The Royal Institute of Technology
School of Information and Communication Technology

Rezaul Hoque

Scalable Network Tomography System

Master’s Thesis
Stockholm, November 30, 2009

Examiner: Vladimir Vlassov,
The Royal Institute of Technology (KTH), Sweden

Supervisor: Andreas Johnsson,
Ericsson Research, Sweden
This work is dedicated to my father who has always inspired me to look beyond boundaries
Network Tomography enables network operators to measure end-to-end network metrics from several locations at the same time which are useful to a wide range of applications. In previous studies, network tomography was applied to attain a full mesh with active probing but most of the systems have central control and thus lack scalability. There are also a few which are distributed in nature but do not ensure full mesh coverage. They perform measurements of a subset of paths and try to use inference for remaining. But in case of measuring available bandwidth, inference is not a feasible technique.

In contrast, we set out to achieve a full mesh-probing scheme which scales to the nation-wide Internet without using any inference technique. In this thesis report a novel network tomography system that uses probabilistic distributed scheduling for peer selection to probe for measurements is proposed. The system ensures that eventually the solution tends to full mesh probing. Bandwidth measurements congests the network and utilizes more system resource compared to measuring metrics like link loss or latency. This is a constraint which is also taken into account. In pursuit of providing full mesh coverage in a distributed system, each node needs to know about each other, and hence a group membership management protocol is introduced. A simple way to store the probed data in a distributed storage which can be queried in $O(1)$ message hop is also presented. The system is designed as a plugin system which allows any existing measurement tool to be plugged in and perform measurements to extract the metrics of the network by active probing.
I would like to express my gratitude to all those who gave me the possibility to complete this thesis. I am deeply indebted to my supervisor Andreas Johnsson of Ericsson Research who has taught me how to carry out research work and gave me directions at every aspect of my work. In addition I am thankful to my co-supervisors Christofer Flinta and Svante Ekelin for their time and insightful remarks on my work.

I would also like to thank my examiner Vladimir Vlassov of Royal Institute of Technology who has done critical review of my work and suggested modifications to make it even better.

When things felt never ending, the light of hope came in the form of all my friends here and around the globe, who inspired and instilled courage in me to carry on. You all know who you are so I thank you all. I am deeply thankful to my brother Fazlul Hoque for his unconditional support throughout. You have truly been a source of inspiration. I am also ever grateful to my mother who has taken the pain of my absence, so that I can fill my ambitions.

Finally, I would also like to thank Per Karlsson, and the whole Packet Technologies (TLA) Research Group at Ericsson for giving me this opportunity to conduct interesting research and make my stay here a memorable one.

Stockholm, November 30th, 2009

Rezaul Hoque
# Contents

Abstract i  
Contents iv  
List of Figures vii  
List of Symbols and Abbreviations ix  

1 Introduction 1  
1.1 Motivation ......................... 1  
1.2 Problem Statement .................... 2  
1.3 Research Questions ................... 3  
1.4 Thesis Outline ...................... 3  

2 Related Work 5  
2.1 Network Tomography Overview ............ 5  
2.2 Existing Research .................... 6  
2.2.1 Overlay Management ................. 7  
2.2.2 Inferential Network Monitoring .......... 8  
2.3 Bandwidth Estimation Tool ............... 9  
2.3.1 Bandwidth Available in Real Time ......... 9  
2.4 Summary ................................ 10  

3 Network Tomography Issues 11  
3.1 Network Scalability .................... 11  
3.2 Node Scalability ...................... 12  
3.3 Administration Control and Usage ........... 13  
3.4 Additional Constraints .................. 13  
3.5 Measurement Storage ................... 14  
3.6 Summary ................................ 14
## CONTENTS

### 4 Proposed Network Tomography System
   4.1 Peer Selection and Measurements .................................. 15
   4.2 Switching Roles .................................................. 17
   4.3 Handshaking ..................................................... 18
   4.4 Performing Measurements ........................................ 18
   4.5 Measurement Data Storage ........................................ 19
   4.6 Measurement Data Query .......................................... 20
   4.7 Bootstrap ........................................................ 20
   4.8 Node Join ....................................................... 20
   4.9 Node Discovery .................................................. 22
   4.10 Summary ......................................................... 22

### 5 Proposed Algorithm
   5.1 Controlled Random Scheduling ................................. 23
   5.2 Lazy Group Membership Management ...................... 25
      5.2.1 Lazy Join .................................................. 25
      5.2.2 Lazy Leave ............................................... 26
      5.2.3 Conflict Resolution ...................................... 26
   5.3 Data Storage and Retrieval ..................................... 26
   5.4 Analysis of the System .......................................... 27
      5.4.1 Bootstrap Node Failure .................................. 27
      5.4.2 Control Load .............................................. 27
      5.4.3 Measurement Data Storage ............................... 28
      5.4.4 Piggybacking Control Messages ...................... 28
   5.5 Summary ......................................................... 28

### 6 System Simulation
   6.1 Simulator ....................................................... 31
   6.2 Simulation Results ............................................... 31
      6.2.1 Node Behavioral Strategy ............................... 32
      6.2.2 Varying Max Active Per Node ......................... 34
      6.2.3 Message Overhead ........................................ 35
      6.2.4 Measurement Coverage .................................. 36
      6.2.5 Concurrent Measurement Pairs ....................... 37
      6.2.6 Lazy Group Membership ................................ 38
      6.2.7 Varying Alive List ...................................... 39
   6.3 Summary ......................................................... 40

### 7 System Design and Implementation
   ................................................................. 41
7.1 Design Overview ........................................ 41
7.2 Implementation Details ................................ 42
  7.2.1 Packages Descriptions .............................. 42
  7.2.2 Class Descriptions .................................. 43
7.3 Execution Process ........................................ 44
7.4 Summary ..................................................... 45

8 Conclusion .................................................. 47
  8.1 Summary of Work ........................................ 47
  8.2 Research Findings ....................................... 48
  8.3 Future Work .............................................. 49

Bibliography .................................................. 51

A Screen Shots ............................................... 57
List of Figures

2.1 Link Capacity ........................................... 9
2.2 End-to-end Network Probing ............................ 10
3.1 Distributed Denial of Service Attack .................. 12
4.1 Total Alive List ......................................... 16
4.2 Peer Selection .......................................... 16
4.3 Switching Node Roles ................................. 17
4.4 Successful Handshake .................................. 18
4.5 Unsuccessful Handshake ............................... 18
4.6 Performing Measurement ............................... 19
4.7 Node Join Request .................................... 21
4.8 Node Joining ........................................... 21
6.1 Node Behavior .......................................... 33
6.2 Concurrent Measurement .............................. 34
6.3 Message Overhead ..................................... 35
6.4 Measurement Coverage ............................... 36
6.5 Concurrent Measurements ............................. 37
6.6 Lazy Group Membership View ....................... 38
6.7 Varying Alive List ..................................... 39
7.1 Components ............................................ 42
A.1 Screen shot - Bootstrap Node ......................... 58
A.2 Screen shot - Measurement Node ..................... 58
List of Symbols
and Abbreviations

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACK</td>
<td>Acknowledgement</td>
</tr>
<tr>
<td>BART</td>
<td>Bandwidth Available in Real Time</td>
</tr>
<tr>
<td>CDN</td>
<td>Content Distribution Network</td>
</tr>
<tr>
<td>AL</td>
<td>Alive List</td>
</tr>
<tr>
<td>CL</td>
<td>Crash List</td>
</tr>
<tr>
<td>TAL</td>
<td>Total Alive List</td>
</tr>
<tr>
<td>CR</td>
<td>Control Random Function</td>
</tr>
<tr>
<td>CRS</td>
<td>Controlled Random Scheduling</td>
</tr>
<tr>
<td>DDoS</td>
<td>Distributed Denial of Service</td>
</tr>
<tr>
<td>DHT</td>
<td>Distributed Hash Table</td>
</tr>
<tr>
<td>GMV</td>
<td>Group Membership View</td>
</tr>
<tr>
<td>IP</td>
<td>Internet Protocol</td>
</tr>
<tr>
<td>ISP</td>
<td>Internet Service Provider</td>
</tr>
<tr>
<td>LGMM</td>
<td>Lazy Group Membership Management</td>
</tr>
<tr>
<td>VoIP</td>
<td>Voice over IP</td>
</tr>
<tr>
<td>NPQ</td>
<td>Network Path Quality</td>
</tr>
<tr>
<td>ICMP</td>
<td>Internet Control Message Protocol</td>
</tr>
<tr>
<td>GUI</td>
<td>Graphical User Interface</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

Network Tomography [1, 2] is an area of research where a network is studied from an end-to-end perspective utilizing many participating nodes. Traditionally the nodes, which are located on the network edge, actively send probes in-between each other in order to measure performance parameters such as delay, loss and round-trip time. The combination of several end-to-end measurements supplies Internet providers and network operators with a "network performance map" that can be used in their operations, monitoring and maintenance systems.

One main advantage of network tomography, especially when placing the measurement nodes at the network edge or in end devices, is that networks composed of several administrative domains can be measured and monitored without access to any intermediate nodes. The network in-between is rather treated as a black box. Network tomography therefore scales, as the number of end nodes in a network naturally increase at a slower rate compared to the number of possible intermediate nodes when the network size grows.

1.1 Motivation

The Internet is growing rapidly and applications like VoIP, streaming video and distribution of other content are becoming increasingly popular. Due to the increasing demand on transferring content from one point to another, Internet Service Providers (ISP) and their customers experience network performance degradation caused by the best effort feature that is a major design principle of the Internet.
One practical way to achieve good performance is by monitoring certain metrics that describe the network. If the available bandwidth for several paths is known, it helps in making intelligent decision on how to optimize performance.

"If you cannot measure it, you cannot improve it" - Lord Kelvin

Having an active set of fresh data to deal with at all times is even better. For instance, in systems like Content Distribution Network (CDN) knowing metrics like bandwidth between to content servers helps in optimized routing of data which boosts the overall performance of the system. Thus ISP’s are highly interested to know certain metrics of the network like congestion and link loss. This helps them to make intelligent decisions to avoid congestion and bottlenecks. As a result, network tomography systems have been studied enthusiastically.

1.2 Problem Statement

Management of network measurement nodes can either be done using a centralized controller or a distributed control system. In a centralized controlled system, there is a single point of failure. When the central controller and its possible redundant backup fails, the system becomes non operational. Another disadvantage is the fact that network monitoring based on central controller does not scale very well to large networks. As the number of monitoring nodes increases, the load on the central server increases proportionally.

On the other hand a distributed control system does not have the drawback of single point of failure and also scales well, but it takes more control messages to maintain the overlay of network. To enable all the nodes to know about the other nodes in the network, protocols like group membership management [3] is needed. In addition, a scheduler has to be run in all nodes, compared to only running in only one node in a centralized system.

With the introduction of end-to-end available bandwidth measurement methods for continuous monitoring [4, 5, 6, 7, 8] new requirements are put on network tomography systems. For example, available bandwidth measurement methods introduce more overhead in terms of measurement
traffic compared to measurements of delay, loss and round-trip time. There are also requirements on scheduling of measurements to avoid overloading the participating nodes.

1.3 Research Questions

Based on the problem statement in the previous section, here are some Research Questions (RQ) that this thesis work must attempt to answer. The research questions are as follows:

RQ-01: If operators want to deploy bandwidth measurement methods, there is a need to control the induced overhead. How to achieve control over the network load posted by the network tomography tool?

RQ-02: How to make a scalable network tomography tool, specially when doing bandwidth measurements as they are considered to overload a bit more compared to measuring other metrics?

RQ-03: How to automate monitoring so that minimum or no supervision at all would be required while the tool will keep probing for measurements?

RQ-04: Is it feasible to use peer-to-peer based monitoring systems as an alternate to traditional centralized solutions?

RQ-05: What are the benefits of using a peer-to-peer monitoring systems?

1.4 Thesis Outline

The thesis presents a study on the scalability issues that are introduced from utilizing end-to-end available bandwidth measurement methods as part of a network tomography system. What precautions must be taken in
order not to overload participating nodes or the network while actively probing the network. A solution is presented eliminating the need for a central server for scheduling measurements. Peer selection is cleverly done to avoid too many measurements occurring at the same time. The problems of making a decentralized system for control management are identified. A novel architecture to do active network measurements is proposed and evaluated using simulations. Further it has been implemented as an active network measurement tool.

The key feature of the solution is that it takes into account the requirements posted by available bandwidth measurement methods, and tries to keep the control overhead low and make the system scalable. In addition it also tries to achieve some overall control over the amount of traffic it generates, that is the total amount of messages each network tomography end nodes are injecting to the system. A simple way of storing measurement results have been proposed, to provide easy accessibility.

After introducing the work in Chapter 1, related work in the field of network tomography is discussed in Chapter 2. Chapter 3 outlines the common network tomography issues and challenges that need to be addressed. The original work is presented in Chapters 4 which discusses the proposed system overview followed by Chapter 5, which further presents the algorithm. Chapter 6 evaluates the system using simulations. Then in Chapter 7 the implementation details of the tool is described. Chapter 8 draws a conclusion of the entire work and provides some suggestions for future work.
Chapter 2

Related Work

Network Tomography system has been an active field of study in both academia and the industry. Network Tomography research can be divided into two parts. The first part of research deals with control mechanisms for peer selection to perform measurements while the second part of research focuses on inferential network monitoring. The study of the thesis work is mainly focused on the first part that is control mechanisms for peer selection. In order to perform the actual bandwidth measurements from end-to-end nodes, an existing bandwidth estimation tool particularly Bandwidth Available in Real Time (BART) [4, 6, 7] is used.

2.1 Network Tomography Overview

It is highly desirable to have certain knowledge about network metrics. It helps in making intelligent decisions for network applications and is also helpful for operators to monitor traffic conditions. They can identify congestion points and performance bottlenecks. This is the reason why measuring and analyzing the network metrics have been studied carefully.

"Network tomography is the study of a network’s internal characteristics using information derived from end point data" - Wikipedia

Large-scale network inference involves estimating network performance parameters based on traffic measurements at a limited subset of the nodes. Vardi [9] was one of the first researchers to rigorously study this sort of
problem and he coined the term network tomography due to the similarity between network inference and medical tomography.

Several Internet-wide Network Tomography systems are operational today. A few of the parameters used to measure performance are for example jitter, delay, packet loss, round-trip time and also IP-layer topology obtained from ICMP-based methods. There are also research initiatives on methods and tools for inferring link or sub path characteristics from combining multiple measurements conducted between different end nodes.

The research on Network Tomography systems can be classified in at least two dimensions. The first type of classification refers to overlay management and control of network tomography systems. The control mechanisms for performing tasks such as starting, stopping, storing and analyzing measurement results are in the extreme cases either centralized or decentralized. The second dimension concerns with whether the network tomography system only performs end-to-end measurements or if an analysis of combined measurements is made in order to infer for example link characteristics and network topology. The focus of the thesis work is on the overlay management and control of the network tomography system, to do peer selection and perform active network monitoring.

2.2 Existing Research

The naive way of achieving real-time end-to-end measurements is by simultaneously measuring all paths between end hosts. In a large network this can pose a heavy load on nodes and network resources. In turn it may also give inaccurate results due to congestion in the network caused by too many active measurements sharing the same path at the same time. Inference is a solution to achieve scalability in monitoring large networked systems. Instead of measuring on all paths between end nodes, inference based solutions require few measurements from each node and infer all-pair properties from those few measured paths. There has been a large body of research focusing on latency estimation [10, 11, 12, 13, 14, 15, 16], loss rate [17, 18] and few on bandwidth estimation [19, 20]. Measuring bandwidth usually requires more probe packets to estimate it compared to measuring the other metrics such as latency or loss rate. Hence the study
of network tomography which measures bandwidth actively posses more challenges which are discussed in Section 3.4.

2.2.1 Overlay Management

Overlay management includes mechanisms for maintaining the network and peer selection to do measurements.

Chen et al. [21] proposed a measurement path selection algorithm based on linear algebraic approach which selectively monitors k linear independent paths that can fully describe all the $O(n^2)$ paths. The measurements of the k paths are used to estimate the path of all the others. They formally defined the method and showed that $O(n \log n)$ paths may fully cover $O(n^2)$ paths in a full mesh manner where n is the number of end nodes, by extensive simulations. Their study aims to infer the loss rates of all links in a subset of the network, i.e. all links in an overlay network constructed by the n end nodes. It is a centrally controlled system. Although this algorithm is well designed, it lacks scalability and not practical for the Internet scale. The target of n is about 100 in this algorithm due to the computation time of linear algebra.

Tagami et al. [17] introduces a distributed measurement path selection method and a data management method that constructs a Distributed Hash Table (DHT) on measurement paths. They use the DHT to do a distributed measurement path selection using inter-node associations of DHT for covering major portion of the network. To lower the overhead of data management method, the control messages are piggybacked on top of the measurement themselves. Their study reveals that the selected path traverses more than 96% links of the entire network. The main objective was to infer the loss rates of all links in a subset of the network, i.e the end nodes which construct the overlay of the network. They also show at that their proposed architecture has higher tolerance to message loss than centralized architecture.

The paper on measurement and estimation of Network QoS among Peer Xbox 360 game players by Lee et al. [22] performed active probing for measuring network path quality (NPQ) depending on network delay and capacity measured between players prior to each Internet game match up. Probe packets were sent between Xbox 360 consoles in order to find player with good enough network performance. They study was focused
on temporal and geographical correlations in the NPQ data and proposed three different kind of predictors which can provide a rough estimate of NPQ data between pair of players, based on previous NPQ estimates.

The DIMES system [23] is a distributed test bed for performing round-trip time and ICMP-based measurements between DIMES clients. A recent extension to the DIMES system is the Inter-packet Delay Measurement (IDM) module that adds available bandwidth estimation capabilities to the DIMES clients. The IDM module requires synchronization of nodes in order not to congest the network or overload the participating nodes. A centralized experiment planning control system is introduced in the paper.

ETOMIC [24] is a measurement infrastructure consisting of almost two dozen nodes located throughout Europe. The nodes are capable of estimating performance parameters such as one-way delay, jitter and packet loss in-between each other. High-resolution time stamps are enabled through DAG cards. Further, a GPS receiver synchronizes the ETOMIC clock. The control of the ETOMIC nodes is centralized in its nature.

2.2.2 Inferential Network Monitoring

By combining topology knowledge (either known beforehand or obtained through ICMP-based methods such as trace route) with end-to-end measurements it is possible to infer link characteristics.

In [2], an introduction to inferential network monitoring methods is given. Two main problems exist in this area of research. The first and most obvious one is to develop statistical methods that infer link-level characteristics from end-to-end measurements and topology data. The second problem is to minimize the number of required end-to-end measurements to solve the inference problem.

Demirci et al. [25] describes a method for monitoring networks utilizing both end-to-end measurements and monitoring of underlying link segments. By monitoring underlying link segments the number of end-to-end paths to monitor is reduced. It should be noted that this technique requires collaboration between the overlay network operator and the underlying network provider.

In [17], a distributed system for selecting which end-to-end paths to
monitor in large-scale networks is described. The method basically selects a set of other nodes to perform measurements to. If the number of measurement nodes is large enough the probability of covering enough links is shown to be high. A distributed data management system utilizing distributed hash tables is also proposed.

2.3 Bandwidth Estimation Tool

Currently the most accurate bandwidth measurement techniques is to directly measure the fastest rate that traffic can be sent through a network. Wide-scale deployment of these ‘heavy-weight’ bandwidth tests can overwhelm the network with test traffic. Accurate measurement of bandwidth is difficult if simple large data volume techniques are not used [26]. In particular there are two popular bandwidth estimation techniques, single packet and packet pair techniques.

2.3.1 Bandwidth Available in Real Time

BART ("Bandwidth Available in Real Time") [4, 6, 7] is a method for estimating path available capacity and other capacity-related parameters in real time over packet-switched network paths.

![Figure 2.1: Link Capacity](image)

The concepts of the performance parameters capacity, utilization and available capacity are shown in Figure 2.1 which illustrates three links that are part of a network path. Each link has a capacity that defines the maximum rate at which IP traffic can be sent. At any given point in time the links are utilized. This is exemplified by the shaded area in each box. Then, for each link the available capacity is calculated as the difference between the link capacity and the utilization.
The BART method relies on actively sending probe traffic over a network path in order to determine at which probe rate the path shows signs of congestion. This rate defines the available capacity - the fraction of the capacity not utilized by IP traffic\textsuperscript{1}. The concept of sending probe traffic is shown in Figure 2.2. A BART sender is transmitting IP packets at randomized inter-packet separations towards a receiver; the separation is affected by other IP traffic sharing the network. The receiver timestamps each incoming IP packet and calculates the new inter-packet separation.

\begin{figure}[h]
\centering
\includegraphics[width=0.5\textwidth]{network-probing.png}
\caption{End-to-end Network Probing}
\end{figure}

The inter-packet separation at the sender and receiver are analyzed by a Kalman filter, a statistical method for tracking properties in real time that are not directly observable, and the output from that analysis is the available capacity.

\section*{2.4 Summary}

The related research work done in the field of network tomography both in the area of overlay monitoring for peer selection and inferential monitoring shows many approaches in solving the challenges to make scalable active network tomography tool. It is also noted from the recent research that it is possible to send application data inside the probe packets [17, 27]. This reduces the overhead in terms of probe traffic and thus increases the value of available bandwidth estimation methods. Bandwidth estimation requires more packet injection into the network and thus it should be taken into account when designing an active network tomography system.

\textsuperscript{1}The terms available capacity and available bandwidth are used interchangeable
Chapter 3

Network Tomography Issues

Active Network Tomography systems continuously measures to obtain network metrics. To do measurements it require injecting packets into the network. It also needs control packets to maintain the scheduling for doing measurements. Moreover, in a decentralized network tomography system, maintaining the knowledge of the nodes that are part of the network itself needs management packets. In short network tomography system puts load on the network to extract network metrics. However, the primary objective of a network tomography system is to extract network metrics so that the network load can be balanced for optimized routing. Thus its behavior is conflicting to its objective. So such a system should be carefully designed so that, the increase of traffic caused by control messages should be insignificant comparing with the actual load present on the network. This chapter discusses issues that need to be taken into consideration when designing the measurement control system which is an essential component of any active network tomography system.

3.1 Network Scalability

The main drawback, closely coupled to active measurements, is the fact that probe packets are injected into the network. Thus, if the network is heavily measured from different end nodes the load on individual links can unintentionally become high.

Measuring performance parameters such as packet loss, delay, round-trip time or jitter on an end-to-end path does not require a high
amount of probe traffic, enabling the network to work without much concern even when one end-to-end measurement is scaled up into many ongoing measurements at the same time over the same links. The number of required probe packets is usually a tune able parameter in methods that estimate available bandwidth. However, more packets are required in comparison to measuring the above mentioned performance parameters. The scalability issue arises when several network tomography nodes at the same time, independently of each other, measure paths that partly overlap. The overlapping network segments may be congested due to probe traffic.

3.2 Node Scalability

Available bandwidth measurement methods [28] require high precision time stamps on sent and received probe packets. If one node is participating in more than one end-to-end measurement there is a risk for delays in time stamps induced by probe packets originating from other nodes. Thus, the available bandwidth estimation results may be affected by other concurrent measurement sessions even though they are not measuring the same path. The conclusion is that the receiver node, that is one of the nodes actually conducting the measurements, may become the bottleneck.

Figure 3.1: Distributed Denial of Service Attack

In the extreme case, if the measurements are not scheduled in a cleaver way, several nodes may initiate an available bandwidth measurement towards one common receiver at the same time as depicted in the figure.
Below. This node will then experience something similar to a distributed denial-of-service attack (DDoS). It will become overloaded by probe traffic and thus the different probe traffic streams will affect each other in the one common bottleneck - the receiver node as illustrated in Figure 3.1.

3.3 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 include 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. What innovations can reduce administration traffic? At the same time address the issues of network and node scalability.

Further, the manual labor required to administrate and manage a monitoring system must be keep at a minimum. Autonomic network management, and in particular with this paper in mind, autonomic network monitoring is an active research topic. This is further elaborated on in [29]. As a remark it should be noted that reducing the administrative network traffic also ease the node scalability problem which was discussed in Section 3.2.

3.4 Additional Constraints for Bandwidth Measurement

Measuring metrics like delay, round trip time does not deal with heavy number of message packets needed to be injected in the network. For instance, to measure round trip delay, only one packet needs to be injected in the network which travels to the destination and comes back. In contrast to do bandwidth measurement a number of packets need to be sent in order to get a good estimation [6, 7, 8]. Often, the more packets injected the better estimation can be attained. It causes node scalability
which is discussed in Section 3.2. When designing an active network monitoring tool, this has to be taken into consideration that the cost of each measurement done is high. Here cost means, the amount of traffic that is injected in the network.

### 3.5 Measurement Storage

Measurement data should be stored cleverly for easy access. As in active network tomography systems, continuous measurement of nodes can pile huge amount of measurement data. All measured data can be stored in one node for easy access. But the huge amount of data in one machine may lead to slower retrieval of data. Clever ways of storing using indexing are sometimes used. Data can also be stored in a distributed storage and is a way to relieve the load from one node. Caution should be taken to choose a way to storage, so that data is available, accessible and query is quick.

### 3.6 Summary

To build an active network tomography system there are some common network tomography issues such as node scalability, network scalability and administration control. These issues are all related to minimizing the amount of generated traffic by the network tomography system and to avoid distributed denial of service attacks on end nodes.
Chapter 4

Proposed Network Tomography System

The proposed Network Tomography system consists of a set of participating nodes that are capable of performing active network measurements. There is no central control system, each node is equipped with an autonomic management protocol that dictates how to schedule measurements with other nodes, how to treat joining and leaving from the network as well as where to store measurement results. In order to do the bandwidth measurement, existing bandwidth measurement tool is used.

4.1 Peer Selection and Measurements

The system contains two kinds of nodes, measurement nodes and sensor nodes. Measurement nodes are responsible for triggering measurement with the selected peer nodes. And sensor nodes only participate in measurements. Each node can either be a measurement node or a sensor node, and periodically each node re-asses its role. The nodes can be configured to set the maximum number of concurrent measurements it can perform at a time.

Each node maintains a list of all other nodes it knows about in the system. The list is called the total alive list. When the node initially starts, it randomly chooses a node in the list to start with, and then onwards moves sequentially to the next node in the list and so on. Figure 4.1 depicts the traversing of a total alive list.
Choosing from where to start in the list is important and needs to be random, or else if everyone had the same list and all started choosing the same partners to do measurements they would all get rejections. It would create a distributed denial of service attack. The total alive list is a circular list.

In the scenario depicted in Figure 4.2, the configuration for maximum concurrent per node is set to 1. Node A and node E are measurement nodes. Node A choose node D to trigger measurement with and thus sends a handshake message to see if node D is available to perform a measurement. In this case node D is free as it is not participating in any other measurement. It sends back a positive acknowledgement and then they perform a measurement among themselves. Similarly node E performs a measurement with node A.

In snapshot 2, after node B has completed doing its measurement it moves on to the next node in its list that is node E to trigger measurement with. It sends out a handshake message to it, but receives a negative acknowledgement as node E is a measurement node and itself if performing measurements. As shown in snapshot 3, node B moves on to the next node in its list. It sends out a handshake message to node F. But node F is currently participating in a measurement with node E, and as it is configured that each node can only participate in at most one measurement at a time, it sends back a negative acknowledgement. Node
4.2. SWITCHING ROLES

B can move on to the next node in its list which is node A. It sends a handshake message to it. Node A is free and sends back a positive acknowledgement and further does a measurement with it.

4.2 Switching Roles

After each periodic time $t$, nodes re-evaluate what its role should be using a Controlled Random (CR) function. This function basically does a probability check whether it should be a measurement node or sensor node. This ensures that each node at some point of time will become a measurement node and thus can trigger measurements with nodes in its list. The system can be configured to say what percentage of measurement nodes should be present. The CR function using probability ensures that approximately the system configured percentage of measurement nodes exists in the system at any given time. Given this property of being able to set the percentage of measurement nodes in the system, and also given the fact that only measurement nodes inject packets into the network, it can be safely estimated how much messages are being injected into the network.

As show in the Figure 4.3, at time $T$ Node A, D and E are measurement nodes, and the rest of the nodes are sensor nodes. After a periodic time $t$, using the Control Random (CR) function every node re-evaluates its role. Node B which was a sensor node, re-evaluates to become a measurement node. Node C and F both remain as sensor nodes. Node E remains as measurement node. It is to be noted that switching here doesn’t refer having to switch roles, rather does a probability and based on its outcome either switches to another role or remains as it is.
4.3 Handshaking

When a node triggers a measurement with another node, it sends a HANDSHAKE_REQ message to the node. As a response it either sends back a HANDSHAKE_ACK which is a positive ack or it sends a HANDSHAKE_NACK which is a negative ack. This is depicted in Figure 4.4 and 4.5. The receiving node sends a positive response only if it is free to accept any measurements. Each node is configured to have a maximum number of concurrent measurements it can perform at a time.

![Figure 4.4: Successful Handshake](image)

![Figure 4.5: Unsuccessful Handshake](image)

4.4 Performing Measurements

Only after receiving a HANDSHAKE_ACK message, the node starts the measurement tool (e.g. BART tool) to perform the measurement. When the node sends back the HANDSHAKE_ACK message, it also starts the BART Receiver in a random port and packs the ack message with this port number. This port is used by the sender node to trigger a measurement,
4.5 Measurement Data Storage

The measurement value is calculated on the receiver node which has the BART Estimator running. When measurement value is calculated this value is stored locally. A copy of this value is also sent back to the sender node, which triggered the measurement. The values are saved along with the IP and port of the partner node with whom the measurement was performed. For instance if a measurement between node A and B is done, node A will save the data as [B’s IP and port, Bandwidth, TimeStamp] and B will store the same data as [A’s IP and port, Bandwidth, Timestamp].

Another alternative of storing measurement data is to store it in a central database. But this again would have introduced a single point of failure. To avoid complete data loss, replication needs to be introduced. In addition, with increasing number of nodes in the system, the central database will start getting too many requests to save data at the same time, and thus might suffer from distributed denial of service (DDoS). In contrast the proposed system of data storage has an implicit replication of...
as the data is saved on both the nodes participating in the measurement. The probability of both nodes failing at the same time is less. The proposed system also avoids the possibility of DDoS completely.

### 4.6 Measurement Data Query

A query for the bandwidth between Node A to Node B, is found by simply requesting both or either Node A or Node B for the bandwidth. If the pair had measured a bandwidth among them, both would have stored it locally, and hence can return the query result. Thus the query takes only $O(1)$ messages.

### 4.7 Bootstrap

The first node in the system is a special node usually called the bootstrap node. It has to be started manually. All other nodes joins must know the the IP and port in which this bootstrap node is running. Hence the access information of this bootstrap nodes are known before hand. when a node wants to join the system it sends a JOIN_REQ to the bootstrap node. The bootstrap node sends back a list of all nodes it currently knows about. When ever a new node joins, the bootstrap node saves a its total alive list locally, so that if crashes and wakes up again, it can regain a view of the system. An analysis of a scenario where saving the list is necessary is discussed in Section 5.5.1.

### 4.8 Node Join

A simple JOIN_REQ is illustrated in Figure 4.7. When a node sends the bootstrap node a JOIN_REQ, it replies back with a JOIN_ACK message which contains the total alive list of the bootstrap node.

A simple example showing the node joining process is show in Figure 4.8. Node A is started first as is the bootstrap node. Then Node C, D and E joined the system each through the Node A (bootstrap node). Initially the bootstrap has an empty total alive list. When node B requests to join, it gets Node A’s (bootstrap node) total alive list. As it is aware of Node
A's existence, it adds node A to its total alive list. Similarly as Node A gets a join request from B it knows about Node B, hence it adds it to its total alive list. Snapshot 1 depicts the state after the node has joined the system. Total alive list of Node A has [ B ] and Node B contains [ A ].

![Node Join Request](image1)

Figure 4.7: Node Join Request

Further, Node C sends a join request to bootstrap node A. Node A sends its total alive list which contains [ B ]. Node C updates its total alive list to contain [ B ]. It also adds Node A to its list, as it is aware of node A, and thus its list is updated to contain [ B, A ]. Similarly since Node A got a request to join from node C, it adds C to its total alive list, and its list is also updated and has [ B, C ].

![Node Joining](image2)

Figure 4.8: Node Joining

As it is obvious from the scenario now is that the previous nodes, do not get to know about the new nodes which joins the system. The last node joined gets the most information about the system from the bootstrap node. All nodes exchange smaller lists containing new node and crash node information among each other by which gradually ever node gets to know about each other. This lists are piggybacked on handshake messages.
4.9 Node Discovery

Each node maintains two small lists, alive list (AL) and crash list (CL) apart from the maintaining the total alive list (TAL). The AL holds the n most recent nodes to be discovered alive, and CL holds the last n nodes to be discovered as crashed. When a node has been contacted for measurements and is not in its TAL, it is assumed to be a new node and hence put in the AL. And when node is contacted for measurement and doesn’t respond at all, it is assumed to be have crashed and put in the CL. When ever two nodes interact with each other to perform measurements, they exchange each others AL and CL lists which is piggybacked in the handshake messages, as shown in Figure 4.4 and 4.5. Using the AL and CL the TAL list is updated and hence gets a better view of the nodes in the system. Small lists are used in compared to sending the whole TAL, to minimize data transfer. And the updating scheme becomes simpler.

4.10 Summary

The measurement node chooses nodes from its total alive list and triggers measurement one after another. Sensor nodes receive measurement requests, and if they are free, participates in the measurement. The actual measurement is done by the BART tool. After measurement is done, the results are saved locally on both nodes. Each node periodically re-asses its role using the control random function of whether it will be a measurement or sensor node. The system continuously keeps triggering measurements, and each nodes switch roles periodically.
Chapter 5

Proposed Algorithm

The algorithm for the proposed Network Tomography system is divided into three main parts. Peer selection to perform measurements, overlay management for maintaining the group of nodes in a decentralized network and a mechanism to store and query the measurement data. Peer selection is done by the measurement nodes and triggers measurement with them. Each node switches role periodically, so that every node becomes a measurement node at some point and can trigger measurements with other nodes. Overlay management is done by exchanging small lists containing the last few joins and crash node information. This information is piggybacked on top of the handshakes messages, which are done in peer selection process. Measurement data are stored on both the participating nodes locally. To query for data requires $O(1)$ message.

5.1 Controlled Random Scheduling

A decentralized control protocol is proposed for the active measurements of end-to-end available bandwidth between nodes in a network. The protocol takes into account not to overload the network or the participating measurement nodes by limiting the number of concurrent measurement each node can participate in. The number of concurrent measurement possible per node is configurable. The algorithm has some degree of control over the overall ongoing active measurements in the whole system and thus it is named Controlled Random Scheduling (CRS).

Each node maintains a view of the entire network by running a Lazy
Group Membership Management Protocol (LGMM). This simple LGMM allows every node to know every other node in the system, thus forming a complete mesh structure in due time. Periodically each nodes re-evaluates whether to be a Measurement (M) or Sensor (S) node, using a controlled random function.

**Controlled Random Function**

The controlled random function is responsible for deciding which role the node should play. The possible outputs are either M node or S node. The input to the function is desired percentage of M nodes denoted as $P$.

$$\text{IF } (\text{random()} \mod 100) < P \text{ THEN}
\begin{align*}
&\text{RETURN Measurement} \\
\text{ELSE} &\text{RETURN Sensor} \\
\text{END IF}
$$

**Pseudo-code : Controlled Random Scheduling**

Peer selection to perform measurements using the control random scheduling algorithm is outlined as follows:

1. Using the controlled random function decide whether the node should be M or S Node.

2. If it is a M Node:
   - Initiate measurements with the members of the group.
   - After time $t$ expires, the node saves its state, noting the nodes with which it has performed measurements already.\(^1\)
   - Go to step 1.

3. If it is a S Node:
   - Participate in maximum $x$ number of measurements at the same time if requests for participation are received from M nodes.\(^2\)
   - After time $t$ expires go to Step 1.

\(^1\) $t$ is the periodic time after which each node re evaluates its role
\(^2\) $x$ is the number of active measurements a node can participate at a time
5.2 Lazy Group Membership Management

To perform measurements between two nodes, each node needs to be able to find other nodes in the network. In a centrally controlled system, this information is present to the central controller. But in a distributed system, the information is not available. Hence a group membership protocol needs to be introduced that keeps track of participating nodes and also handles joining and leaving of nodes. Introducing any group membership [3, 30, 31] naturally adds more overhead in terms of management messages needed to maintain a consistent view of the system at all nodes. The more accurate view of the group is required, more the control messages are needed to achieve it. In the proposed system it is not necessary to have a strict view of all the members at all times. Thus a new Lazy Group Membership Management (LGMM) is introduced, which does not bother itself in trying to keep a complete and consistent view at all times, rather just ensures that eventually the view is installed at all nodes. Every node maintains three lists:

- Total Alive List (TAL) - Total set of alive nodes
- Alive List (AL) - Most recent nodes discovered as alive
- Crash List (CL) - Most recent nodes discovered as crashed

When nodes communicate with another node to perform measurements, they send handshake messages to confirm the availability of the peers. The AL and CL lists are piggybacked in this handshake messages. Upon receiving a new CL and AL list, the node updates the TAL list to give a more updated view of the nodes currently at the system, and also updates each others AL and CL list if required.

5.2.1 Lazy Join

When a node joins not every one instantly gets to know about that new node. Either when the node becomes an M node and starts measuring with other nodes, those nodes get to know about the new node. Or when some other node discovered the new node and further participates in measurement with other nodes, through the exchange of AL lists, the information of the new node join slowly reaches everyone.
5.2.2  Lazy Leave

When a node crashes and is detected by another node, similar to lazy join, it is not propagated to every other node instantly. Instead through sharing of CL lists the view is stabilized.

The lazy join and leave might cause for different nodes to have different views of the system at a certain time, but the views will not differ largely. And since the mechanism is self healing (exchanging lists to continuously update each others view) we can say that the view of each node only improves and moves toward the actual view.

5.2.3  Conflict Resolution

Conflicts might occur when updating views. For instance when a node A shared a crash list where it mentioned node X is supposed to be dead, but node A has X in his alive list saying this node just joined recently. As there is no ”happen before” (synchronization of events) relation between events, there is no way to settle the conflict of who might be right. It is thus resolved by performing a ping to the node which is doubted to be alive or dead, in this case Node X. The lists are updated according to the ping response. If the receive a response that means it is alive else the marks it as crashed.

5.3  Data Storage and Retrieval

Measurement data is stored in a way so that they are accessible when needed. A simple way to keep the data in a distributed storage which can be accessed in $O(1)$ is to save the measured data in both the nodes locally. The measured data is saved in the following format:

<table>
<thead>
<tr>
<th>Node IP and Port</th>
<th>Bandwidth</th>
<th>Time Stamp</th>
</tr>
</thead>
</table>

When a node performs a measurement between A to B, results of the measurement is stored both in A and B. This ensures a replication factor of 2 for each data implicitly. Due to the simplistic nature of the data; it is thus possible to directly hop to the location of the data. When a data is requested for the node IP pair, the measured data can be either fetched
from any of the node as the data should be available in both the nodes. For instance, if we require the bandwidth of node A to node B, we can simply request either node A or node B to get the bandwidth. The query takes $O(1)$ message to retrieve.

5.4 Analysis of the System

5.4.1 Bootstrap Node Failure

If the bootstrap node crashes, the system will not come to a halt. As the rest of the nodes are independent to the bootstrap node, they will keep doing measurements. The only problem would be no new nodes can join the system. The bootstrap node saves its TAL list every time a new node joins the system through it. If it did not save this list then in a scenario where the bootstrap node crashes and wakes up again, and then some new nodes joins the system, it would not know about the previous nodes and hence would give an empty TAL list to the joining node. This would lead to a formation of different clusters. The new nodes would form another cluster of measurement nodes which will not perform measurements together with the nodes in the cluster formed before the crash. The behavior of the bootstrap node makes the system very robust and there is no single point of failure.

5.4.2 Control Over the Network Load

The measurement nodes are responsible for triggering measurements and has a direct relation to the amount of traffic generated by the network tomography system. The percentage of measurement nodes desired for system is configurable. Thus it is possible to control the amount of messages being injecting in the network. However, the lowering the number of measurement nodes might decrease traffic generated by the system, but it will also mean that the system will take longer time to complete full sweep of measurements with all other nodes. A trade-off between the amount of probe traffic and the time to complete a full-mesh measurement result has to be considered by the operator deploying the network tomography system.
5.4.3 Measurement Data Storage

The measurement data are not stored centrally because then a single point of failure is introduced into the system. Though it is known that the central server may be more reliable than the measurement nodes, as it would have been a dedicated server equipped to store data. Nevertheless, since each node which is also intended to be used by operators will be running the nodes in reliable machine. The current approach has a data replication factor of 2, which would have taken extra cost to maintain in a central database. If a central database would have been used, then all nodes have to push data to the server, and it would become a bottleneck. Further, the amount of measurement data can become extensive and thus such transfer of data may also congest the network. The query on the central database would have taken $O(1)$ messages which is the same as it takes in the current distributed storage, as discussed earlier. It could have been chosen to replicate more, but that would have introduced more messages in order to maintain the replications.

5.4.4 Piggybacking Control Messages

In lazy group membership management, piggybacking small list was used to update view of the system as in Section 5.1, to reduce message injection into the network. The other alternative was active gossiping [3] to update views, where messages are sent out actively to every other one when it discovers a node join or crash. If there is no nodes joining or crashing then the active gossiping does not send out any message, but in the piggyback approach, it continuously exchanges lists whenever a handshake is done. Hence it can be noted that if the network is stable and there is minimum churn, active gossiping method generates less control message compared to the piggybacked messages.

5.5 Summary

A simple peer selection algorithm CRS is presented which is responsible for choosing peer nodes to perform measurements with. Nodes periodically decides whether to switch between M and S node depending on the CR function. A group membership algorithm LGMM is presented which is very lazy in updating its view, but ensures that eventually it installs the
correct view at all nodes. The LGMM piggybacks update information on top of CRS’s handshake messages, eliminating the need for it to send control packets. Also a simple data storage algorithm is described where measurement data is stored locally on the participating nodes of the measurement.

In the analysis it has been shown that even if the bootstrap node crashes it does not introduce single point of failure, and in such a case the system continues to operate as expected. The only exception being no new nodes can join the system if the bootstrap node is dead. The TAL list of the bootstrap is saved locally to avoid a special case where network partition might occur. The analysis also points out that piggybacking data might cost more overhead in systems with no churn.
Chapter 6

System Simulation

The proposed Network Tomography system is evaluated in a simulator. The behavior of a node in the system is studied varying different parameters to analyze the effect it has on the overall system. The properties of the system are highlighted using the simulation results.

6.1 Simulator

PeerSim [32] is used to simulate the proposed network tomography system. The event based model engine of PeerSim is used. It is an open source Java based peer-to-peer simulation framework aimed to develop and test any kind of peer-to-peer algorithms in dynamic environments. The main features of PeerSim are as follows:

- Scales up to 1 million peers
- Highly configurable
- Open architecture and component based

6.2 Simulation Results

It is assumed that each node can participate in at most one ongoing measurement at a time and thus the max active (max concurrent per node) parameter is set to 1 for the simulations. This is of course allowed to change in an operational system. However, for several of the parameters
studied, having only one concurrent measurement is considered the "worst case". A system of approximately 100 nodes is used if not mentioned otherwise. The desired percentage of M nodes is set to 30%. The time required to perform a BART measurement is set to vary between 1 to 1.5 minutes. The group membership’s alive and crash list size is set to 5. That means each node maintains a list of last 5 new nodes that it discovered and last 5 nodes that it knows has crashed. And all nodes have internal back-off behavior which is discussed in the next section. Some of the simulations and their analysis are presented below.

6.2.1 Graph A : Behavioral Strategies of Measurement Nodes

Two behavior strategy for the measurement node has been studied.

Method A - Internal Back-off Behavior

When a M node tries a handshake and gets a negative response, the M node goes to sleep mode for a very small amount of time before moving on to the next scheduled node. This is similar to an internal back-off time. At this time the node does not trigger any new measurements, therefore temporarily behaves like a S node. It will accept incoming requests for measurements if the condition \(\text{currentActive} < \text{maxActive}\) is true. The node behavior is summarized as follows:

- Internal back off time, in between triggering for measurements
- Accepts incoming requests during internal back-off time

Method B - No Internal Back-off Behavior

When a M node tries a handshake and gets a negative response it simply moves on to the next scheduled node. No waiting period. M nodes will not accept any incoming requests. They only trigger measurements. The node behavior is summarized as follows:

- Triggering for measurements one after another (No delay in between)
- Does not accept any incoming requests
6.2. SIMULATION RESULTS

Figure 6.1: Node Behavior

Analysis

From the two graphs shown in Figure 6.1, it can be observed that method A over performs method B, especially when M nodes are high in the system, method A gives a reasonable ratio of successful to unsuccessful messages, compared to the poor ratio achieved by method B. The reason why method A performs well even when M nodes are more, is for the internal back off for M increases the probability of an incoming request to be accepted. In method B, when M nodes increases, most of the nodes does not have the ability to accept any measurement request as they are triggering measurements themselves and are not allowed to accept any incoming requests. Hence the number of unsuccessful messages increases. Whereas in the same scenario, method B performs dramatically well, the reason is for its internal back-off mechanism. When the M node goes to internal back-off there is a good possibility for it to accept an incoming request.

For rest of the simulations all nodes have internal back-off behavior.
6.2.2 Graph B: Effect of varying Concurrent Measurement per Node

The y-axis is the normalized ratio of successful to unsuccessful handshake messages, and the x-axis is the normalized ratio of measurement nodes over sensor nodes. Two graphs have been plotted, one with the configuration of max active per node set to 1 and the other to 2. From the graphs it can be observed how the ratio of successful messages vary with max active per node.

![Graph B: Effect of Concurrent Measurement per Node](image)

**Figure 6.2: Effect of Concurrent Measurement per Node**

**Analysis**

The x-axis shows the ratio of M nodes over S nodes. And the y-axis represents the ratio of successful message to unsuccessful message handshakes. Both the ratios are normalized. From the graph in Figure 6.2, we can see as ratio of M nodes increases, the ratio of successful messages comes down. This is solely because, most nodes are M node and are trying to trigger measurements, and thus they are not free to participate. Hence the number of unsuccessful messages increases and the

\[\text{Max concurrent measurement possible per node}\]
6.2. SIMULATION RESULTS

The overall ratio of successful messages goes down. Looking at the two graphs, we can say that as the number of max active per node increases the whole graphs shifts up, that is the ratio of successful over unsuccessful message handshakes increases.

It is notable that increasing the number of max active per node, will in turn also increase the overall number of handshake messages as well as increase the total number of measurements in the system. Therefore it has to be increased wisely depending on whether each node is capable enough to participate in that many measurements at the same time, and on how much load is acceptable to be injected in the network.

6.2.3 Graph C : Message Overhead per Node

The graph plotted in Figure 6.3 shows a relation between the message overhead per node as the total node of system increases. As each node receives handshake requests from other nodes, it has to acknowledge each message by either a negative or positive response. This is a load on the node. Therefore we simulate to see how the load on each node increases as the number of overall nodes in the system increases.

![Figure 6.3: Message Overhead per Node](image-url)
CHAPTER 6. SYSTEM SIMULATION

Analysis

It can be observed from the plotted graphs that with increasing number of nodes in the system, the handshake message overhead (control messages) per node remain almost constant. That is the load is distributed equally among all the nodes.

6.2.4 Graph D : Measurement Coverage Percentage

To show that eventually each node has full mesh coverage. A system of 50 nodes, were tested and among that a random node was selected to see how much time it needs to do a complete measurement with all other nodes in the system. Similarly experiment with a system of 100 and 150 nodes were done. The results of all three experiments are plotted in Figure 6.4. The x-axis represents unit time in seconds and the y-axis represents the measurement coverage percentage.

![Image of graph showing measurement coverage percentage over time for different node counts]

Figure 6.4: Measurement Coverage Percentage

Analysis

It can be observed that all the graphs tend to move towards 100% measurement coverage, therefore it can be concluded that eventually the each node performs measurement with all other nodes in the system
system ensures full measurement coverage for every node. Another observation from the graphs plotted is that the larger the number of nodes in the system the longer timer it takes for a node to attain full mesh measurement coverage.

### 6.2.5 Graph E: Concurrent Measurement Pairs in the System

The Figure 6.5 demonstrates the relation between percentage of measurement nodes in the system with the number of total ongoing concurrent measurements. The x-axis represents unit time and the y-axis represents the number of ongoing concurrent measurements in the whole system. The first scenario had 20% measurement nodes in the system and the second one had a 30% measurement nodes. Graphs for both the scenario are plotted.

![Figure 6.5: Concurrent Measurement pairs in the system](image)

**Analysis**

It can be deduced that the active concurrent measurement pairs are fairly constant in time. It can also be observed that as the percentage of M nodes is increased in the system, the overall concurrent measurement pairs
in any specified time increases, but still it is fairly constant in time. It is also notable that, the percentage of M nodes is directly proportional to the number of active concurrent measurement pairs in the system. Therefore there is a direct control over the percentage of M nodes in the system and this percentage is configurable. Thus, there is direct provides control over the number of active concurrent measurement pairs in the system. It can be concluded from the graphs that the system is highly scalable, and there is enough control over the system to ensure the network is not over used and hence can maintain network scalability.

6.2.6 Graph F : Lazy Group Membership View

The lazy group membership management protocol which is discussed in Section 5.3, is simulated to show that all nodes eventually attain a complete view of the system. From the simulation three random nodes were chosen and has been observed to see if they attain a complete view of the system or not. The three graph shown in Figure 6.6 depicts the view of the three nodes and how it changes over time. The x-axis of the graph represents unit time and y-axis represents system view percentage.

![Figure 6.6: Lazy Group Membership View](image)
6.2. SIMULATION RESULTS

Analysis

When node A joined it had 1% view of the system. Node B had around 15% and node C around 40% view of the system. Gradually as time progressed, all the three nodes converge to 100% view of the system. From the graphs it can be concluded that the lazy group membership view management eventually installs a 100% view of the system for every node.

6.2.7 Graph G : Lazy Group Membership View with varying Alive List

The lazy group membership management protocol which is discussed in Section 5.3, is simulated to show that all nodes eventually attain a complete view of the system. The alive list which is exchanged to update views among nodes, has a direct impact on how fast the final view is attained. A simulation was carried out by having the alive list set to 5 and another time set to 10. The simulation results are plotted in Figure 6.7.

![Figure 6.7: Lazy Group Membership view with varying alive list](image)

Analysis

From the two graphs in Figure 6.7, it can be seen that green graph (alive nodes = 10) takes less time to reach 50% view coverage. After that both
the graphs have almost the same rate. It can be concluded that increasing the list size has considerable effect on the rate in which view of the system is installed at nodes.

6.3 Summary

Simulations show that the proposed network tomography system is both node scalable and network scalable. It also depicts full mesh measurement coverage property. The direct relationship of messages injected into the network by the network tomography system, versus the amount of measurement nodes present in the system are highlighted. The more the number of measurement nodes in the system, the greater load it injects into the network.
Chapter 7

System Design and Implementation

The Network Tomography tool has been implemented based on the proposed algorithms discussed in Chapter 7. It is developed using Java and hence is platform independent. The tool has been designed such that any external measurement tool can be easily incorporated in it for doing measurements. The implementation has been done in modular approach where each components are separated. The implemented tool is tested by deploying it on several separate machines connected by the Internet\(^1\).

7.1 Design Overview

The tool is divided into four main components, Controlled Random Scheduler, Group Membership, Storage and an optional Bootstrap component as shown in the Figure 7.1. The bootstrap component is started only if the tool is started in bootstrap node. It is responsible for handling new nodes which requests to join the group. The Control Random Scheduler component is responsible for handshaking with other nodes to ensure their availability to participate in a measurement or not. It is also responsible to carry out the actual measurement by using any third party measurement tool (e.g BART Tool). The Group Membership component is responsible for maintaining view of nodes which join and leave the group. This component ensures a each node to know about all

\(^1\)None of the machines are behind Network Address Table (NAT) or firewalls
other nodes in the system, so that they can schedule to trigger measurements. And the Storage component is basically responsible for storing the measured data locally.

7.2 Implementation Details

The packages and some of the important class descriptions are given in the following sections.

7.2.1 Packages Descriptions

Here are the list of packages and their descriptions:

- **com.ericsson.sntt.config** package contains classes to hold configurations and constants for the whole project.

- **com.ericsson.sntt.core** package contains classes which does the actually scheduling, triggers measurement and stores the measured data.

- **com.ericsson.sntt.exception** package contains custom exception classes.

- **com.ericsson.sntt.gui** package contains classes which are responsible for the graphical user interface (GUI) of the project.

- **com.ericsson.sntt.log** package contains classes for managing the logs generated by the project. The classes here are mainly wrapper classes for the Log4j library.\(^2\)

\(^2\)Log4j is a open source logging library
• com.ericsson.sntt.messages package contains classes which are used for messages passing between nodes.

• com.ericsson.sntt.utils package contains classes which has utility functions.

### 7.2.2 Class Descriptions

Here are some of the important classes and their descriptions:

- **BootstrapServer**
  This implements a server which is running on a well established IP and Port. Nodes send JOIN_REQ to it, to become part of the network.

- **HandshakeServer**
  This implements a server running on IP and port. This IP and Port is the unique identifier for the node. The server listens for HANDSHAKE messages in order to negotiate if it can participate in a measurement or not. It receives the handshake messages and passes it on to the HandshakeConnection class for further processing.

- **HandshakeConnection**
  This is mainly responsible for checking what message is received and what actions need to be performed based on the received message.

- **SNTTCore**
  This class is the entry point. It parses the command line arguments to see if it has to start as a bootstrap node or a normal node. This starts the handshake server and if it is a bootstrap node, then it also starts the bootstrap server.

- **SNTTViewer**
  This class implements the GUI components and event handling.
**ExecuteProcess**  This class is a helper utility class which can run other programs in shell from within this project and pipe its output as needed. This is used to run BART tool.

**HandshakeMessage**  This is a message class. It can piggyback AL, TL, and TAL lists. This message class has opCode (e.g. OP_HANDSHAKE_REQ, OP_JOIN_REQ, etc) by which each message type is identified.

### 7.3 Execution Process

The tool is distributed as executable jar and is packed in a folder named 'dist'. The 'dist' folder structure is as follows:

```
|-- dist
  |-- lib
  |-- sntt.properties
  |-- SNTT.jar
  |-- README.TXT
```

The 'lib' folder contains the necessary third party libraries. The sntt.properties is the properties file for the project. The SNTT.jar is the main executable file. And README.TXT contains basic information.

The sntt.properties file can be edited to set the properties of the node. The sntt.properties file has the following configurable information in it:

```plaintext
# This is to set how many concurrent measurements the node can participate in.
maxActive 1

# This sets the percentage of desired M nodes in the system
PercentageOfMNode 30

# This sets the TL and AL list size
listSize 5
```
# This are the Bootstrap Node IP and Port
bootStrapNodeIp 192.36.157.58
bootStrapNodePort 4001

The nodes can be started from the shell. To start the node as a bootstrap node:

```
> java -jar SNTT.jar -b
```

To start as a normal node:

```
> java -jar SNTT.jar
```

## 7.4 Summary

The network tomography tool is implemented in Java, which currently does bandwidth measurements using the BART tool. The entry point to the project is through the 'SNTTCore' class. It is responsible for command line argument parsing and also starts the 'HandshakeServer' which listens to incoming messages from other nodes. The 'SNTTCore' class also starts the 'BootstrapServer' if the tool is started as a bootstrap node. The 'HandshakeConnection' class is responsible for both the processing of handshakes messages to check for availability of nodes to participate in a measurement and extract the piggybacked data in the handshake messages in order to update the group membership view of the system.
Chapter 8

Conclusion

Network Tomography system continuously probes for measurements using only end-to-end probes as opposed to making measurements on every internal link in the network. It extract metrics like link loss rates, delay. Further in this thesis work it is shown how bandwidth can be measured effectively using network tomography in a decentralized system. A full mesh measurement can take considerable time, but it depends on the amount of load injected in the network and also load per node that is allowable. It has been observed that time is the constrain for such systems as when it takes longer time to do full mesh measurements, the measured data might not be of interest anymore. However, there are applications where a few hours of time needed to measure a network of 100 nodes is acceptable. The decentralized approach eliminates single point of failure but it also introduces more overhead in terms of maintaining the network. They are also scalable. Despite the benefits of a decentralized Network Tomography system, the network operators however have not yet appreciated the benefits of peer-to-peer monitoring systems for two reasons; firstly the hype that peer-to-peer systems are used for illegal things and secondly the research on this topic is far from complete.

8.1 Summary of Work

The thesis work did an extensive study on field of active network tomography system and documented the current state of the art. The challenges of making a distributed control system to do active network measurements were investigated and the research challenges were
identified. Further the additional constrain posed by doing bandwidth measurement were taken into consideration and a complete distributed active network bandwidth measurement system was proposed and evaluated by performing simulations. A complete running prototype of an active network tomography tool for doing bandwidth measurement was implemented as well.

8.2 Research Findings

The research questions stated in Section 1.3 which were posted at the start of the thesis are summarized below with the findings of the thesis work.

RQ-01 : If operators want to deploy bandwidth measurement methods, there is a need to control the induced overhead. How to achieve control over the network load posted by the network tomography tool?

Findings A network tomography system is proposed in this thesis work which can control the amount of ongoing measurements at any time, by controlling the amount of nodes which can trigger measurements. Thus enabling network operators to have a high degree of control over the network load injected by the network tomography tool.

RQ-02 : How to make a scalable network tomography tool, specially when doing bandwidth measurements as they are considered to overload a bit more compared to measuring other metrics?

Findings In order to achieve scalability, the network tomography tool has configurable parameter on the amount of load each node can process, and also on the amount of load the tool can put on the network. This parameters let any operator tune the tool according to their needs. The proposed network tomography tool is a decentralized system and thus also helps the system to scale well.
8.3. **FUTURE WORK**

**RQ-03:** How to automate monitoring so that minimum or no supervision at all would be required while the tool will keep probing for measurements?

Findings The thesis work addressed this problem and came out with a simple control mechanism applying probabilistic scheduling. This probabilistic scheduling enables the system to run continuously covering full mesh coverage without any operator supervision.

**RQ-04:** Is it feasible to use peer-to-peer based monitoring systems as an alternate to traditional centralized solutions?

Findings In this thesis work it is proved that a peer-to-peer based monitoring system works with reasonably good performance (full mesh coverage possible and minimum traffic injections).

**RQ-05:** What are the benefits of using a peer-to-peer monitoring systems?

Findings One of the main advantages of using a peer-to-peer based monitoring over centralized system, is resilience to failures. That is any node can crash, but the system will keep working. Other advantages include distributed data storage, inherent replication factor of 2 and distributed control.

### 8.3 Future Work

The thesis work laid a platform to carry out more research work on the field of active network tomography.

"*It is a bad plan that admits of no modification*" - Publilius Syrus

The functionality of the proposed system is already simulated in a simulator, thus the further simulations can be done varying the
configurable parameters in order to observe how the system behaves. An
active network tomography tool has also been developed, which can be
used to further study the property of the system and how it performs in
live networks. The next natural directions for the research work are as
follows:

- Further simulation to gain a even better view of the behavior and
  performance of the system.

- Deploy the tool in Planet Lab, or any other live network, to see how
  it behaves in real environment.

- Since it is a P2P system, if two nodes are behind firewalls or Network
  Address Table (NAT), they cannot establish a connection between
  themselves; hence need to implement NAT traversal.

- Plug in more measurement tools to the system, so that the system
  can do other measurements apart from the bandwidth measurements
  using BART.

- Study security concerns of storing data in a decentralized storage to
  avoid measurement data might leakage.

- Study and implement ways to authenticate nodes in the network.
Bibliography


Appendices
Appendix A

Screen Shots
Figure A.1: Screen shot - Bootstrap Node

Figure A.2: Screen shot - Measurement Node