity. Section 4.6 presents and discusses results to show that SCS successfully denoises noisy link delay data in a blind manner without *a priori* knowledge of the routing matrix. Section 4.7 summarizes the

chapter.

4.2 Error Sources in Network Tomography

SNMP and NetFlow are the main contributors towards the error term (ε) along with the heterogeneity of the network components in terms of vendors and hardware/software platforms that are used by various types of networking technologies [43, 44, 45, 46].

SNMP is applied for collecting data that is used for management purposes including network delay tomography. SNMP [45] periodically polls statistics such as the byte count of each link in an Internet Protocol (IP) network. In SNMP, the commonly adopted sampling interval is 5 minutes. The management station cannot start the management information base (MIB) polling for hundreds of the router interfaces in a network at the same time (at the beginning of the 5-minutes sampling intervals). Therefore, the actual polling interval is shifted and could be more than 5 minutes.

The traffic flow statistics are measured at each ingress node via NetFlow [43, 44]. A flow is an unidirectional sequence of packets between a particular source and destination IP address pair. For each flow, NetFlow maintains a record containing a number of fields such as source and destination IP addresses, number of bytes and number of packets transmitted etc. This flow level information would be sufficient to provide direct traffic matrix measurement if complete NetFlow data were collected for the entire network. The high cost of deployment limits the NetFlow capable routers. Also, products from vendors other than Cisco have limited or no support at all for NetFlow [43, 44]. Therefore, sampling is a common technique to reduce the overhead of a detailed flow level measurement. The flow statistics are computed after applying sampling at both packet level and flow level. Since the sampling rates are often low, inference from the NetFlow data may be noisy.

Both SNMP and NetFlow use the User Datagram Protocol (UDP) as the transport protocol. The operating nature of UDP may add to the error term of the model due to hardware or software problems resulting in data loss in transit [43, 44, 45, 46].

Having different vendors for network components along with hardware/software platforms that are used by various types of networking technologies and the inherited shortcomings of the distributed computing also contributes toward introducing errors. The risk of errors increases if there are more components in a system. The physical and time separation and consistency are also a problem and a source of error [46].

4.3 Related Work

The authors of [46], when estimating a traffic matrix with imperfect information, have mentioned the presence of errors in network measurements. However, they did not present any solution in particular to the errors in link measurements, though they have considered these errors when they have compared the traffic matrix with and without network measurement errors. A traffic matrix quantifies aggregate traffic volume between any OD pairs in a network, which is essential for efficient network provisioning and traffic engineering. They have applied statistical signal processing techniques to correlate the data obtained from both (SNMP and NetFLow) measurement infrastructures. The traffic under passive tomography was determined by considering a bi-model approach for error modeling in [46]; one model for the SNMP errors and the other model for NetFlow errors. They have also categorized errors in various categories such as erroneous data and dirty data. This dissertation, on the other hand, has used a single model to represent noise irrespective of the nature of the noise source as shown in Equation 4.1. The error model in this chapter is simpler than the model in [46] as it considers all the errors as a single collective parameter, ε, irrespective of the sources that have caused these errors. Though the data for simulations has been collected by active tomography, this method could be applied to any type of tomographic data.

As described in Section 4.2, various kinds of sources introduce errors in the original data and the use of this data for making further estimation can increase the effect of errors. There is a need of some techniques to denoise the erroneous network tomography data and SCS is one such technique. A brief description of SCS is given in the next section.

4.4 Sparse Code Shrinkage (SCS)

The proposed application of SCS [47] proceeds as follow. SCS exploits the statistical properties of the data that is to be denoised. A general model of SCS is $S = ZX$, where S represents mixed observed signals, X is available data (link delays in this dissertation) and Z is a mixing weights matrix. The SCS model may also be represented as $\tilde{S} = Z_s X^e$, where $\tilde{S}$ is the sparse and shrunk observed noisy image (data), Z_s is the sparse mixing matrix, and X^e is the denoised estimation of unobserved independent signals (data). By using a noise-free training set of X, use some sparse coding method for determining a

sparse version of matrix Z (Z_s) so that the components S_i in $S = ZX$ have as sparse a distribution as possible.

In general, SCS consists of three steps as shown in Figure 4.1 and explained as follows:

1. The noisy sparse data (S_s^n) is obtained by multiplying Z_s with noisy X (X^n).
2. Apply the component wise shrinkage non-linearity $g_i()$. Denote the obtained components by $\tilde{S} = g_i(S_s^n)$.
3. Estimate the denoised unobserved signals X (X^e) from $X^e = \tilde{S}Z_s^{-1}$.

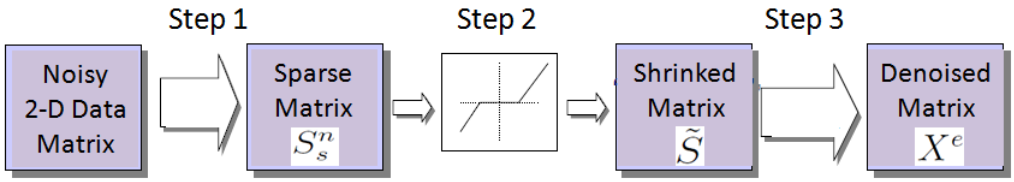

Figure 4.1: Implementation steps of SCS

To estimate the sparsifying transform Z, an access to a noise-free realization of the underlying random matrix is assumed. This assumption is not unrealistic in many applications. For example, in image denoising it simply means that we can observe noise free images that are somewhat similar to the noisy image to be treated, i.e., they belong to the same environment or context. In terms of link delays in networking, it means having link delay readings while a system is operating in normal condition with no abnormalities to cause errors.

4.4.1 Rationale for Selecting SCS

Some other interesting techniques for denoising (of the nature of SCS) were also investigated as part of the literature survey, such as Wiener Filtering and Wavelet Shrinkage methods [47], but SCS was a preferred choice for the following reasons. The Wavelet Shrinkage method adopts a f xed basis to linearly transform image data into another domain where denoising is more tractable. Some of the wavelet methods such as presented in [25] resemble SCS. There are two main differences between the two methods (SCS and Wavelet Shrinkage); the choice of the transformation and the estimation principle. SCS

chooses the transformation using the statistical properties of the data at hand, whereas the Wavelet Shrinkage methods described in [25] use a predetermined wavelet transform. SCS estimates the shrinkage nonlinearities by the maximum likelihood (ML) principle (adapting to the data at hand) whereas some methods [25] use f xed thresholding derived by the min-max principle.

4.5 Sparsity with NNMF

A useful property of NNMF is the ability to produce a sparse representation of data. Such a representation encodes much of the data using a few active components, which makes the encoding easy to interpret. Sparse coding, on theoretical grounds is considered a useful middle ground between completely distributed representations, on the one hand, and unary representations on the other [41]. In terms of network terminology, a highly sparse network means using fewer links out of the total number of links available in a network and a low sparse network is considered closer to the original topology of a network. As the feature of sparsity plays a signif cant role in SCS, NNMF has been considered for the estimation of the sparsifying transformation in the initial step of SCS.
In summary, NNMF is used to factorize a matrix into two factors (possibly matrices with various degree of sparsity) with the constraint that all three matrices must be non-negative.

4.6 Simulation Results of Denoising Tomography Data Through SCS

For validating SCS as a technique to denoise the erroneous link delays, a laboratory test bed was used to collect real link delays. WGN was introduced into the measured link delays to create the effect of errors in the measured link delays. This erroneous data was input to SCS and this data was denoised to get an estimate of the link delays close to the measured link delays. The next subsection describes the test bed that was used for data collection to obtain end-to-end delays and link delays as benchmarks.

4.6.1 Description of Networking Test Bed

The experimental data has been obtained from a test bed that was set up in the Advanced Internetworking Laboratory (AIL) and consisted of six 3800 series Cisco routers, Agilent Router Tester (N2X), and an MRTG capable workstation. OSPF routing was implemented

on routers and N2X. SCS [47] and NNMF [41] are both capable for dealing with scalability in the case of larger networks. The Echopath option of the CSLA was implemented to send four probes and collect the cumulative RTT from a source to each hop. All probes were grouped together. All the probes in the group started at the same time. The group of probes was repeated 100 times with a time difference of 10 seconds between two consecutive repetitions. The selected links were stressed by two sources: extended ping on selected links and traff c injected from the N2X.

Figure 4.1 shows a test bed with the four probes (traveling from right to left) and two of the links (Link 1 and Link 6) stressed with an extended ping of 200 Bytes. The other source of disturbance was the traff c from the Agilent router tester (N2X). The module 1 of N2X was generating a variable packet size from 1000 Bytes to 1500 Bytes. The size of the probes in CSLA was 10 Bytes for this scenario. The condition of the network remains unchanged during the CSLA operation.

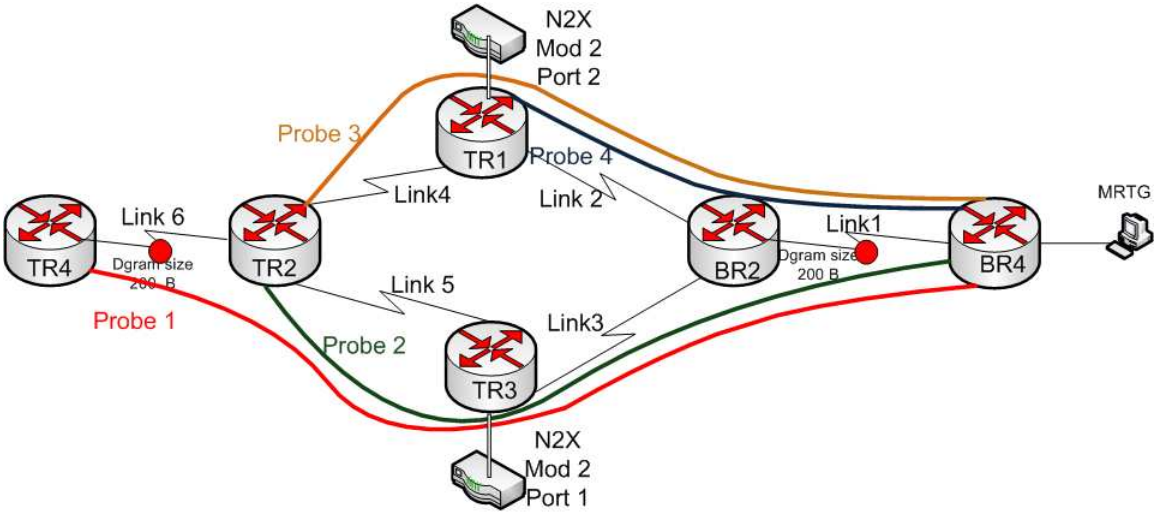

Figure 4.2: Testbed setup for denoising with SCS

4.6.2 Use of Data from Test Bed

Original link delays in Link 1 to Link 6 of Figure 4.1 were collected. The data obtained from the CSLA was in the form of accumulative hop-wise round trip time. The following steps were followed to process the data for obtaining two matrices; a matrix of end-to-end

delays and a matrix of link level delays. Parsing software, written in Java, extracted link delays and end to end delays in the form of two matrices. From the accumulative round trip time from source to each hop, hop to hop delays were calculated to form the delay matrix. From the accumulative round trip time (from the source to the destination), the end-to-end delay matrix was determined. This data was used as a baseline for judging the accuracy to the SCS.

The WGN was simulated through a MatLab based function and measured link delays were converted into noisy link delays. This noisy data was used as an input to SCS. SCS was expected to denoise this data in such a way that the denoised link delays were in close proximity of the measured link delays.

As part of the SCS, there was the need to apply a BSS technique as a sparse coding method for determining the sparse matrix Z_s so that the components S_i in $S = ZX$ had as sparse distributions as possible. NNMF was applied for this purpose. The end-to-end link delays obtained from CSLA were input to NNMF. The MatLab tool NMFpack [40] was used for NNMF factorization . Various combinations of measured link delays and the routing matrix with various sparsity levels were tried to get S_i as sparse as possible. These sparse estimation of S_i were input to step 2 of the implementation of SCS as described in Section 4.4. The following are the steps of SCS in the context of denoising the noisy link delays:

1. Actual noise free end-to-end delays (S) as the training set and link delays (X) for bench marking were measured.

2. With the noise free training set of end-to-end link delays (S), a sparse coding method (NNMF) determined the sparse version of matrix Z_s, so that the row components S_i of $S = ZX$ have as sparse a distribution as possible. Then this sparse mixing matrix Z_s was used in the following steps. Originally, SCS uses ICA in [47] for the estimation of the sparsifying transformation. However, this dissertation used NNMF instead of ICA. The ICA approach may result in negative values in estimated matrices whereas all the matrix components in NNMF are always positive.

3. To emulate the effect of the noisy measurement, White Gaussian Noise (WGN) was introduced to the actual noise free link delay (X) to get a noisy version of link delays (X^n), and the sparse and noisy end-to-end delays (S_s^n) were obtained by multiplying

this noisy link delays (X^n) with the sparse mixing matrix Z_s. The signal power of X in terms of the squared mean of X was 0.486 msec^2 in this contribution. The WGN was generated with the function (wgn(R,C,20$\times$log10($s\_n\times$ stdX)), where R, C, and stdX are dependent on X and s_n is user def ned. Another function (addGauss(X, mu, s_n)) returns X^n and the variance of the Gaussian Noise by having X and noise parameter (zero mean, unit variance) as inputs (please refer to [47] for further details). Noisy and sparse end-to-end delays S_s^n were input as argument u to the following nonlinearity shrinkage function to get $\tilde{S} = g_i(S_s^n)$.

$$g(u) = 1/(1 + \sigma^2 a) sign(u) \max(0, |u| - \sigma^2), \tag{4.2}$$

where σ^2 is the noise variance and its value was 14.909 msec^2 in this contribution. The effect of the shrinkage function is to reduce the absolute value of its argument by a certain amount, which depends on the parameters. Small arguments are thus set to zero.

4. Estimations of denoised link delays were obtained from $X^e = Z_s^{-1}\tilde{S}$.

4.6.3 Comparison of Measured, Errored, and Denoised Link Delays

The results have been displayed in six diagrams (Figure 4.3 to Figure 4.8). Each diagram representing one link, from Link 1 to Link 6. In each diagram, three types of data lines are shown:

1. The actual measurement of the link delays collected from CSLA is shown as solid lines.

2. The link delays after the introduction of the error are shown as the dotted lines.

3. The denoised link delays after the application of SCS are shown as dashed lines.

The vertical axis represents the link delays and horizontal is the number of samples at various times.
It is clear from these six graphs that the denoised link delays are in close proximity to

the actual link delays. The errored link delays were input to SCS and the estimated (denoised) values of link delays are close to the measured values. This shows that the SCS has successfully denoised the noisy link delay data and denoised data is in the proximity of benchmarks.

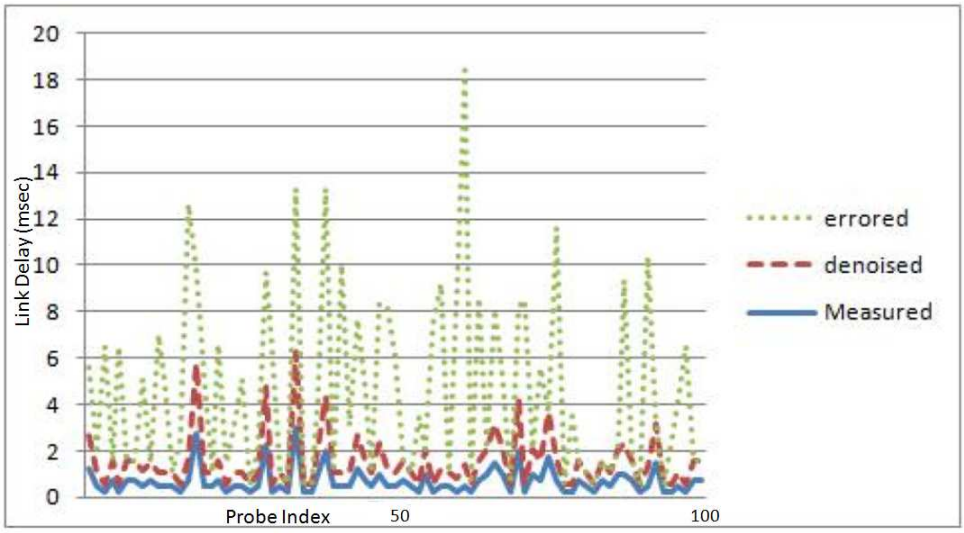

Figure 4.3: Comparison of measured, errored, and denoised link delays on Link1

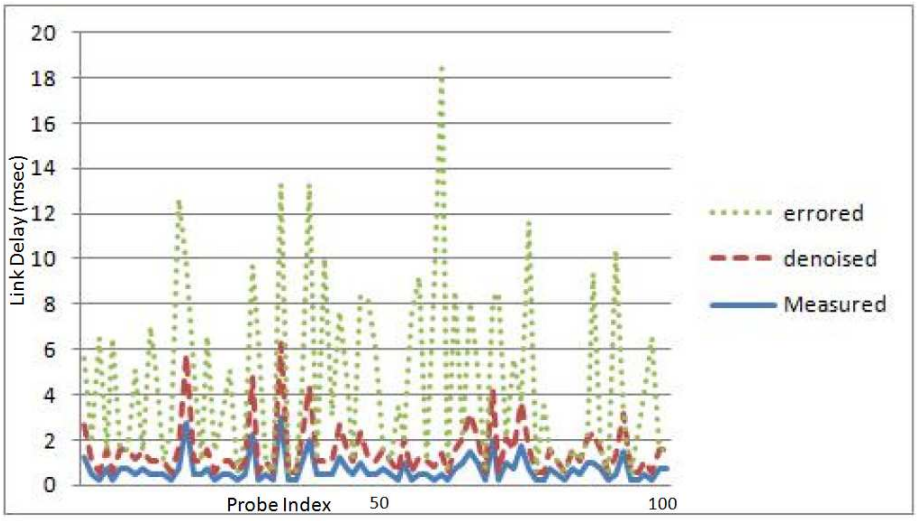

Figure 4.4: Comparison of measured, errored, and denoised link delays on Link2

A comparison of the Mean Squared Errors (MSE) in Table 4.1 between measured and noisy link delays, and MSE (in $msec^2$) between measured and denoised link delays shows

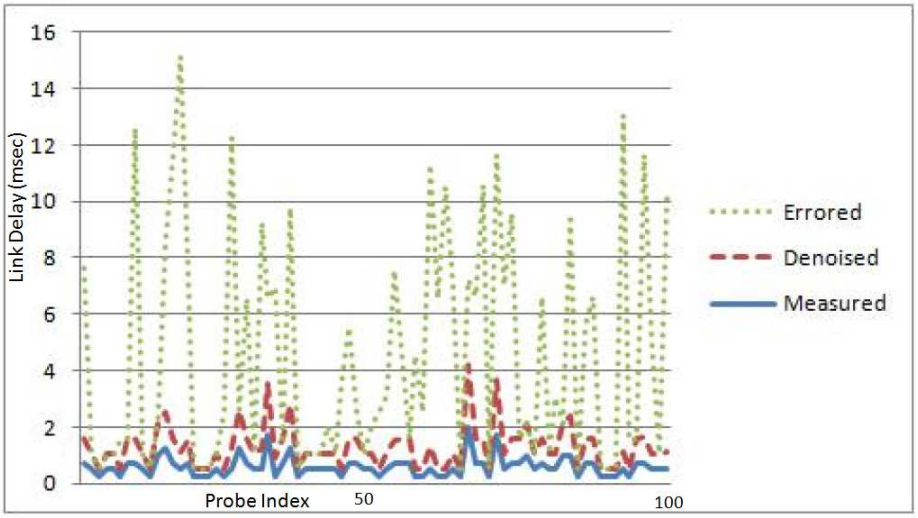

Figure 4.5: Comparison of measured, errored, and denoised link delays on Link3

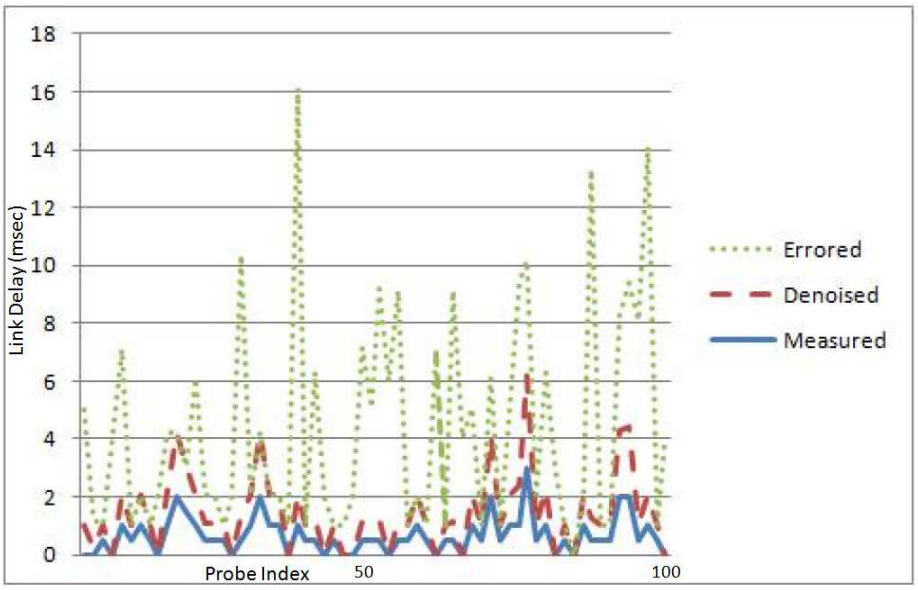

Figure 4.6: Comparison of measured, errored, and denoised link delays on Link4

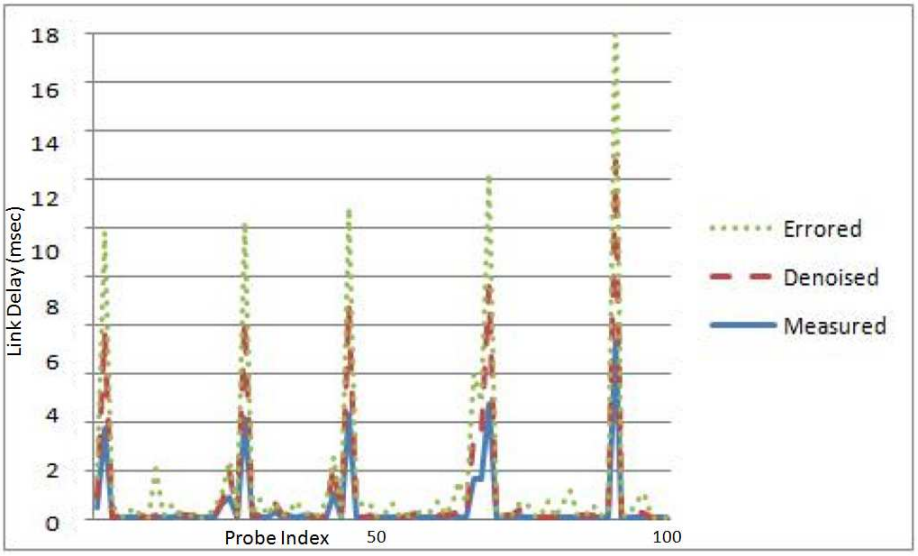

Figure 4.7: Comparison of measured, errored, and denoised link delays on Link5

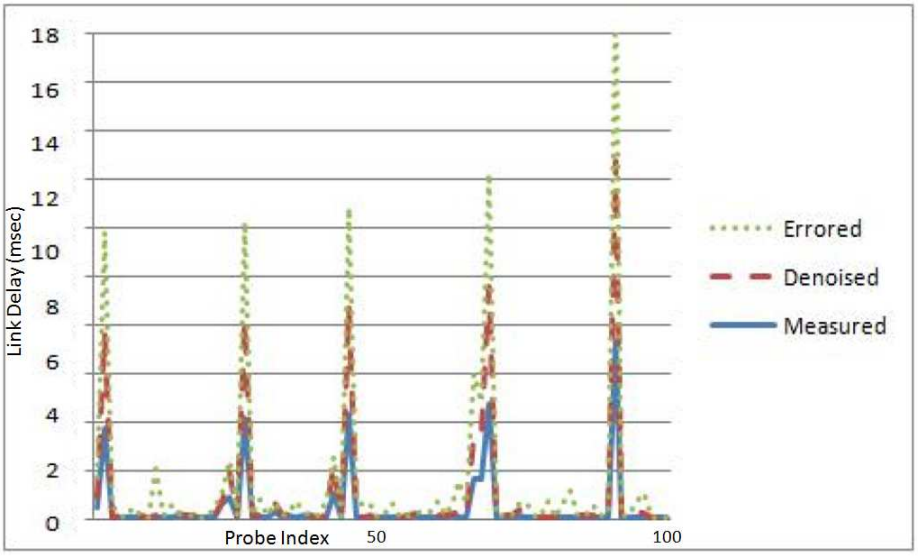

Figure 4.8: Comparison of measured, errored, and denoised link delays on Link6

that the MSE between measured and denoised link delays is less than the MSE between measured and noisy link delays on all the links of the test bed.

Table 4.1: Comparison of the MSE between measured and noisy link delays, and the MSE between measured and denoised link delays

	Link 1	Link 2	Link 3	Link 4	Link 5	Link 6
MSE between measured and noisy link delays	25.490	17.289	16.546	16.546	17.423	27.345
MSE between measured and denoised link delays	0.018	0.018	0.013	0.036	0.818	0.604

.

4.7 Chapter Summary

To summarize, in this chapter, the SCS technique was applied to denoise the network tomography model with errors. To f t well to the research objectives of this dissertation, SCS was modif ed by replacing ICA with NNMF to get positive values in the estimated link delay matrix. The results obtained from the laboratory test bed based experiments proved that SCS successfully denoised the link delays. The comparison of denoised link delays with the error free benchmark data showed them in close proximity.

Chapter 5

Multi-metric Network Tomography

5.1 Introduction

The conventional tomography model ($Y = AX$) uses a single measured parameter matrix (Y) to estimate a matrix of a single unknown parameter (X). In contrast to the conventional model, this chapter proposes the use of direct measurements of multiple metrics to recover indirectly a single parameter with expectation of getting a better estimate as compared to using a single directly measured parameter to estimate a parameter indirectly. The new model is represented by Equation 5.1, where Y_1 and Y_2 are directly observed in order to estimate X indirectly by solving the following inverse equation.

$$Y_1 Y_2 = AX \tag{5.1}$$

For example, instead of recovering link delays from merely end to end path delays, we can estimate link delays from a combination of path delays (Y_1) and PLR (Y_2). By having a better input in terms of two interdependent metrics, multi metrics network tomography produces a better estimation than using only one parameter such as path level link delays. This correlation of two network parameters has been discussed in the literature. For example, the authors of [10] report on the correlation between delay and loss observed by a continuous-media traff c source. This study [10] determines the extent to which one performance measure could be used as a predictor of the future behavior of the other (for example, an increasing delay is a good predictor of future loss) so that an adaptive continuous media application might take anticipatory action based on the observed performance. A variation of NTF called the NTF1 model has been applied for this purpose.

The rest of the chapter is organized as follows. Section 5.2 reviews the related work. Section 5.3 discusses NTF. Section 5.4 details simulation arrangements. Section 5.5 presents and discusses results. Section 5.6 serves as the chapter summary.

5.2 Related Work

This section investigates related work in the domains of the multiple metric network tomography concept and the interdependence of delays and PLR.

5.2.1 Multi-metric versus Additive Metrics

To best of the author's knowledge, there has never been an implicit consideration of directly measured multiple metrics for an indirect estimate of a network metric. In [48] however, there is evidence of considering multiple metrics in the form of additive metrics. A framework was proposed for analyzing topology using ideas and tools from phylogenetic inference in evolutionary biology. The phylogenetic inference problem determines the evolutionary relationship amongst a set of species. The framework is built upon additive metrics. Under an additive metric the path metric (path length) is expressed as the summation of the link metrics (link lengths) along the path. In [48], the intent is to use estimated distances between the terminal nodes (end hosts) to infer the routing tree topology and link metrics. Based on the framework some inference algorithms have been presented as an alternative to network tomography.

They [48] consider that G = (V, E) denotes the topology of the network, which is a directed graph with node set V (end hosts, internal switches and routers, etc.) and link set E (communication links that join the nodes). For any nodes *i* and *j* in the network, if the underlying routing algorithm returns a sequence of links that connect *j* to *i*, they say *j* is reachable from *i*. They call this sequence of links a path from *i* to *j*, denoted by P(*i*, *j*).

As per their terminology, *d(e)* can be viewed as the length of link *e*, and d(*i*, *j*) can be viewed as the distance between nodes *i* and *j*. Basically, an additive metric associates each link on the tree with a f nite positive link length, and the distance between two nodes on the tree is the summation of the link lengths along the path that connects the two nodes. Suppose $T(s, D) = (V, E)$ is a routing tree with source node s and a set of destination nodes (*D*). Let

$$d(E) = d(e) : e \epsilon E \tag{5.2}$$

denotes the link lengths of $T(s, D)$ under additive metric *d*. Remember $U = s \bigcup D$ is the set of terminal nodes on the tree. Let

$$d(U^2) = d(i, j) : i, j \epsilon U \tag{5.3}$$

denotes the distances between the terminal nodes.

The above review makes it clear that considering additive metrics is different from multiple metrics based network tomography. Actually, this phylogenetic based technique is claimed to be an alternative to network tomography [48]. Therefore, the work in this thesis of considering multiple metrics is a novel way of improving the conventional mono-metric network tomography model.

5.2.2 Correlation of Link Delays and PLR

The authors of [10] examine the correlation between packet delay and packet loss experienced by a continuous media traff c source. They [10] study the extent to which one performance measure can be used to predict the future behavior of the other (for example, whether an observed increasing delay is a good predictor of future loss) so that an adaptive continuous media application might take anticipatory action based on the observed performance. They provide a quantitative study of the extent to which such correlation exists. There are two examples in this regard mentioned in the following discussion.

When the buffer reaches its capacity, packet losses begin to occur. The receiver of the continuous-media application thus sees increased delay, and eventually losses.

When packets from a continuous-media application arrive at a buffer that is already full, they are dropped. As other sources (for example, TCP connections) detect congestion and decrease their transmission rate, the queue length at the buffer will decrease, and packets from the continuous-media application will start to be queued, rather than dropped. The receiver sees losses followed by high, but possibly decreasing, packet delays.

The authors of [10] introduce a lag, loss-conditioned average delay, in calculating the average delay conditioned on loss. Specif cally, the average packet delay, conditioned on a loss occurring at a time lag *j* packets in the past, is the average delay of all packets in the trace that have a loss *j* packets before them in the trace. That is,

$$E[d_i \mid l_{i-j} = 1] = \sum_{k \epsilon P} d_k / \mid P \mid, \tag{5.4}$$

where $P = k : l_{k-j} = 1$ and $l_k = 0$.

If the loss-conditioned average delay at a positive lag of *j* is higher than the unconditional average delay (that is, the delay averaged over all received packets), the packets that arrive *j* packets after a loss have a higher average delay than the unconditional average delay. That

is, a loss occurring *j* packets in the past can be taken as a precursor to a higher delay later. This discussion shows that delay and PLR are interdependent and correlated based on loss-conditioned average delay. This evidence motivated the consideration of multi metric network tomography in this dissertation. NTF has been applied to carry out the multiple metric network tomography and NTF is brief y described in the next section.

5.3 Nonnegative Tensor Factorization (NTF)

During the background research for f nding a mathematical technique that could deal with multiple metrics and be capable of matrix factorization, NTF appeared to be an appropriate choice, because it can manage a direct estimation of a matrix from the input of multiple matrices. In this chapter, two parameters (PLR and link delay) are measured directly to be applied to estimate the link delays. There is a need for a mathematical model that could take inputs of two or more matrices and could give estimates of one or more parameters. NTF, a matrix factorization technique, is capable of fulf lling this objective. Matrix factorization is an important area in signal processing and linear algebra, with applications in many other areas. BSS and related methods, for example, ICA, employ a wide range of unsupervised learning algorithms and have found important applications from engineering to neuroscience.

Some of the concepts involved in NTF are explained (as in [41]) in the following paragraphs of this section. NTF utilizes tensors, which are generalizations of vectors and matrices. For example, a third-order tensor (or three-way array) has three modes (or indices or dimensions). Many modern applications generate large amounts of data with multiple aspects and high dimensionality for which tensors (i.e., multi-way arrays) provide a natural representation. These include text mining, clustering, Internet traff c, telecommunication records, and large-scale social networks [41].

A tensor is a multi-way array or multi-dimensional matrix. The order of a tensor is the number of dimensions, also known as ways or modes.

A formal def nition of a tensor follows: let $I_1, I_2, \cdot, \cdot, \cdot, \cdot, I_N \epsilon N$ denote index upper bounds. A tensor $\underline{Y} \epsilon R^{I_1, I_2, \cdot, \cdot, \cdot, \cdot, I_N}$ of order N is an N-way array where elements $y_{i_1, i_2, \cdot, \cdot, \cdot, \cdot, i_N}$ are indexed by $i_n \epsilon$ 1,2,$\cdot, \cdot, \cdot,I_n$ for 1$\leq$ n $\leq N$.

Unfolding or matricization or f attening is a process of reordering the elements of an N^{th} order tensor into a matrix. There are various ways to order the f bers of tensors, therefore,

the unfolding process is not unique.

Higher-order tensor decompositions are frequently used in a variety of fields including psychometrics, image analysis, graph analysis, and signal processing. Two of the most commonly used decompositions are the Tucker Decomposition and Parallel Factorization (PARAFAC), which are often considered as higher order generalizations of the matrix SVD or PCA.

The PARAFAC algorithms decompose a given tensor into a sum of multi-linear terms, in a way analogous to the bilinear matrix decomposition. Unlike SVD, PARAFAC usually does not impose any orthogonality constraints. A model which imposes nonnegativity on factor matrices is called the NTF or Nonnegative PARAFAC [41].

Figure 5.1 illustrates one of the three ways of the basic 3D NTF1 model, which is an extension of the NTF model. The NTF1 model, given a three-way (third-order) tensor formed by a set of matrices $Y_q \epsilon R_+^{I \times T_q} (q = 1, 2, \cdot, \cdot, \cdot, \cdot Q)$, formulates a set of nonnegative and sparse matrices $A\ \epsilon R_+^{I \times J}$, $C\ \epsilon R_+^{Q \times J}$, and $A\ X_q \epsilon R_+^{I \times J}$ for $q = 1, 2, \cdot, \cdot, \cdot, \cdot Q$ with reduced dimensions ($J << I < T_q$).

Global matrix representation using row-wise unfolding of the three-way array is shown in Figure 5.1 and is expressed (error free model) as $Y_q = AD_qX_q$. Thus, only the mixing matrix A and the set of scaled source matrices X_q need to be found whereas due to scaling ambiguity the matrix C does not need to be calculated explicitly [41].

There are several possible approaches to find or identify an extended NTF1 model such as global strategy, or local strategy, or a combination of both. A global strategy based on alternating minimization of cost function is shown in Equation 5.5.

$$D_F(\overline{Y} \| A\overline{X}) = \frac{1}{2} \| \overline{Y} - A\overline{X} \|_F^2 \tag{5.5}$$

A local strategy based on alternating minimization of cost function is shown in Equation 5.6.

$$D_F(Y_q \| AX_q) = \frac{1}{2} \| Y_q - AX_q \|_F^2 \ \ (q = 1, 2, ., ., ., ., ., Q) \tag{5.6}$$

The local strategy (based on alternating minimization of cost function) has been applied in this dissertation. The setup of simulations is discussed in the next section.

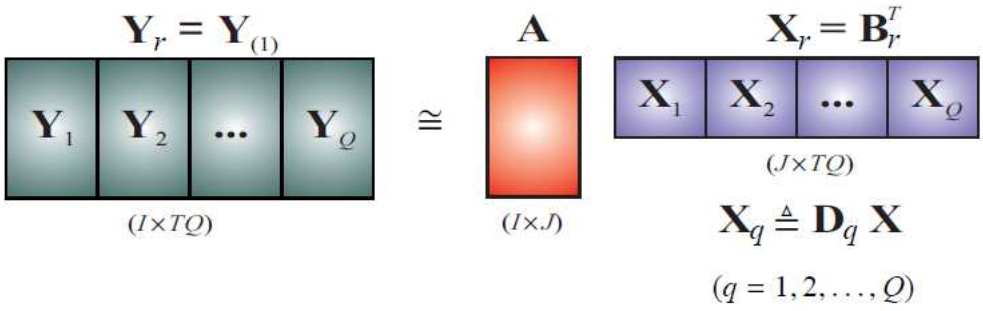

Figure 5.1: Decomposition into two matrices using row-wise unfolding [41]

5.4 Simulation Arrangement for Multi-metric Network Tomography

The experimental data was obtained from a test bed that consisted of six 3800 series Cisco routers and four Cisco IPTV nodes as shown in Figure 5.2. Two types of traff c was utilized in the test bed: a multimedia traff c through Cisco IPTV setup and Path Echo option of Cisco CSLA. The Cisco IPTV source was connected to the router, BR4, and all the four probes were also initialized from the same router. Routers TR1, TR2, and TR4 were the recipients of the multimedia transmission from the cisco IP/TV and the four probes.

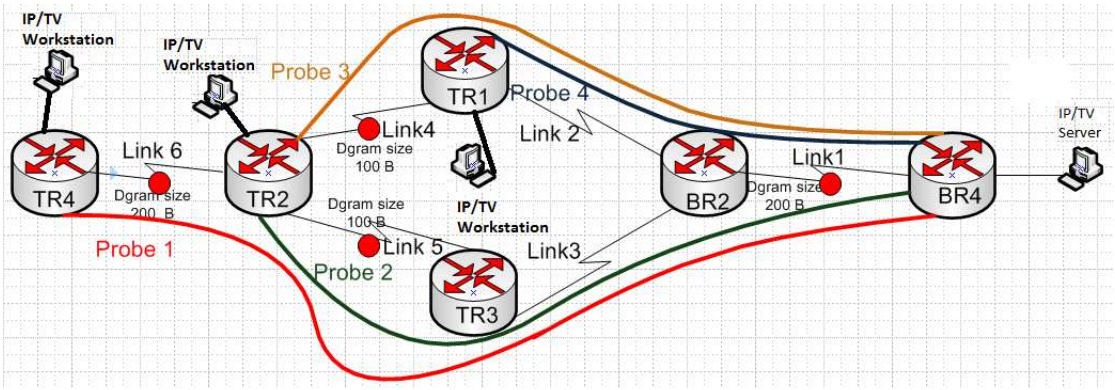

Figure 5.2: Testbed setup for multiple network tomography

5.4.1 Estimation of Link Delay from Path Delays

In this first part of the simulation, the link delays were estimated from path level delays and then the correlation between the estimated and measured link delays was determined. The Echopath option of the CSLA was implemented to send four probes and collect the cumulative RTT from source to each hop. All probes were grouped together. All the probes in the group start at the same time. The group of probes was repeated 100 times with a time difference of 10 seconds between two consecutive repetitions. The selected links were stressed with extended ping.

Figure 5.2 shows the test bed with the four probes and two of the links (Link 1 and Link 6) were stressed with an extended ping of 200 Bytes. The condition of the network remained unchanged during the CSLA operation.The data obtained from the CSLA is in the form of accumulative hop-wise RTT. The following steps were followed to process the data to obtain two matrices; a matrix of end-to-end delays and a matrix of link level delays.

Parsing software, written in Java as explained in Section 3.4.2, extracted link delays and end to end (path) delays in the form of two matrices. From the accumulative RTT from source to each hop, hop to hop delays were calculated to form the link delay matrix. From the accumulative RTT from the source to the destination, end to end delay matrix was determined. Path level delays (V) were input to NNMF. The MatLab tool NMFpack [40] was used for NNMF factorization. The coefficient of correlation between the estimated link delay (H) and actual link delay (X) was determined by using a modified component of EEGLAB.

For finding the correlation coefficient, matching rows in two matrices (H and X) were found and their correlation was determined. As a result, a column vector of correlation coefficients between the best-correlating rows of matrices H and X was obtained.

Figure 5.3 shows the correlation between the estimated and true link delays.

5.4.2 Estimation of Link Delay from a Combination of Path Delays and Packet Loss Rate (PLR)

In the second part of the simulation, it was desired to estimate link delays from a combination of path level delays and PLR data. The correlation between the estimated and the measured link delays was measured again and it was expected that this correlation would be better than the correlation shown in Figure 5.3. In the same test bed, two types of traffic

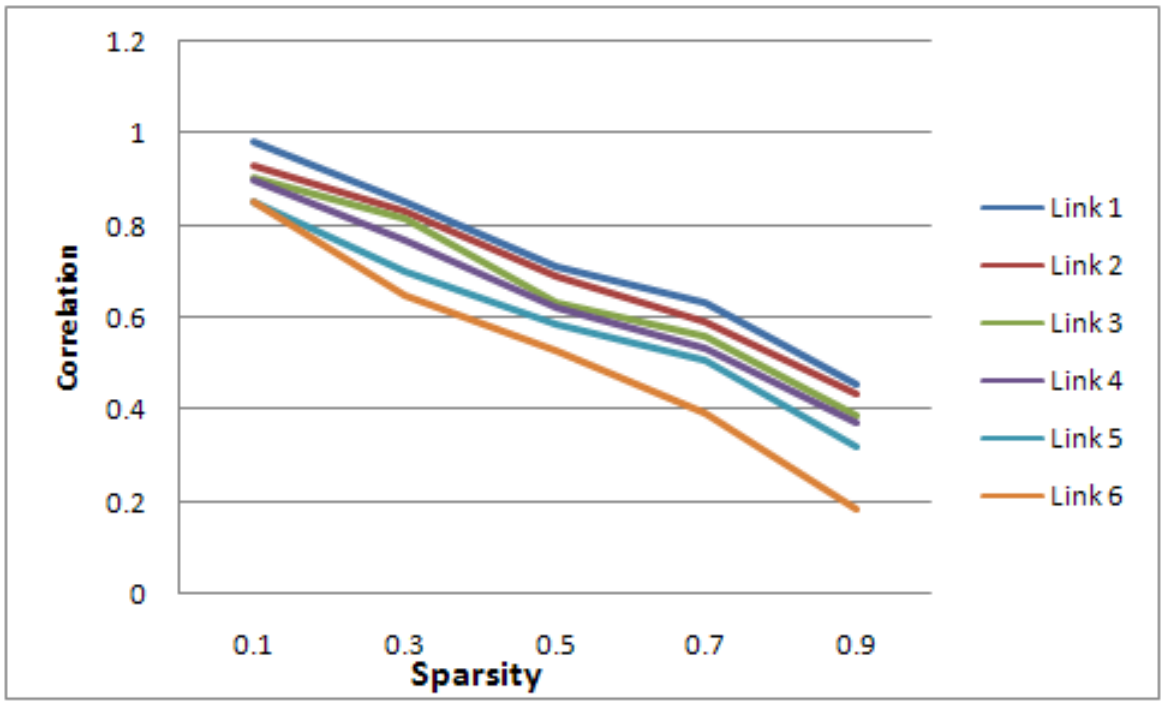

Figure 5.3: Correlation between the true link delay and the estimated link delay from a single parameter; path delays

were injected; CSLA and Cisco IP/TV. A combination of the path level delay and PLR was obtained. This data was used to recover link level delays by inputting this data to row wise unfolding of NTF1 model with local strategy based on alternating minimization of the cost function.

Quality of service (QoS) provides a better service to certain f ows over other f ows. This is done by either raising the priority of a f ow or limiting the priority of another f ow. The Cisco IP/TV traff c and the CSLA traff c were identif ed as the two traff c classes and the Cisco IP/TV traff c class was provided preferential service over the CSLA traff c.

Two types of CSLA ping traff c were used in this experiment as explained below:

1. The Path Echo operations record statistics for each hop along the path that the operation takes to reach its destination. The IP/ICMP Path Echo probe computes this hop-by-hop response time between a Cisco router and any IP device on the network by discovering the path using traceroute.

   The UDP Plus operation is a superset of the UDP Echo operation. In addition to measuring UDP round-trip time, the UDP Plus operation measures per-direction packet-loss. Packet loss is a critical element in CSLAs. Packet loss is reported for how many packets are lost, and in which direction (source to destination or destination to source). A show command of CISCO IOS displays various parameters including *PacketLossSD* and *PacketLossDS*, that are packet loss from source to destination and destination to source packet loss, and these two parameters give values at each repetition of the show command. These two parameters (*PacketLossSD* and *PacketLossDS*) were instrumental in determining PLR for this part of the simulation. The observed PLR on multimedia traff c was in the range of 5 to 15 percent due the combination of quality of service enforcement on two types of traff c (ping and multimedia) and traff c disturbances from extended pings.

2. Cisco IP/TV multimedia traff c was sent from the source (BR4) to workstation connected to TR4, TR2, and TR1, Figure 5.2. Cisco IP/TV servers use IP multicast to optimize bandwidth for live and scheduled video, broadcasting a single, real-time stream per program over the network, regardless of audience size.

Cisco IP/TV multicast application solves the scalability limitations of unicast. The primary advantage of multicast over unicast is that it replicates the video stream closest to users

at the last possible point in the network, as opposed to unicast, which replicates a single video stream for each user at the source. The video and ping traff c were classif ed into two classes by using the QoS option of Cisco IOS.

Matrices of the end-to-end delays, and PLR were input to the modif ed components of NTF MatLab toolbox to get the matrix of link delays. The matrix of the measured link delays has already been estimated in Subsection 5.4.1. Figure 5.4 shows a correlation between the estimated and true link delays. By comparing Figure 5.3 and Figure 5.4, it is evident that

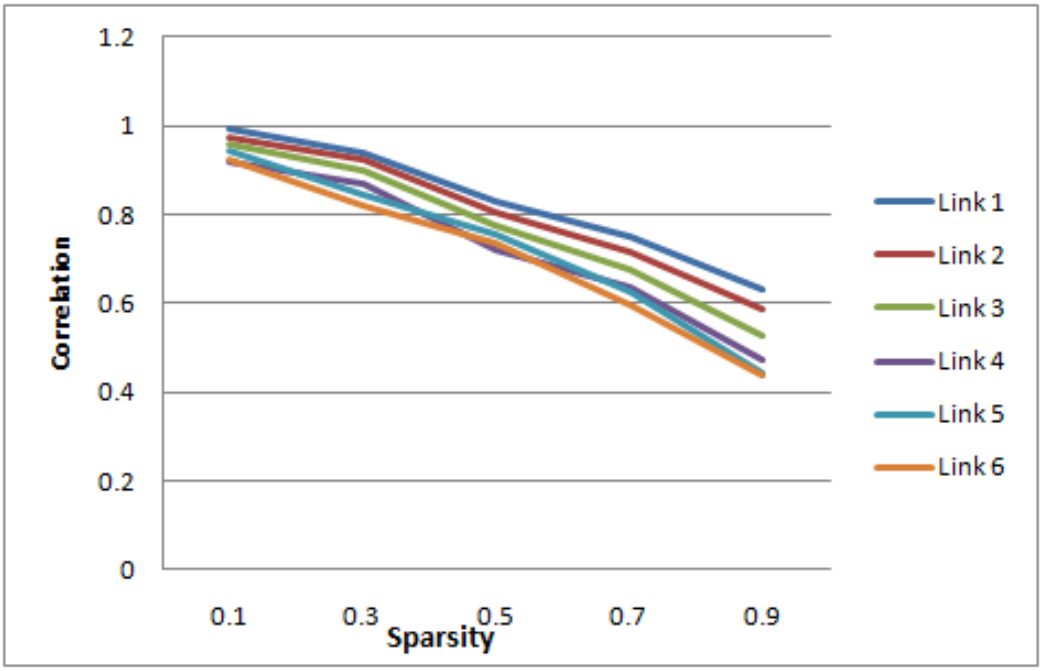

Figure 5.4: Correlation between the true link delay and the estimated link delay from two parameters; path delay and PLR

the estimation of the link delays by using the input of two matrices, the path delays and the PLR is better than the estimation of the link delays from a single metric of the path delays.

5.5 Chapter Summary

In summary, this chapter introduced a novel concept of multiple metrics network tomography. Link delays were estimated from path delays for the mono-metric network tomography and from the multi metrics network tomography. The estimated link delays by using multi metric (path delays and PLR)were better than the estimated link delays by using a

mono-metric of the path delays.

Chapter 6

Distributed Network Tomography

6.1 Introduction

All types of network tomography, active, passive, and technology identif cation, work in a central manner. There is a single data repository, that is the node where network tomography is being performed. This single node may be the node that sends unicast or multicast in active tomography, or the node sensing the OD traff c matric from link measurements. A single data repository brings its challenges such as susceptibility to redundancy, and computation and communication complexities. These problems can be taken care of by multiple data repositories in the form of a distributed system.

Coulouris [49] def nes a distributed system as a system in which hardware or software components located at networked computers only pass messages to communicate and co-ordinate their actions. The main features of a distributed system include [49, 50] functional separation, reliability, scalability, and economy.

In general, the functional separation is based on the functionality/services provided, capability and purpose of each entity in the system, for example more than one node performing network tomography at the same time and using even different methods of network tomography.

Reliability is implemented by the long term data preservation and backup (replication) at different locations. Scalability enables the addition of more calculating nodes to include additional networks without affecting the performance of the existing networking infrastructure.

As a consequence of these features, the various entities (nodes performing tomography) in a distributed system can operate concurrently and possibly autonomously. Estimation of desired network parameters by means of network tomography can be carried out independently and the distributed results co-ordinated at well-def ned stages by exchanging messages. Also, nodes performing tomography are heterogenous, and failures are independent. There is no single *tomographing-node* that has the knowledge of the entire state of

the system, but the workload is divided and refined information is exchanged among the *tomographing-nodes*.

The following is the rationale behind distributed network tomography [51, 52].

1. Network Scalability: One main challenge for network tomography systems is the fact that probe packets are injected into the network. When the network is measured from different end nodes the load on individual links can unintentionally become high. Reusing the probe packets for sending application data according to the proposals in [51] will reduce the overhead. However, in scenarios where an operator aims at pure monitoring, the probe-packet overhead may still be an issue.

2. Node Scalability: Available capacity measurements as well as measurements of the other time-stamp dependent parameters such as one-way delay require high precision time stamps on sent and received probe packets. Increasing the number of simultaneous measurements escalates the risk of biased time stamps caused by probe packets originating from the other nodes. Thus, parameter estimation results may be affected by other concurrent measurement sessions even though they are not measuring the same path.

3. Administration, control and usage: For all distributed systems the overhead in terms of administration and control should be limited to reduce unnecessary network traffic. This is especially true when designing network tomography systems where the participating measurement nodes load the network not only with administrative traffic but also with probe traffic. The traffic for administration, control and usage in a network tomography system includes control traffic for scheduling measurements, traffic for replicating and storing measurement data as well as the traffic generated when a user is searching for specific data. Further, the manual labor required to administrate and manage a monitoring system must be kept to a minimum.

Network tomography has always considered operation centered at a single node, no matter what type of network tomography is applied. This introduces all the disadvantages of centralized computing and takes away advantages of the distributed computing. This was the research motivation behind the novel concept of the blind distributed network tomography in this chapter. The rest of the chapter is organized as follows. Section 6.2 presents related work. Section 6.3 explains the concept of distributed network tomography. Section

6.4 presents and discusses simulation results to show the successful implementation of the distributed network tomography without *a priori* knowledge of the routing matrix. Section 6.5 presents a chapter summary.

6.2 Related Work

In [52] a self-organizing, decentralized, and scalable network tomography control system is described with the claim that this system autonomously and continuously measures network performance without central control. This system of network tomography is capable of triggering active measurements. There is no central control system; rather, each node is equipped with an autonomous management protocol that decides how to schedule measurements with other nodes, how to treat joining and leaving the network as well as where to store measurement results.

The network tomography system contains three types of node roles; bootstrap, master, and slave. The master nodes are responsible for triggering measurements among selected peer nodes, which can be either another master node or a slave node. The slave nodes participate in measurements but cannot trigger new measurements. Each node is at a given point in time either a master or a slave. The roles are periodically re-assessed.

For managing switching between master and slave roles, the master-to-slave ratio is a configurable system parameter. It is part of the control protocol and determines the probability for a node to become a master or a slave when re-assessing its role. This property gives the network operator a parameter for controlling how many concurrent measurements are ongoing at a specific point in time.

The first node in the system is called the bootstrap node. It has to be started manually by the administrator of the system. The other nodes must be configured with both IP address and port numbers of the bootstrap node. When a node joins, it receives a list of the other participating nodes, and from this list the measurement peers can be selected.

The version of the network tomography system, presented in [52], uses distributed data storage. In association with the measurement between two nodes, the results are stored at both the master and the corresponding slave. To obtain measurement results, the operator queries the node that is responsible for a specific measurement. This storage scheme provides distributed and replicated storage.

In [52], a distributed system for selecting which paths to measure in large-scale networks

is also described. The method basically selects a set of nodes to perform measurements. If the number of measurement nodes is large enough, the probability of covering enough links for inferential analysis is shown to be high. A distributed data management system utilizing distributed hash tables is also proposed.

The distributed system described in [52] is very complicated and needs a lot of communications overhead for learning the states of various types of nodes such as bootstrap, master, and slave nodes. These different types of nodes require to communicate with each other for changes in role and prioritization. For example, all the nodes need to contact a bootstrap node to get registered with that bootstrap node. The framework presented in [52] uses a distributed data management system that has its own communication requirements and is communication and calculation intensive as it needs to run algorithms to deal with coding and decoding of the hash tables.

As compared to the framework of distributed network tomography in [52], the proposed mechanism in this chapter is simple and efficient as there are no multiple types of nodes. There is no requirement of frequent communication among nodes for coordination and the computation intensity is effectively distributed as described in the following section.

6.3 Distributed Network Tomography Steps

To get the advantages of the distributed system, the following blind method for network tomography is proposed that consists of three steps.

1. Tomography at multiple nodes: Independently, the tomography is performed at a selected number of *tomographing-nodes* (N) in the network. Any kind of network tomography (such as passive, active, and topology identification) can be performed independently at these selected nodes. The sectional results consisting of the inferred parameters that cover various segments (N) of the network are available for exchange among *tomographing-nodes*.

2. Exchange of network tomographic data: As the selected nodes communicate through a routing protocol, OSPF, the results of the distributed data repositories at these N *tomographing-nodes* are mutually communicated by using a modified version of OSFP updates with link-local signaling (LLS).

3. Aggregating results: After the mutual exchange, each *tomographing-node* appends all the results received from the other contemporary *tomographing-nodes* to its own results. At the end, each *tomographing-node* is aware of the net estimated statistics for the entire network that was the objective of a distributed network tomography.

6.4 Simulation Results

On a relatively larger test bed, as described in the following subsection and Figure 6.1, distributed network tomography was implemented. Network tomography was carried out at four distinct locations on various portions of the test bed by using the blind network tomography with NNMF that was introduced in Chapter 3. The test bed uses OSPF for its routing operation. A modif ed OSPF with LLS was applied to enable OSPF to carry out the exchange of estimated data obtained from distributed tomography.
Link delays were estimated at each of the four components from end to end (path) delays at each of the four smaller portions of the large test bed.
The simulation process consists of the following steps.

1. The link delay matrices at four *tomographing-nodes* in the four section of the test bed were estimated from the data obtained from the test bed by using the network tomography technique described in Chapter 3.

2. The exchange of the estimated delay matrices among the four selected nodes (*tomographing-nodes*) were implemented in NS-2 by implementing the modif ed OSPF with LLS.

The next subsections describe the test bed used to generate the data for simulations and implementation details of the above two simulation steps.

6.4.1 Description of Test Bed

The data for simulations was obtained from a test bed in the Advanced Internetworking Laboratory (AIL) at Dalhousie University that consists of 3600 and 3800 series Cisco routers. The test bed includes 24 routers as shown in Figure 6.1. The topology in Figure 6.1 is made of four identical sub topology blocks and the sub topology is shown in Figure 6.2 to illustrate the details of the traff c and devices.
In Figure 6.2, the Path Echo option of the Cisco CSLA was implemented to send four

probes to collect the cumulative RTT from source to each hop. All probes were grouped together. In each of the four subsections (Figure 6.2) of the test bed, all the four probes in the group started at the same time and travel from right to left.

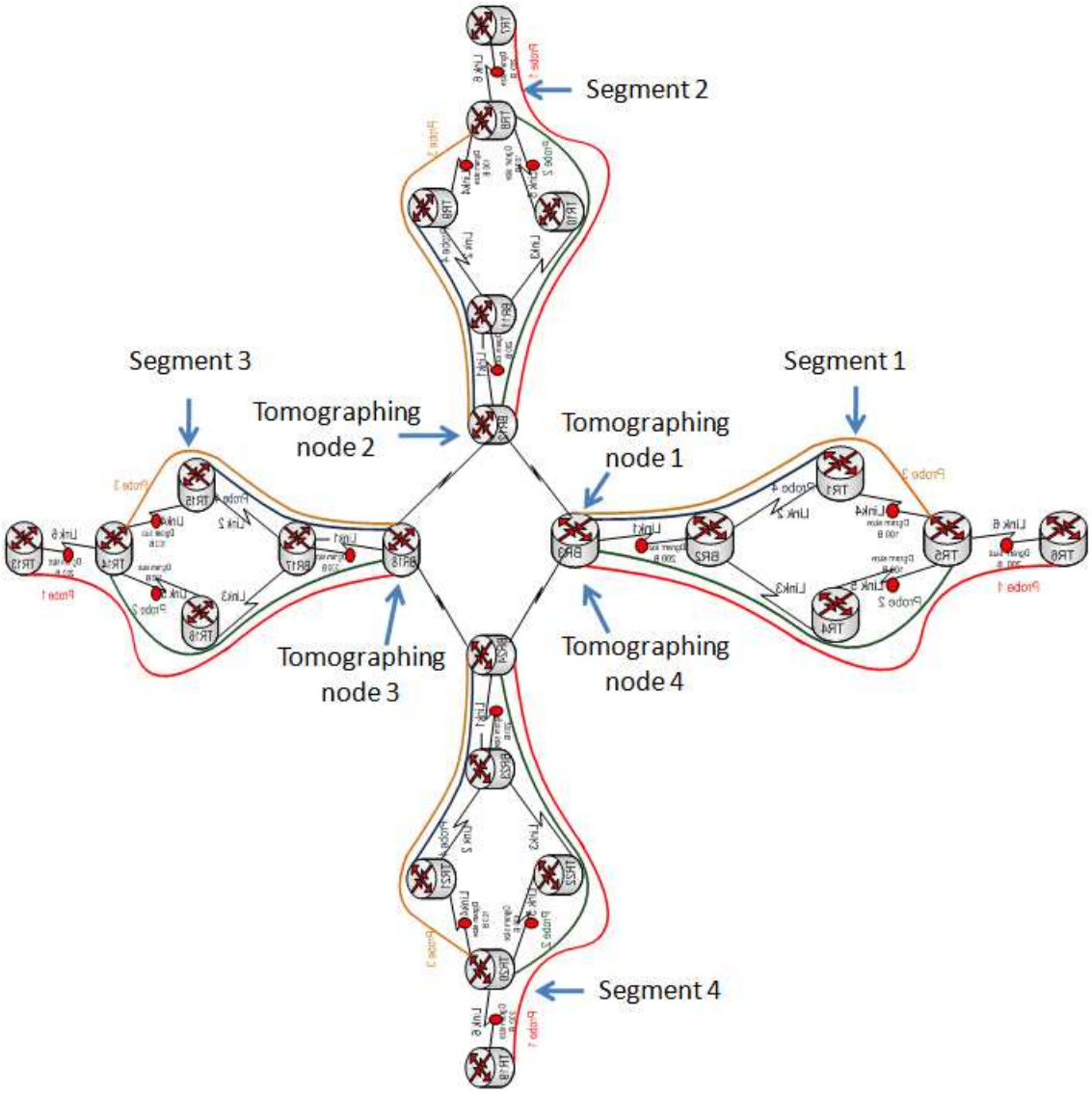

Figure 6.1: Test bed setup for distributed network tomography

6.4.2 Processing of Collected Data

The data obtained from the CSLA was in the form of accumulative hop-wise RTT; the steps described in Subsection 3.5.2 were followed to process the data for obtaining two

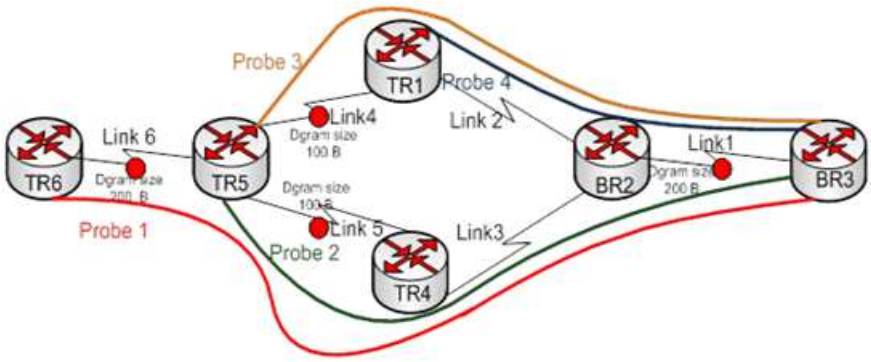

Figure 6.2: Detail of a component of the Test bed Setup for distributed network tomography

matrices; a matrix of end-to-end (path) delays and a matrix of link level delays. To find the correlation coefficient, matching rows in two matrices (estimated and measured) were found and their correlation was determined. As a result, a column vector of correlation coefficients between the best-correlating rows of matrices (estimated and measured) was obtained.

6.4.3 Exchange of Data Among *Tomographing-Nodes*

The test bed operates on OSPF. The format of OSPF [RFC2328] packets does not support flexible data transfer. An example where such a technique could be used is exchanging some capabilities on a link: standard OSPF utilizes the Options field in Hello and Exchange packets such as in the case of the exchange of estimated network tomography data among nodes performing network tomography, but there are not so many bits left in it. One potential way of solving this problem could be introducing a new packet type. However, that would mean introducing extra packets on the network, which may not be desirable. Thus, this section describes (based on [RFC 4813] how to exchange data using existing standard OSPF packet types.

The mechanism for implementing LLS is described below. To perform LLS, OSPF routers add a special data block at the end of OSPF packets or right after the authentication data block when cryptographic authentication is used. Like OSPF cryptographic authentication, the length of the LLS block is not included into the length of the OSPF packet, but is included in the IP packet length. Figure 6.3 illustrates how the LLS data block is attached.

The LLS data block may be attached to OSPF packets of two types – type 1 (OSPF Hello),

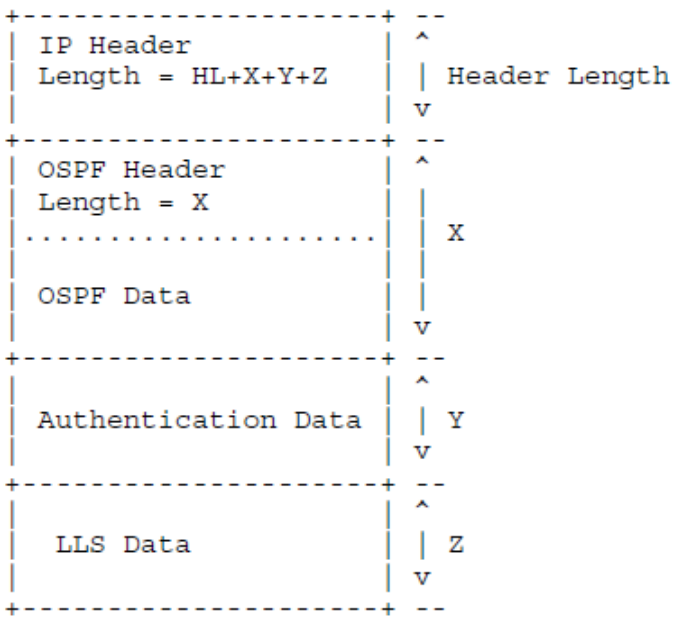

Figure 6.3: LLS data block in OSPF version 2 [RFC 4813]

and type 2 (OSPF Database Description (DBD)). The data included in the LLS block attached to a Hello packet may be used for dynamic signaling, since Hello packets may be sent at any moment in time. However, delivery of LLS data in Hello packets is not guaranteed. The data sent with DBD packets is guaranteed to be delivered as part of the adjacency forming process. This option has been adopted for the simulation in this chapter.

Some key bits in the LLS packet structure are defined below: Options field: A new bit, called L (L stands for LLS), is introduced to the OSPF Options field (see Figure 6.4). The value of the bit is 0x10. Routers set the L-bit in Hello and DBD packets to indicate that the packet contains the LLS data block.

L-bit: this bit is set only in Hello and DBD packets. It is not set in OSPF Link State

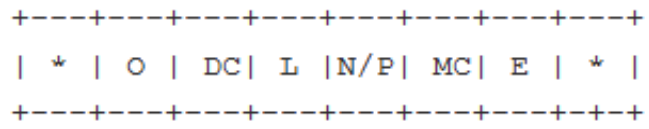

Figure 6.4: OSPF version 2 options field [RFC4813]

Advertisements (LSAs) and may be used in them for different purposes.

LLS data block: the data block used for link-local signaling is formatted as described below (see Figure 6.5 for illustration).

Checksum: the checksum f eld contains the standard IP checksum of the entire contents

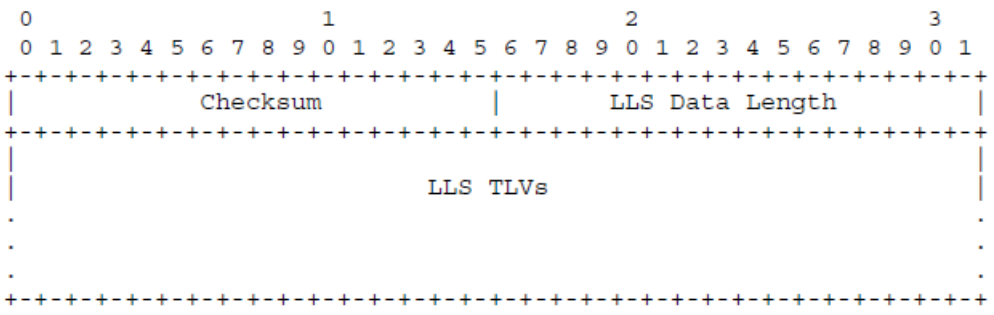

Figure 6.5: Format of LLS Data Block [RFC4813]

of the LLS block.

LLS length: the 16-bit LLS data length f eld contains the length (in 32-bit words) of the LLS block including the header and payload. Implementations should not use the Length f eld in the IP packet header to determine the length of the LLS data block. All TLVs (Type/Length/Value) must be 32-bit aligned (with padding if necessary).

LLS TLVs: The contents of the LLS data block is constructed using TLVs. See Figure 6.6 for the TLV format.

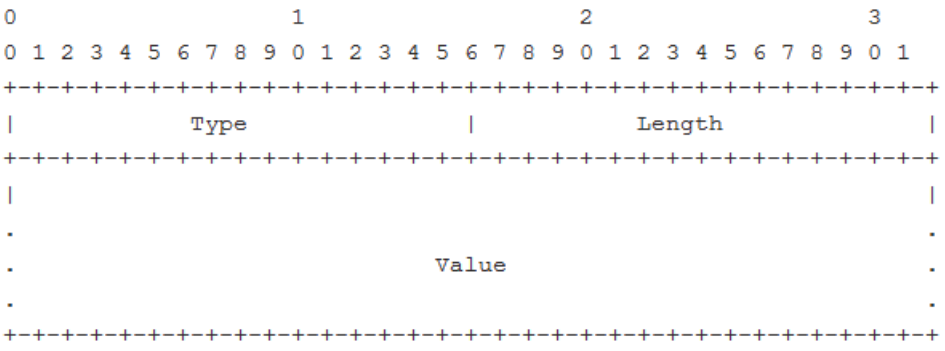

Figure 6.6: Format of LLS TLVs [RFC4813]

The Type f eld contains the TLV ID that is unique for each type of TLVs. The Length f eld contains the length of the Value f eld (in bytes) that is variable and contains arbitrary data. Note that TLVs are always padded to the 32-bit boundary, but padding bytes are not

included in the TLV Length f eld (though it is included in the LLS Data Length f eld of the LLS block header).

Extended Options (EO)TLV: Bits in the Value f eld do not have any semantics from the point of view of the LLS mechanism. This f eld may be used to announce some OSPF capabilities that are link-specif c. Also, other OSPF extensions may allocate bits in the bit vector to perform boolean link-local signaling. The length of the Value f eld in EO-TLV is 4 bytes. The value of the Type f eld in EO-TLV is 1. EO-TLV should only appear once in the LLS data block.

Currently, [RFC4811] and [RFC4812] for LLS use bits in the Extended Options f eld of the EO-TLV.

By default NS-2 does not include OSPF as one of the routing protocols. A protocol equivalent of OSPF that is called Partial Link State implementation, and named as "rtProto" protocol, is used in NS-2 as one of the routing protocols. Therefore the existing "rtProto" protocols has been modif ed (as given in Appendix B) to include the packet formats described in this section.

6.4.4 Interpretation of Results

Two categories of results are shown; one category of results shows the correlation between the estimated link delay in all the four sections of the test bed and the second category of the results show the correlation between the estimated and the actual link delays for the entire test bed after the exchange of data among the four sections of the test bed. The results were expected to show a strong correlation between the estimated and true link delays in both cases.

Results from the Segments of the Test Bed:

The NNMF based blind network tomography method described in Chapter 3 was applied on all the four segments. The correlation among the estimated and the measured link delays is shown in Figure 6.6 to Figure 6.9 for segment 1 to segment 4 respectively.

Combined Results for the Entire Test Bed and Discussion on Results

Figure 6.11 shows the result for the entire test bed. Network tomography data from the four segments was exchanged among *tomographing-nodes* as per Subsection 6.4.3. This

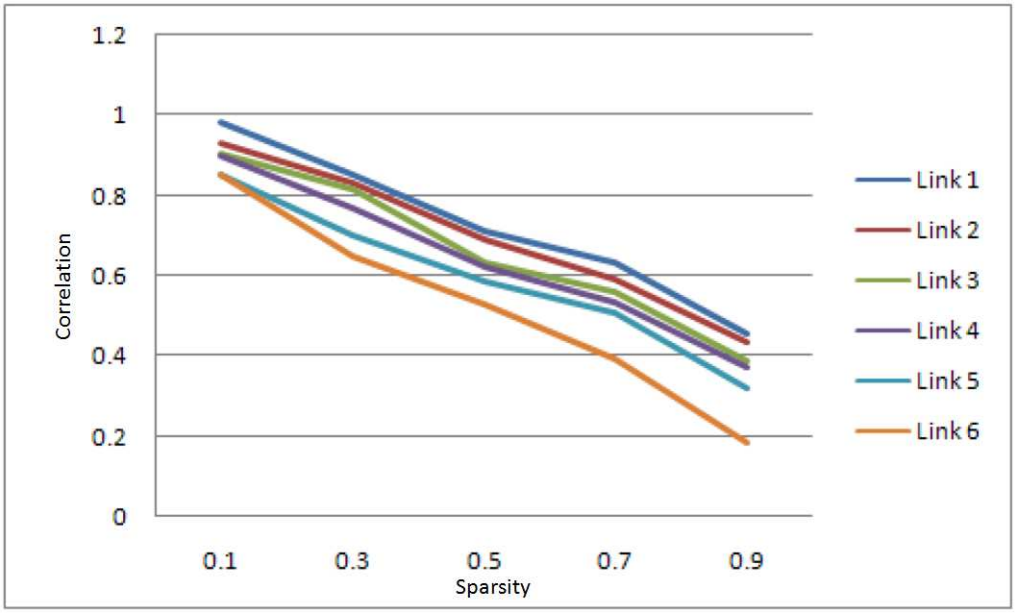

Figure 6.7: Correlation between H and X with extended pings for Segment 1 on the right side

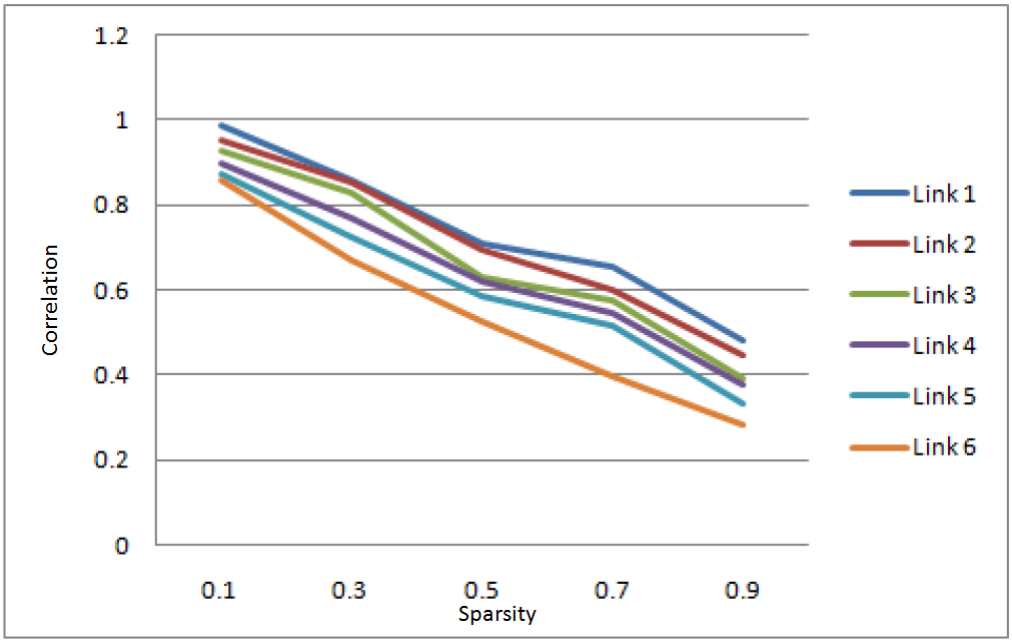

Figure 6.8: Correlation between H and X with extended pings for Segment 2 on the top

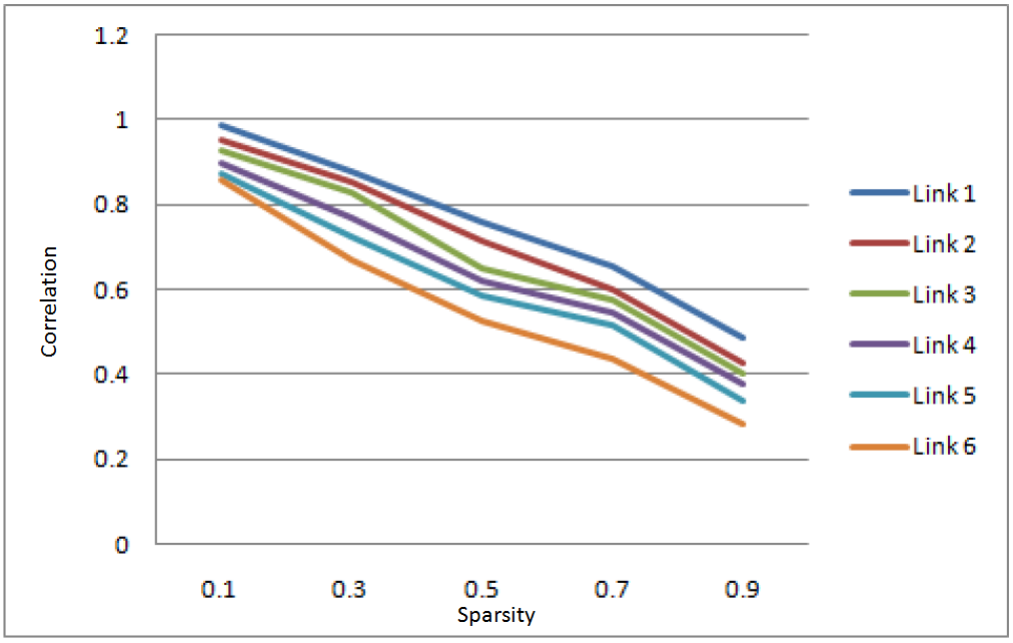

Figure 6.9: Correlation between H and X with extended pings for Segment 3 on the left side

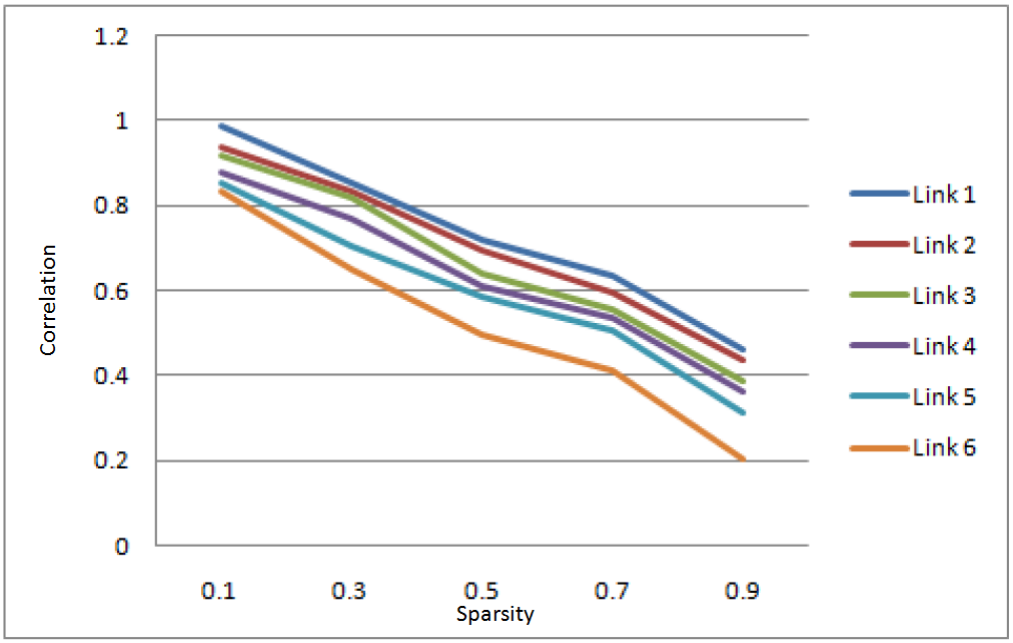

Figure 6.10: Correlation between H and X with extended pings for Segment 4 on the bottom

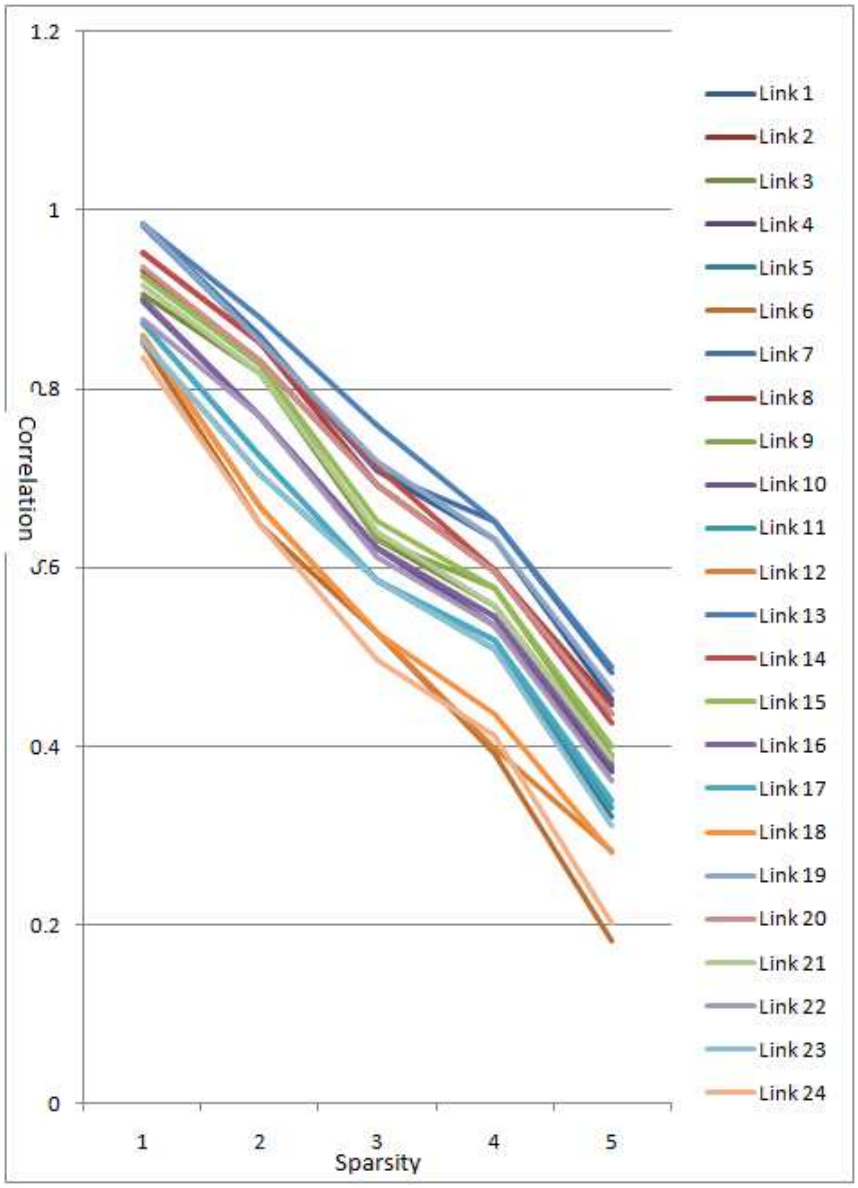

Figure 6.11: Correlation between H and X with extended pings for all segments combined

shows the correlation of the estimated and the true link delays for all the 24 links in the system.

In these simulations, the link delays were estimated for the entire network, but with shorter probes. If the link delays were to be estimated without distributed arrangements, a number of full length probes from a single *tomographing-nodes* to various destinations at the other ends of the network would be required to cover all the links in the entire network. This would increase traff c due to many probes with each probe covering more hops. Also, when the network is measured from different end nodes in central network tomography the load on individual links can unintentionally become high. Therefore, the number of probes required to do the job of estimating link delays could have been much higher to cover all the links in the entire network. Designing the probe paths and then ensuring the traversed route to be the same as per the design of the probe would have been extremely challenging. Whereas with the distributed scheme for carrying out network tomography, the number and distance of each probe were reduced to fewer hops. Therefore, the total volume of traff c for the four components of the localized traff c would be much less as compared to the probing throughout the entire network. The processing was done on the quarter of the data at each location simultaneously. The exchange of the estimated data from each quarter was done with the modif ed routing protocol; OSPF by simulating “rtProto” protocol in the NS-2 simulator.

Link delays were estimated for the whole test bed with the pings or probes of the smaller length and localized nature, because each *tomographing-node* has to do processing on limited data by covering only a segment of the entire test bed.

6.5 Chapter Summary

This chapter introduced a new mechanism for implementing distributed network tomography to ensures network and node scalability. This mechanism does not require any special protocol for implementation, rather by implementing LLS implementation of OSPF, the data exchange between the segments of a network was achieved. The link delays were estimated from the four segments of a test bed and these estimations were exchanged by using a variation of OSPF (LLS), and all the four *tomographing-node* had an overall estimate of the links in the whole test bed.

Chapter 7

Conclusions and Future Work

This chapter summarizes the contributions of this dissertation and gives possible directions for future work. Section 7.1 describes the summary of the contributions and Section 7.2 outlines the direction for future work.

7.1 Summary of Contributions

The research work in this dissertation makes four significant novel contributions to the field of network tomography. These research contributions were focused on the blind techniques for performing network tomography, the modeling of errors in network tomography, improving estimates with multimetric-based network tomography, and distributed network tomography. All of these four research problems, related to network tomography, were solved by various blind techniques including NNMF, SCS, and NTF. These contributions have been verified by processing the data obtained from laboratory experiments and by examining the correlation between the estimated and measured link delays.

The first contribution presented the application of a blind sparse matrix-based technique to eliminate the unrealistic assumption of a known routing matrix in network tomography. The objective was to get an appropriate method for the inverse problem solution where the routing matrix was accommodated by the statistical ability of such methods as NNMF. NNMF (with the feature of sparsity) was used to factorize a matrix into two factors (matrices); link delays were estimated as a result. The path delay and the link delays were obtained from the data from a laboratory experiment. The path delays were used as an input to NNMF to estimate link delays. The estimated link delays were correlated with the measure link delays and the correlation was observed as close to 1.

The second contribution, a modified version of SCS was applied to denoise the network tomography model with errors by using NNMF in place of ICA. It was presumed that the errors in the measurement of various parameters for network tomography were introduced by various factors such as SNMP and NetFlow measurements. The estimated error free

link delays were compared with the original (error free) link delays based on the data obtained from a laboratory test bed. The simulation results revealed that the denoising of the tomography data had been carried out successfully by applying SCS.

The third contribution introduced a novel concept of multiple metric network tomography in contrast to the conventional network tomography where only one directly measured parameter is considered for the indirect estimation of another single parameter. By using two directly observed parameters (PLR and path delays), a single parameter (link delays) was inferred indirectly. Based on the evidence from literature that there is a correlation between link delays and PLR parameters, PLR and path delays were directly measured to estimate link delays indirectly. A variation of the non NTF, NTF1 model was applied for this purpose to estimate link delays from a duplex of metrics; PLR and path delays. Simulation results showed a better correlation between the estimated and measured link delays when a duplex of metrics was used as compared to using only the path level link delays for estimating the link delays on a test bed.

The fourth contribution introduced a novel technique for implementing distributed network tomography. In conventional tomography research, network tomography processing is carried out by a single node in a centralized manner. A distributed system of network tomography estimation was proposed where more than one node was responsible to carry out network tomography on a distributed pattern. The estimated parameters (link delays) were exchanged among multiple data repositories by implementing an LLS capable version of OSPF in the NS-2 network simulator. This gives the benef ts of distributed computing and avoids the disadvantages of centralized computing. The simulation results verif ed that the link delays from various sections of a test bed were estimated by the respective *tomographing-nodes*; these estimations were exchanged among the *tomographing-nodes*, and f nally these sectional estimations were appended at each *tomographing-node* to get the link delays of all the links in the entire test bed.

7.2 Future Research Directions

This section presents a summary of the research directions that will be pursued in the near future as a continuation of the research contributions in this dissertation. The following are a few potential research directions:

- Topology identification has been mentioned as the third type of network tomography but technically it is a subset of active network tomography. As a by-product of the NNMF-based tomography (the first contribution of this dissertation), the routing matrix is obtained along with the link delay matrix as one of the two estimated matrices. It would be interesting to investigate how to refine this estimation of the routing matrix with appropriate signal processing and statistical techniques. The estimated routing matrix by NNMF is in fractions as compared to the original routing table where entries are in the form of 1's and 0's. Therefore, statistical techniques will be explored to convert the estimated fraction from NNMF to 1's and 0's.

- Error modeling has been covered by this dissertation in a unified manner by considering all the errors under one heading, but these errors can also be categorized separately and separate models can be created to represent them. Then both kinds (a single model representing all errors and specified models for various types of errors) can be compared for optimized performance levels. Also, more types of errors other than WGN can be inducted and if needed different models can be devised for dealing with these errors. Finally, the effectiveness of the error removal capability of the proposed models can be tested for various types of errors.

- The contributions of this dissertation were tested for active tomography; as a future extension, these contributions can be implemented by using the data obtained from passive network tomography.

- This dissertation has tested the developed techniques on the data obtained from the laboratory test beds. For obvious reasons, the sizes of these test beds were limited; therefore it would be interesting to implement the techniques presented in this dissertations on the data obtained from some real life networks. This would require more work while parsing the data and some efficient bulk data management and storing techniques will have to be explored.

- One of the many ways to categorize network tomography is how the active network tomography is being implemented; namely in single source and multiple sources active networks. This dissertation deals with the single source active network tomography only. It is promising to consider multiple source active network tomography to

implement the techniques presented in this dissertation. This future work will need to develop a framework to cover issues such as how to utilize the data from multiple sources, whether for verif cation or for unif cation. If the purpose of the framework is unif cation then some kinds of data integration protocols might need to be designed. Implementing and dealing with the multiple sources sending multiple probes to perhaps a single destination would be another challenge. Examining and f ltering the data to f nd out the ways to ref ne the data is another future research challenge.

- Multi-metric tomography in the context of the error model of network tomography can be investigated by using some mathematical models that are capable of denoising an estimated matrix in a multi-metric environment, for example, an NTF model with denoising capability.

Bibliography

[1] M. Raza, B. Robertson, W. Phillips and J. Ilow, "Network calculus based modeling of anomaly," in *Proc. IEEE SPECTS*, Jul. 2010, pp. 323-336.

[2] R. Castro, M. Coates, G. Liang, R. Nowak and B. Yu, "Network tomography: recent developments," *J. Statistical Science*, vol. 19, pp. 499-517, 2004.

[3] R. Castro, M. Coates, G. Liang, R. Nowak and B. Yu, "Internet Tomography," *IEEE Signal Processing Magazine*, vol. 19, pp. 45-65, 2002.

[4] E. Lawrence, G. Michailidis, V. Nair and B. Xi, "Network tomography: a review and recent developments," *J. Ann Arbor*, vol. 1001, pp. 1087-1107, 2006.

[5] M. Slaney and A. Kak. *Principles of computerized tomographic imaging*. IEEE Press, 1988.

[6] H. Nguyen and P. Thiran, "Active measurement for multiple link failures diagnosis in IP networks," *J. Passive and Active Network Measurement*, pp. 185-194, 2004.

[7] Y. Vardi, "Network tomography: estimating source-destination traff c intensities from link data," *J. of the American Statistical Association*, vol. 91, pp. 365-377, 1996.

[8] M. Rabbat, R. Nowak and M. Coates, "Multiple source, multiple destination network tomography," in *Proc. IEEE INFOCOM*, 2004, vol. 3, pp. 1628-1639.

[9] Y. Zhang, Z. Ge, Z, A. Greenberg and M. Roughan, "Network anomography," in *Proc. ACM/USENIX IMC*, 2005.

[10] S. Moon, J. Kurose, P. Skelly and D. Towsley, "Correlation of packet delay and loss in the Internet,", Univ. Massachusetts, Amherst, MA, Tech. Rep. 1998.

[11] A. Benveniste, M. Goursat and G. Ruget, "Robust identif cation of a nonminimum phase system: blind adjustment of a linear equalizer in data communications," *IEEE Trans. Automatic Control*, vol. 25, pp. 385-399, 2002.

[12] D. Godard, "Self-recovering equalization and carrier tracking in two-dimensional data communication systems," *IEEE Trans. Commun.*, vol. 28, pp. 1867-1875, 2002.

[13] F. Alayyan, R. Shubair, A. Zoubir and Y. Leung,"Blind channel identif cation and equalisation in OFDM using subspace-based methods," *J. commun.*, vol. 4, pp. 472-484, 2009.

[14] E. Moulines, P. Duhamel, J. Cardoso and S. Mayrargue, "Subspace methods for the blind identif cation of multichannel FIR f lters," *IEEE Trans. Signal Processing*, vol. 43, pp. 516-525, 2002. .

[15] M. Raza, B. Robertson, W. Phillips and J. Ilow, “Network tomography by non negative matrix factorization (NNMF),” in *Proc. IEEE SPECTS*, Jul. 2010, pp. 58-64.

[16] M. Raza, B. Robertson, W. Phillips and J. Ilow, “Denoising network tomography estimation,” in *Proc. IEEE DCNET*, Jul. 2010.

[17] M. Raza, B. Robertson, W. Phillips and J. Ilow, “Error modeling in network tomography by sparse code shrinkage (SCS) method,” in *Proc. IEEE Globecom*, Dec. 2010. .

[18] M. Raza, B. Robertson, W. Phillips and J. Ilow, “Multimetric network tomography,” in *Proc. IEEE DCNET*, Jul. 2010.

[19] A. Medina, N. Taft, K. Salamatian, S. Bhattacharyya and C. Diot, “Traff c matrix estimation: existing techniques and new directions,” *ACM SIGCOMM CCR*, vol. 32, pp. 161-174, 2002.

[20] Y. Zhang, M. Roughan, N. Duff eld and A. Greenberg, “Fast accurate computation of large-scale IP traff c matrices from link loads,” *ACM SIGMETRICS PER*, vol. 31, pp. 206-217, 2003.

[21] M. Coates, R. Castro, R. Nowak, M. Gadhiok, R. King and Y. Tsang, “Maximum likelihood network topology identif cation from edge-based unicast measurements,” *ACM SIGMETRICS PER*, vol. 30, pp. 11-20, 2002.

[22] M. Coates and R. Nowak, “Network tomography for internal delay estimation,” in *Proc. ICASSP*, 2001, vol. 6, pp. 3409-3412.

[23] V. Padmanabhan, L. Qiu and H. Wang, “Passive network tomography using bayesian inference,” in *Proc. ACM SIGCOMM*, 2002, pp. 93-94.

[24] C. Tebaldi and M. West, “Bayesian inference on network traff c using link count data,” *J. American Statistical Association*, vol. 93, pp. 557-573, 1998.

[25] Y. Zhang, M. Roughan, G. Lund and D. Donoho, “An information-theoretic approach to traff c matrix estimation,” in *Proc. ATACCC*, 2003, pp. 301-312.

[26] R. Caceres, N. Duff eld, J. Horowitz, D. Towlsey and T. Bu, “Multicast-based inference of network-internal characteristics: accuracy of packet loss estimation,” in *Proc. IEEE INFOCOM*, 1999, vol. 1, pp. 371-379.

[27] G. Liang, and B. Yu, “Maximum pseudo likelihood estimation in network tomography,” *IEEE Trans. Signal Processing*, vol. 51, pp. 2043-2053, 2003.

[28] N. Duff eld, F. Presti and D. Towsley, “Multicast-based inference of network-internal delay distributions,” *IEEE/ACM Trans. Networking*, vol. 10, pp. 761-775, 2002.

[29] Y. Tsang, M. Coates and R. Nowak, “Network delay tomography,” *IEEE Trans. Signal Processing*, vol. 51, pp. 2125-2136, 2003.

[30] M. Shih and A. Hero, "Unicast-based inference of network link delay distributions with f nite mixture models," *IEEE Trans. Signal Processing*, vol. 51, no. 51, pp. 2219-2228, 2003.

[31] N. Duff eld, J. Horowitz, F. Presti and D. Towsley, "Multicast topology inference from end-to-end measurements," *J. Advances in Performance Analysis*, vol. 3, pp. 207-226, 2000.

[32] J. Cao, D. Davis, S. Wiel, and B. Yu. J. Cao, D. Davis, S. Wiel and B. Yu, "Time-varying network tomography: router link data," *J. American Statistical Association*, vol. 95, no. 452, pp. 1063-1075, 2000 .

[33] E. Lawrence, G. Michailidis, and V. Nair, "Maximum likelihood estimation of internal network link delay distributions using multicast measurements," in *Proc. CISS*, Mar. 2003, pp. 330-336. .

[34] P. McCullagh and J. Nelder. *Generalized linear models*. Chapman & Hall/CRC, 1989.

[35] M. Coates, and R. Nowak, "Sequential Monte Carlo inference of internal delays innonstationary data networks," *IEEE Trans. on Signal Processing*, vol. 50, no. 2, pp. 366-376, 2002.

[36] D. Moore, C. Shannon, D. Brown, G. Voelker and S. Savage, "Cooperative association for Internet data analysis," *Physical Review Letters*, vol. 87, no. 25, 2008.

[37] N. Duff eld, J. Horowitz, F. Presti and D. Towsley, "Network delay tomography from end-to-end unicast measurements," *Lecture Notes in Computer Science*, pp. 576-595, 2001.

[38] B. Ratner. *Statistical modeling and analysis for database marketing: effective techniques for mining big data*. CRC Press, 2003.

[39] E. Oja, A. Hyvarinen, and J. Karhunen. *Independent Component Analysis*, John Wiley & Sons, 2001.

[40] P. Hoyer, "Non-negative matrix factorization with sparseness constraints," *J. Machine Learning Research*, vol. 5, pp. 1457-1469, 2004.

[41] A. Cichocki, R. Zdunek, A. Phan, and S. Amari. *Nonnegative Matrix and Tensor Factorizations: Applications to Exploratory Multi-way Data Analysis and Blind Source Separation*. Wiley, 2009.

[42] A. Delorme and S. Makeig, "EEGLAB: an open source toolbox for analysis of single-trial EEG dynamics including independent component analysis," *J. Neuroscience Methods*, vol. 134, no. 1, pp. 9-21, 2004.

[43] A. Clemm. *NetFlow Services Solutions Guide*, Cisco Press, 2006.

[44] A. Clemm. *Network Management Fundamentals*, Cisco Press, 2006.

[45] M. Nagaraja, R. Chittal and K. Kumar, “Study of network performance monitoring tools-SNMP,” *IJCSNS* vol. 7, no.7, pp. 310-314, Jul. 2007.

[46] Q. Zhao, Z. Ge, J. Wang and J. Xu, “Robust traff c matrix estimation with imperfect information: making use of multiple data sources,” *ACM SIGMETRICS PER*, vol. 34, no. 1, pp. 133-144, 2006.

[47] A. Rinen, “Sparse code shrinkage: denoising of nongaussian data by maximum likelihood estimation,” *J. Neural Computation*, vol. 11, no. 7, pp. 1739-1768, 1999.

[48] S. Bhamidi, R. Rajagopal and S. Roch, “Network delay inference from additive metrics,” *J. Random Structures Algorithms*, vol. 37, no. 2, 2006.

[49] J. Dollimore, T. Kindberg, and G. Coulouris. *Distributed Systems: concepts and design*, Addison Wesley, May, 2005.

[50] A. Tanenbaum and M. Van Steen. *Distributed Systems: Principles and Paradigms*. Pearson Prentice Hall, 2007.

[51] P. Papageorge, J. McCann and M. Hicks, “Passive aggressive measurement with MGRP,” *ACM SIGCOMM CCR*, vol. 39, no. 4, pp. 279-290, 2009.

[52] R. Hoque, A. Johnsson, C. Flinta, M. Bjrkman and S. Ekelin, “A self-organizing scalable network tomography control protocol for active measurement methods,” in *Proc. SPECTS*, Jul. 2010, pp. 80-87.

Appendix A

Non Negative Matrix Factorization (NNMF)

In this dissertation, the following adaptation of NNMF is implemented to meet sparsity requirements. The explanation of the algorithmic steps is presented below:

1. User def nes the dimensions of matrices V (for example; 6 $\times$ 100 in Chapter 3 of this dissertation), W (for example; 6 $\times$ 10 in Chapter 3 of this dissertation), H (for example; 10 $\times$ 100 in Chapter 3 of this dissertation), and the value of the system components r (for example; 10 in Chapter 3 of this dissertation)

2. Initialize W, H to random positive matrices

3. Cost Function: The cost function applied was based on divergence,

$$D(A\|B) = \sum_{ij}(A_{ij}\log\frac{CA_{ij}}{B_{ij}} - A_{ij} + B_{ij}) \tag{A.1}$$

   A cost function is used for the formulation of the NNMF optimization problem to minimize $D(V\|WH)$ with respect to W and H, subject to the constraints W, $H \geq 0$.

4. Update Rule: For this cost function, there are rules for updating W and H after selecting initial values of W and H. At each iteration W and H are multiplied and $D(V\|WH)$ is calculated. The values of W and H are updated until $D(V\|WH)$ reaches a minimum threshold. At this moment, the values of W and H represent the f nal estimate.

   A projected gradient descent algorithm for NNMF with sparseness constraints takes a step in the direction of the negative gradient, and subsequently projects onto the constraint space, making sure that the taken step is small enough that the objective function is reduced at every step.

   In the following algorithm, $\otimes$ and $\oslash$ denote element-wise multiplication and division, respectively. Moreover, μ_w and μ_H are small positive constants (step-sizes).

Algorithm 1 NNMF with sparsity constraint based on a sparse-parameter

1: *Step # 1*: Initialize W, H to random positive matrices
2: *Step # 2*: If constraints apply to W or H or both, project each column or row respectively to have unchanged L_2 norm and desired L_1 norm
3: *Step # 3*: If sparseness constraints on H apply, then project each row of H to be non-negative, have unit L_2 norm, and L_1 norm set to achieve desired sparse-parameter
4: **for** Iterate **do**
5: **if** sparseness constraints on W apply **then**
6: $W := W - \mu_w(WH - A)H^T$
7: Project each column of W to be non-negative, have unchanged L_2 norm, but L_1 norm set to achieve desired sparseness
8: **else**
9: take standard multiplicative step $W := W \otimes (VH^T) \oslash (WHH^T)$
10: **end if**
11: **if** sparseness constraints on H apply **then**
12: $H := H - \mu_H W^T(WH - A)$
13: Project each row of H to be non-negative, have unit L_2 norm, and L_1 norm set to achieve desired sparse-parameter
14: **else**
15: take standard multiplicative step $H := H \otimes (W^T V) \oslash (W^T WH)$
16: **end if**
17: **end for**

5. **Sparsity Parameter:** The main strength of the above algorithm is the projection operator which enforces the desired degree of sparseness. This parameter is described in detail in the following part of this appendix. Sparsity for the routing matrix (represented by W in the following algorithm) is imposed on each row indicating that each probe going through a few nodes.

   Sparsity for the link delay matrix (represented by H in the following algorithm) is imposed on each column indicating the simultaneous bottlenecks causing the delay and x is a row vector of H used in NNMF. The sparsity for link delays is imposed by the following function that indicates that we can control a vectors sparsity by manipulating its L_1 and L_2 norms by using the following function [40],

$$sparsness(x) = \frac{\sqrt{(T) - \frac{\|x\|_1}{\|x\|_2}}}{\sqrt{(T) - 1}}, \tag{A.2}$$

where T is the dimensionality of x.

Many of the steps in the above algorithm require a projection operator which enforces sparseness by explicitly setting both L_1 and L_2 norms (and enforcing non-negativity). This operator is def ned as follows.

Given any vector x, f nd the closest non-negative vector s with a given L_1 norm and a given L_2 norm.

Algorithm 2 Sparsity with a given L_1 norm and a given L_2 norm

1: *Step # 1*: Set $s_i := x_i + (L_1 - \sum x_i)/dim(X)$

2: *Step # 2*: Set Z := { }

3: *Step # 3*: Iterate

- Set$m_i := L_1/(dim(X) - size(Z))$ if i is not member of Z and 0 if i is member of Z
- Set $s := \mathbf{m} + \alpha(s - \mathbf{m})$, where $\alpha \geq 0$ is selected such that the resulting s satisf es the L_2 norm constraint. This requires solving a quadratic equation.
- If all components of s are non-negative, return s, end
- Set $Z := Z \bigcup \{i; s_i < 0\}$
- Set $s_i := 0$
- Calculate $c := (\sum x_i - L_1)/(dim(X) - size(Z))$
- Set $s_i := s_i - c$
- Repeat the steps under iterate

The above algorithm works as follows: We start by projecting the given vector onto the hyperplane $\sum x_i = L_1$. Next, within this space, we project to the closest point on the joint constraint hypersphere (intersection of the sum and the L_2 constraints). This is done by moving radially outward from the center of the sphere (the center is given by the point where all components have equal values). If the result is completely non-negative, we have arrived at our destination. If not, those components that attained negative values must be f xed at zero, and a new point found in a similar fashion under those additional constraints.

Appendix B

OSPF Link-local Signaling Implementation in NS-2.34

Open Shortest Path First (OSPF), an intra-domain routing protocol, is implemented in NS-2.34. OSPF packet is not flexible enough to enable applications exchanging data for situations such as link-local signaling. RFC 4813 proposes Link-Local Signaling (LLS) by adding a special data block at the end of the OSPF packet. NS-2.34 does not include the OSPF as it only provides the partial Link state implementation in rtProto protocol. Therefore, the existing rtProto protocol has been modified to include the header of OSPF by adding the data field along with other fields as mentioned in the RFC 4813 and RFC 5613, and as described in Section 6.4.3 of this dissertation. A scenario with four *tomographing-nodes* has been simulated as per Figure 6.1 to exchanging data using modified OSPF by adding a data block.
The programs used in this dissertation are modified versions of the following NS-2.34 source files to incorporate the OSPF-LLS.

- Common/packet.h
- hdr-ls.h
- rtProtoLS.cc
- rtProtoLS.h
- tcl/lib/ns-default.tcl

In the following subsections, listing of these program (used in this dissertation) is provided, with the Section B.1 showing additional data structure used in this dissertation and Section B.6 describes the procedure to run the program.

B.1 Common/packet.h

Incorporated rtProtoLS header NS-2.34, specially adding following code in common/packet.h

```
#ifdef NSLS
#define PT_RTPROTO_LS  18

 "rtProtoDV", "CtrMcast_Encap", "CtrMcast_Decap", "SRM" ,
 "ntype" , "rtProtoLS"
#endif /* ifdef NSLS */

#ifndef NSLS
#define PT_NAMES "tcp", "telnet", "cbr", "audio", "video", "ack",
"start", "stop", "prune", "graft", "message", "rtcp", "rtp",
#endif /* ifndef NSLS */
```

B.2 hdr-ls.h

```
struct hdr_LS {
        // metrics variable identifier
        u_int32_t mv_;
    int msgId_;
     // Adding new private variable for data
    int data_;

        u_int32_t& metricsVar() { return mv_; }
    int& msgId() { return msgId_; }
    // Property function for setting and retriving data value
    int& dataValue() { return data_; }

    // Header access methods required by PacketHeaderManager
    static int offset_;
    inline static int& offset() { return offset_; }
    inline static hdr_LS* access(const Packet* p) {
        return (hdr_LS*) p->access(offset_);
    }
};
```

B.3 rtProtoLS.cc

```
void rtProtoLS::sendpkt(ns_addr_t dst, u_int32_t mtvar,
u_int32_t size, int data)
{
    dst_ = dst;
    size_ = size;
        Packet* p = Agent::allocpkt();
    hdr_LS *rh = hdr_LS::access(p);
        rh->metricsVar() = mtvar;
    rh->dataValue() = data;

        target_->recv(p);
}

void rtProtoLS::recv(Packet* p, Handler*)
{
    hdr_LS* rh = hdr_LS::access(p);
    hdr_ip* ih = hdr_ip::access(p);
    // -- LS stuffs --
    if (LS_ready_ || (rh->metricsVar() == LS_BIG_NUMBER))
        receiveMessage(findPeerNodeId(ih->src()), rh->msgId());
    else
        Tcl::instance().evalf("%s recv-update %d %d", name(),
                    ih->saddr(), rh->metricsVar());

     // is the value which is being sent via send()
     rh->dataValue();
    Packet::free(p);
}
```

B.4 rtProtoLS.h

Modif ed sendpkt function prototype

```
void sendpkt(ns_addr_t dst, u_int32_t z,
```

```
u_int32_t mtvar, int data);
```

B.5 tcl/lib/ns-default.tcl

Add following lines in tcl/lib/ns-default.tcl

```
Agent/rtProto/LS set preference_        120
Agent/rtProto/LS set INFINITY           [Agent set ttl_]
Agent/rtProto/LS set advertInterval      2
```

B.6 procedure

Replace the DV to LS in tcl/ex/simple-eqp1.tcl f le so to use rtProto/LS Agent. Copy the attached hdr-ls.h, rtProtoLS.cc, rtProtoLS.h in linkstate folder in NS-2.34. Recompile NS to include the changes in above source f les by writing 'make' in NS folder. It will compile the whole NS and object f les of the modif ed source f les will be created. Now, OPSF-LLS can be used to exchange arbitrary data

B.7 Complete source f les including the modif ed/added functions

B.7.1 rtProtoLS.cc

```
#include "config.h"
#ifdef HAVE_STL

#include "hdr-ls.h"
#include "rtProtoLS.h"
#include "agent.h"
#include "string.h" // for strtok

// Helper classes
class LsTokenList : public LsList<char *> {
public:
LsTokenList (char * str, const char * delim )
: LsList<char*> () {
for ( char * token = strtok (str, delim);
```

```
      token != NULL; token = strtok(NULL, delim) ) {
push_back (token);
}
}
};

class LsIntList : public LsList<int> {
public:
LsIntList (const char * str, const char * delim)
: LsList<int> () {
for ( char * token = strtok ((char *)str, delim);
      token != NULL; token = strtok(NULL, delim) ) {
push_back ( atoi(token) );
}
}
};

static class rtProtoLSclass : public TclClass {
public:
rtProtoLSclass() : TclClass("Agent/rtProto/LS") {}
TclObject* create(int, const char*const*) {
return (new rtProtoLS);
}
} class_rtProtoLS;


int rtProtoLS::command(int argc, const char*const* argv)
{
        if (strcmp(argv[1], "send-update") == 0) {
                ns_addr_t dst;
dst.addr_ = atoi(argv[2]);
dst.port_ = atoi(argv[3]);
                u_int32_t mtvar = atoi(argv[4]);
u_int32_t size  = atoi(argv[5]);
```

```
                // Getting data value from the argument list
                sendpkt(dst, mtvar, size, atoi(argv[6]));
                return TCL_OK;
        }
/* --------------- LS specific --------------- */
if (strcmp(argv[1], "lookup") == 0) {
if (argc == 3) {
int dst = atoi(argv[2]);
lookup (dst);
/* use tcl.resultf () to return the lookup results */
return TCL_OK;
}
}
if (strcmp(argv[1], "initialize") == 0) {
initialize ();
return TCL_OK;
}
if (strcmp(argv[1], "setDelay" ) == 0 ) {
if ( argc == 4) {
int nbrId = atoi(argv[2]);
double delay = strtod ( argv[3], NULL);
setDelay(nbrId, delay);
return TCL_OK;
}
}
if (strcmp(argv[1], "setNodeNumber" ) == 0 ) {
if ( argc == 3 ) {
int number_of_nodes = atoi(argv[2]);
LsMessageCenter::instance().setNodeNumber(number_of_nodes);
}
return TCL_OK;
}
if (strcmp(argv[1], "computeRoutes") == 0) {
computeRoutes();
```

```
return TCL_OK;
}
if (strcmp(argv[1], "intfChanged") == 0) {
intfChanged();
return TCL_OK;
}
if (strcmp (argv[1], "send-buffered-messages") == 0) {
sendBufferedMessages();
return TCL_OK;
}
if (strcmp(argv[1], "sendUpdates") == 0) {
sendUpdates ();
return TCL_OK;
}
return Agent::command(argc, argv);
}

void rtProtoLS::sendpkt(ns_addr_t dst, u_int32_t mtvar,
                        u_int32_t size, int data)
{
dst_ = dst;
size_ = size;
        Packet* p = Agent::allocpkt();
hdr_LS *rh = hdr_LS::access(p);
        rh->metricsVar() = mtvar;
rh->dataValue() = data;  // Adding data to the packet

        target_->recv(p);
}

void rtProtoLS::recv(Packet* p, Handler*)
{
hdr_LS* rh = hdr_LS::access(p);
hdr_ip* ih = hdr_ip::access(p);
```

```
// -- LS stuffs --
if (LS_ready_ || (rh->metricsVar() == LS_BIG_NUMBER))
receiveMessage(findPeerNodeId(ih->src()), rh->msgId());
else
Tcl::instance().evalf("%s recv-update %d %d", name(),
      ih->saddr(), rh->metricsVar());

// rh->dataValue() is the value which is being sent via send()
Packet::free(p);
}

/* LS specific */
// implement tcl cmd's

/* -- findPeerNodeId -- */
int rtProtoLS::findPeerNodeId (ns_addr_t agentAddr)
{
// because the agentAddr is the value, not the key of the map
for (PeerAddrMap::iterator itr = peerAddrMap_.begin();
     itr != peerAddrMap_.end(); itr++) {
if ((*itr).second.isEqual (agentAddr)) {
return (*itr).first;
}
}
return LS_INVALID_NODE_ID;
}

void rtProtoLS::initialize() // init nodeState_ and routing_
{
Tcl & tcl = Tcl::instance();
// call tcl get-node-id, atoi, set nodeId
tcl.evalf("%s get-node-id", name());
const char * resultString = tcl.result();
nodeId_ = atoi(resultString);
```

```
// call tcl get-peers, strtok, set peerAddrMap, peerIdList;
tcl.evalf("%s get-peers", name());
resultString = tcl.result();

int nodeId, neighborId;
ns_addr_t peer;
ls_status_t status;
int cost;

// Tcl MUST return pairs of numbers
for ( LsIntList intList(resultString, " \t\n");
      !intList.empty(); ) {
nodeId = intList.front();
intList.pop_front();
// Agent.addr_
peer.addr_ = intList.front();
intList.pop_front();
peer.port_ = intList.front();
intList.pop_front();
peerAddrMap_.insert(nodeId, peer);
peerIdList_.push_back(nodeId);
}

// call tcl get-links-status, strtok, set linkStateList;
tcl.evalf("%s get-links-status", name());
resultString = tcl.result();
// cout << "get-links-status:\t" << resultString <<endl;
//Tcl MUST return triplets of numbers
for ( LsIntList intList2(resultString, " \t\n");
      !intList2.empty(); ) {
neighborId = intList2.front();
intList2.pop_front();
status = (ls_status_t) intList2.front();
```

```
intList2.pop_front();
cost = (int) intList2.front();
intList2.pop_front();
linkStateList_.push_back(LsLinkState(neighborId,status,cost));
}

// call tcl get-delay-estimates
tcl.evalf ("%s get-delay-estimates", name());

// call routing.init(this); and computeRoutes
routing_.init(this);
routing_.computeRoutes();
// debug
tcl.evalf("%s set LS_ready", name());
const char* token = strtok((char *)tcl.result(), " \t\n");
if (token == NULL)
LS_ready_ = 0;
else
LS_ready_ = atoi(token); // buggy
}

void rtProtoLS::intfChanged ()
{

Tcl & tcl = Tcl::instance();
// call tcl get-links-status, strtok, set linkStateList;
tcl.evalf("%s get-links-status", name());
const char * resultString = tcl.result();

// destroy the old link states
linkStateList_.eraseAll();
// tcl MUST return triplets of numbers

for (LsIntList intList2(resultString, " \t\n");
```

```
     !intList2.empty(); ) {
int neighborId = intList2.front();
intList2.pop_front();
ls_status_t  status = ( ls_status_t ) intList2.front();
intList2.pop_front();
int cost = (int) intList2.front();
intList2.pop_front();
// LsLinkState ls;
// ls.init(neighborId, status, cost);
linkStateList_.push_back(LsLinkState(neighborId,status,cost));
}
// call routing_'s LinkStateChanged()
//   for now, don't compute routes yet (?)
routing_.linkStateChanged();
}

void rtProtoLS::lookup(int destId)
{
// Call routing_'s lookup
LsEqualPaths* EPptr = routing_.lookup(destId);

// then use tcl.resultf() to return the results
if (EPptr == NULL) {
Tcl::instance().resultf( "%s",  "");
return;
}
char resultBuf[64]; // XXX buggy;
sprintf(resultBuf, "%d" , EPptr->cost);
char tmpBuf[16]; // XXX

for (LsNodeIdList::iterator itr = (EPptr->nextHopList).begin();
     itr != (EPptr->nextHopList).end(); itr++) {
sprintf(tmpBuf, " %d", (*itr) );
strcat (resultBuf, tmpBuf); // strcat (dest, src);
```

```
}

Tcl::instance().resultf("%s", resultBuf);
}

void rtProtoLS::receiveMessage(int sender, u_int32_t msgId)
{
if (routing_.receiveMessage(sender, msgId))
installRoutes();
}

// Implementing LsNode interface
bool rtProtoLS::sendMessage(int destId, u_int32_t messageId,
 int size)
{
ns_addr_t* agentAddrPtr = peerAddrMap_.findPtr(destId);
if (agentAddrPtr == NULL)
return false;
dst_ = *agentAddrPtr;
size_ = size;
Packet* p = Agent::allocpkt();
hdr_LS *rh = hdr_LS::access(p);
rh->msgId() = messageId;
rh->metricsVar() = LS_BIG_NUMBER;
target_->recv(p);
// sendpkt( *agentAddrPtr , messageId, size);
return true;
}
#endif // HAVE_STL
```

B.7.2 hdr-ls.h

```
#ifndef ns_ls_hdr_h
#define ns_ls_hdr_h
#include "config.h"
```

```
#include "packet.h"
struct hdr_LS {
        // metrics variable identifier
        u_int32_t mv_;
int msgId_;
// Adding new private variable for data
int data_;

        u_int32_t& metricsVar() { return mv_; }
int& msgId() { return msgId_; }
// Property function for setting and retrieving data value
int& dataValue() { return data_; }

// Header access methods required by PacketHeaderManager

static int offset_;
inline static int& offset() { return offset_; }
inline static hdr_LS* access(const Packet* p) {
return (hdr_LS*) p->access(offset_);
}
};
```

B.7.3 rtProtoLS.h

```
#ifndef ns_rtprotols_h
#define ns_rtprotols_h

#include "packet.h"
#include "agent.h"
#include "ip.h"
#include "ls.h"
#include "hdr-ls.h"

extern LsMessageCenter messageCenter;
```

```
class rtProtoLS : public Agent , public LsNode {
public:
        rtProtoLS() : Agent(PT_RTPROTO_LS) {
LS_ready_ = 0;
}
int command(int argc, const char*const* argv);
void sendpkt(ns_addr_t dst, u_int32_t z, u_int32_t mtvar,
int data);
void recv(Packet* p, Handler*);

protected:
// init nodeState_ and routing_
void initialize();
void setDelay(int nbrId, double delay) {
delayMap_.insert(nbrId, delay);
}
void sendBufferedMessages()
{
routing_.sendBufferedMessages();
}
void computeRoutes() { routing_.computeRoutes(); }
void intfChanged();
void sendUpdates() { routing_.sendLinkStates(); }
void lookup(int destinationNodeId);

public:
bool sendMessage(int destId, u_int32_t messageId, int size);
void receiveMessage(int sender, u_int32_t msgId);

int getNodeId() { return nodeId_; }
LsLinkStateList* getLinkStateListPtr()
{ return &linkStateList_;
}
```

```
LsNodeIdList* getPeerIdListPtr() { return &peerIdList_; }
LsDelayMap* getDelayMapPtr() {
return delayMap_.empty() ? (LsDelayMap *)NULL : &delayMap_;
}
void installRoutes() {
Tcl::instance().evalf("%s route-changed", name());
}

private:
// addr for peer Id
typedef LsMap<int, ns_addr_t> PeerAddrMap;
PeerAddrMap peerAddrMap_;
int nodeId_;
// to differentiate fake and real LS, debug, 0 == no
int LS_ready_;
// needed in recv and sendMessage;
LsLinkStateList linkStateList_;
LsNodeIdList peerIdList_;
LsDelayMap delayMap_;
LsRouting routing_;

int findPeerNodeId(ns_addr_t agentAddr);
};
// ns_rtprotols_h
#endif
```