UNIVERSITY OF OSLO
Department of Informatics

Link Delay Inference in ANA Network

Ebenezer Paintsil

Network and System Administration

Oslo University College

May 28, 2008

Link Delay Inference in ANA Network

Ebenezer Paintsil

Network and System Administration
Oslo University College

May 28, 2008

Abstract

Estimating quality of service (QoS) parameters such as link delay distribution from the end-to-end delay of a multicast tree topology in network tomography cannot be achieved without multicast probing techniques or designing unicast probing packets that mimic the characteristics of multicast probing packets. Active probing is gradually giving way to passive measurement techniques. With the emergence of next generation networks such as Autonomic Network Architecture (ANA) network, which do not support active probing, a new way of thinking is required to provide network tomography support for such networks. This thesis is about investigating the possible solution to such problem in network tomography. Two approaches, queue model and adaptive learning model were implemented to minimize the uncertainty in the end-to-end delay measurements from passive data source so that we could obtain end-to-end delay measurements that exhibit the characteristics of unicast or multicast probing packets. The result shows that the adaptive learning model performs better than the queue model. In spite of its good performance against the queue model, it fails to outperform the unicast model. Overall, the correlation between the adaptive learning model and multicast probing model is quite weak when the traffic intensity is low and strong when the traffic intensity is high. The adaptive model may be susceptible to low traffic. In general, this thesis is a paradigm shift from the investigation of "deconvolution" algorithms that uncover link delay distributions to how to estimate link delay distributions without active probing.

Acknowlegements

There are special people who are behind the success of this work and deserve my appreciation and recognition. I acknowledge the support of my advisor Prof. Demissie Aredo and co-advisor Professor Mark Burgess. Their support was very helpful to see me through this undertaking. I owe special gratitude to Professor Mark Burgess for his dedication to this work and the promptness with which all my difficult questions were answered.

I am also grateful to the rest of my lecturers who gave me the foundation to be able to carry out this work. I am thankful to Kyrre Begnum, Ismail Hassan and others who in diverse ways helped to make this a success.

I acknowledge the good company of my fellow classmates from various part of the world in this work. Your presence made me experienced the dynamics of the whole world. I am thankful for your support and amazing friendship, the two Stiens, Lu,Iman, Saedar, Jeang, Marius and all the others especially Kashif.

Lastly, I want to thank my family (Annabella, Sela (wife) and others) for their amazing support. Their support for me has been unconditionally in every way and at every turn. Also I thank Abraham, Liz for being good friends.

Contents

List of Figures

Acronyms

ANA Autonomic Network Architecture

IC Information Channel

IDP Information Dispatch Point

FB Functional Block

cfEngine Configuration Engine

SLA Service Level Agreement

TCP/IP Transmission Control Protocol/Internet Protocol

MANet Mobile Ad hoc Network

SNMP Simple Network Management Protocol

NAT Network Address Translation

ISP Internet Service Provider

ICMP Internet Control Message Protocol

MCMC Markov chain Monte Carlo

OGC Office of Government Commerce

ITIL Information Technology Infrastructure Library

TTL Time To Live

AI Artificial Intelligent

OSPF Open Shortest Path First

Wi-Fi Wireless Fidelity

TCL Tool Command Language

OTCL Object Tool Command Language

I.I.D Independent and Identically Distributed

List of Tables

Chapter 1

Introduction

Transmission Control Protocol/Internet Protocol (TCP/IP) suite has become the de-facto standard for interconnecting dissimilar networks, hosts and the Internet because of its simplicity and power [2]. However, the typical requirements of TCP/IP networks such as static nodes, fixed topology and wired links makes it less adequate to the needs of contemporary networks. The current Internet architecture is insufficient for the future, since problems such as mobility and non-universal connectivity are already with us [3]. The internet architecture was created with a naive view of what lies ahead in the future. The use of the Internet to provide diverse services with competing objectives have changed the original design goal of a research network to a recognized component of mainstream society [4] and therefore suggests the need for new design principles.

In addition, the emergence of new applications on the Internet have led to the expansion of the Internet at some levels of the architecture. Over the years, the Internet has evolved through vertical and horizontal scalability at some levels of the architecture. More and more applications and protocols are being built to run on top of TCP/IP protocol suite with little regards for the inherent defects and requirements of the TCP/IP protocol suite. TCP protocol requires trusted receiver in order to prevent congestion [5]. This means both the receiver and sender clocks ought to be synchronized. Thus, the default Internet architecture depends on a number of principles including trust, making it vulnerable to attack.

It is widely accepted that the Internet has achieved remarkable success over the years [1] in cases where the typical Internet topology has largely been maintained. This success is at the expense of scalability at all levels of the architecture and flexible network topology or type.

On the contrary, in cases where typical Internet topology or type has changed the result is not encouraging [6]. The emerging networks such as Mobile Ad hoc Network (MANet), sensor network, delay tolerant networks, peer-to-peer networks and

wireless networks perform poorly when connected to the Internet. This is partly due to the limitations of one-size-fits-all TCP/IP protocol suite, as the network layer of the Internet has not aimed at functional scalability by design but has evolved through a 'patchwork' style. The current Internet is end-to-end where network nodes mainly perform forwarding functions and much of the logic is done at the end node. With the introduction of multimedia capabilities in the Internet the end-to-end argument may need reexamination.

Scalability is a watchword in today's computer world. Almost everything is being developed to scale both in functionality and in size. The challenges of scalability is how to manage the complexities that are associated with it. The Internet is scaling in size at an astronomical rate with increasing complexity.

Today's Internet provides diverse services with competing objectives, including educational, social, scientific and corporate objectives. It is the overlapping web of complex infrastructure running diverse applications with diverse quality of service requirements. Different application-generated data flows require different network characteristics. This brings to the fore the necessity for automatic network management.

The growing complexity has necessitated the need for high skilled professionals to maintain and keep the Internet running. Since companies cannot afford to hire such skilled workers required to manage such complexities, they have increasingly resorted to offshore outsourcing and commoditization of system administration in recent years to save labor costs [7].

Complexities gives rise to security concern. Securing computer networks is becoming increasingly difficult and expensive. All these explains why self-healing, self-managing, self-organizing, self-configuring, self-optimizing, self-protecting and self-federating network otherwise known as autonomic network is needed today.

The problems with the Internet support the call for a paradigm shift from a network that scales only vertically to a network that scale both vertically and horizontally, and capable of achieving 'autonomicity'.

1.1 Network Monitoring In General

Typical network monitoring system requires human to decide when to trigger a system that constantly monitors a computer network to identify slow or failing elements and to notify the network administrator in case of failure via email, pager or in another way.

The basic monitoring processes include data collection, data processing, decision making and notification. Until recently, these processes were regarded as static processes which requires human intervention in order to trigger and interpret the monitor-

ing results for decision making. As the systems scale in size and requirements (such as quality of service management) their complexities are also growing at astronomical rate. The need to reduce human intervention has no doubt taken the center stage of research. Besides, the shortage of skilled workers to handle system complexities makes it appropriate to research into reduced human intervention in monitoring [7] and system management as a whole.

Static monitoring requires continual monitoring, usually, without the necessary scheme to manage when monitoring is triggered. It operates on large amount of resource. On-demand monitoring provides mechanisms to dynamically monitor a system when an important event occurs or when necessary. This is necessary to ensure effective and efficient resource management. To further meet the network monitoring challenges of contemporary times such as reduced human intervention, has ignited the need for a system that exhibit self-monitoring capability and eventual attempt towards autonomicity.

Monitoring can be done by using measurements taken from inside the network or at the end-points. Monitoring from the network usually uses data source from NetFlow or SNMP (Simple Network Management Protocol). There are two ways of performing the network monitoring: *active* and *passive* measurements. Active monitoring requires the introduction of additional packets known as probing packets into the network in order to obtain measurements. This could perturb or introduce additional overheads into the network.

On the other hand, passive network monitoring or measurement requires no additional traffic or probing packets and it does not perturb the target network. It gives more detailed information about the network being measured, and can detect active network elements and their properties. It relies on the data gathered through regular operation to monitor.

1.1.1 Challenges of Monitoring

Future self-monitoring systems will be affected by challenges such as the choice between active probing and passive measurement approach, issue of resource management, reduction or elimination of human intervention, centralized or distributed monitoring systems, voluntary cooperation or obligation approach, the kind of network architecture, protocol support, accuracy and computation load, data sampling schemes and many more.

Active measurement or probing may not be able to measure the behavior of network elements behind firewall and NAT devices. Moreover, active network measurement gives less detailed information about the network [8].

Passive monitoring depends on the data collected at the monitoring point. If a

device does not send data through the monitoring point, no conclusion could be drawn about that device. Passive monitoring can neither detect idle nodes or services nor gather information when packets are encrypted. It has no control over the type of traffic that passes through the monitoring point. Passive network monitoring is vulnerable to abuse by worms. It depends on time to live, which could be changed to fool the monitoring system. Autonomic monitoring system should have a scheme to automate the selection of sensor node to ensure adequate coverage.

Current research shows that multicast based active measurement is quite successful [9, 10] in terms of the amount of load introduced into the network, accuracy of result and node support. In a network with **N** nodes, the load in the network grows proportional to N^2 when using unicast packets to infer as against N when using multicast packets. Multicast traffic is bandwidth efficient and makes it suitable for large-scale measurements of both end-to-end and internal network dynamics. The major drawback of multicast traffic is the issue of multicast protocol support.

Future autonomic network blue prints such as autonomic network architecture (ANA) supports multicast protocol. However, current Internet does not fully support multicast protocol hence the implementation of multicast monitoring techniques may be impossible on a large scale.

Self-managing systems are supposed to be on-demand and in-built. On-Demand Monitoring (ODM) requires that monitoring is triggered when necessary and resources are freed when it is not in use. Self-managing also includes new monitoring requirements such as online or realtime decision making, execution of decision, management element which would otherwise be handled by human off-line. In addition, the monitoring device needs to perform its regular functions. Additional requirements means additional resource requirements. More CPU, memory etc will be required for monitoring. An attempt to manage resources will coincide with computational accuracy and speed of decision making. While sampling techniques used by data stream management systems (DSMS) may be appropriate in self-monitoring systems more research is required to minimize the trade off between accuracy and sampling techniques.

Self-monitoring system could be centralized or distributed. A centralized system puts the monitoring burden on a central node which decides when to trigger monitoring event on other nodes or request for monitoring information. In ANA, such a node is called decision maker. At the compartment level, the decision makers issue data collection requests to collect the data and events from the data collection point (sensing node). The data collected is pushed to the decision maker for decision to be taken and send for execution. This processes are recursive. Though there are several decision makers involve in the monitoring and the burden of monitoring is subdivided it may depict the idea of obligation or central authority which is frown upon by Promise Theory. In ad-hoc network environment however, entities cannot achieve self-monitoring from centralized monitoring [11].

Distributed monitoring systems allow entities or systems to trigger monitoring task on each other but not only the decision maker. Monitoring data is distributed among the entities without undue reliance on a single node or a group of nodes. Distributed monitoring system will suffer from high security requirements and better clock synchronization.

The idea of voluntary cooperation and obligation is similar to centralization and distribution. The main cooperative approaches to system design are obligation and voluntary cooperation [12]. The idea of voluntary cooperation is believed to provide a better way of designing loosely coupled and scalable systems than obligation. Voluntary cooperation suggests that, a collection of discrete agents which interact with each other on their own volution (depending on their individual policies) provides a better decoupled and scalable system than those that depend on another agent (supper, master, access objects or central authority) to grant access before interaction or communication could take place (obligation system).

1.2 Motivation and Research Questions

Network monitoring will be a key concept in achieving autonomicity in ANA network. The self-management capability of ANA network depends solely on effective monitoring system [1]. For example, Information Dispatch Point will be able to forward network traffic from busy link to less busy link or through less expensive ISP from more expensive ISP through effective and efficient network monitoring. It may need loss rate or the queuing delay on each link in order to estimate the suitable route. The accuracy and speed in which the data is processed to get results will be very vital in autonomic decision making.

Besides autonomicity, ANA network is a service oriented network. The quality of service guarantees requirements of service oriented applications or realtime applications such as voice over IP, demand efficient network monitoring. Large amount of resources are reserved in advance for such applications leaving the rest for grasp by other applications. In order not to violate Service Level Agreement (SLA), constant network monitoring is needed to ensure adequate service provisioning.

ANA advocates for passive network monitoring as the only means by which the network can be monitored in spite of its numerous limitations (see section 1.1). ANA network totally eliminates active network monitoring using network architecture as one of the main reasons [1]. The other reason is to reduce the load on a single router (node) by seeking the cooperation of other nodes in network monitoring.

It identifies that current approaches to network measurement either has minimal nodes and protocol support as in active measurement or places the entire burden on few nodes or routers (SNMP or NetFlow data collection) as in passive measurement. On

the contrary, current research shows that multicast based active monitoring technique is quite successful[9, 10] in terms of the amount of load introduced into the network, accuracy of result and node support.

Multicast traffic is bandwidth efficient and makes it suitable for large-scale measurements of both end-to-end and internal network dynamics. The major drawback is the issue of multicast protocol support. Multicast protocol support is no issue in ANA network since it fully supports multicast protocol. One of such network monitoring technique that supports multicast protocol is network tomography. Though it suffers from computational load, recent unified pseudo-likelihood approach has been found to reduce the computational load in network tomography and yet produces accurate result [4].

Although ANA network supports multicast protocol, its monitoring system does not support active probing. Unicast approach may still base on active probing and has demerit of exponentially increasing congestion in the network. It also suffers from scalability issues when large network is of interest [13]. It is possible to estimate the origin-destination matrix with passive measurement but the method does not base on cooperation of nodes which is one of the cornerstones of ANA network monitoring. This means the current network tomography inferring approach may need rethinking to work with passive monitoring networks such as ANA network.

Consequently, it is significant to ask the following questions:

- **can network tomography be made to satisfy the requirements of ANA monitoring system in estimation of internal link delay?**
  - **if yes, in what way can this be done?**
  - **what would be the accuracy of the passive network tomography technique as compared to active probing techniques?**

Network Tomography is able to estimate the origin-destination (OD) matrix which is a key input to many routing algorithms such as Open Shortest Path First (OSPF) protocol. It is also able to estimate link delay which is one of the major indicator of network performance [14]. ANA network therefore cannot afford to forgo this powerful concept. Exploring passive approach to network tomography will make the implementation of network tomography possible in ANA or at least open the way for extensive discussion. It will also lead the way for better passive alternative to network tomography in IP network.

Chapter 2

Background and Theory

2.1 Building blocks of ANA

Autonomic Network Architecture (ANA) is a framework for the implementation of future autonomic network that will have both vertical and horizontal functional scalability at all levels of the architecture. Horizontal scalability means the network is able to extend to include more functionalities and vertical scalability means the network allows different ways of including abundant functionalities [15]. In addition ANA network will have autonomic capabilities.

ANA network is not about reinventing the wheels but an attempt to reorganize the existing technology to meet the demands of the contemporary and next generation networks instead of wallowing in perpetual patchwork approach to solve network problems which sometimes fail to meet expectation.

ANA organizes networks into compartments. A network compartment is similar to a mobile phone network cell which allows registered subscribers to enjoy the services of the network provider. Unlike cells, compartments have local addressing scheme with boundaries determined by the kind of communication technology or communication element it represents. Compartment is defined in [1] as "*a policed set of Functional Blocks (FBs), Information Dispatch Points (IDPs) and Information Channels (ICs), which enables communication for its members according to some commonly agreed set of communication principles, protocols and policies* ". Elements of a compartment can be grouped into policies and operational rules. A compartment implements these policies and operational rules using functional blocks. ANA compartment can represent almost every communication element such as the entire communication stack (e.g TCP/IP communication stack), a wireless network, functionalities of one or more network layers and so on. These elements also set the boundary of a compartment. For example, the boundary of a compartment may be determined by the kind of

technology it supports such as IPv6 protocol compartment, wireless or wifi compartment, Ethernet compartment or the policies it supports. Compartments are backward compatible, ie they support past communication technologies.

Some of the differences between a compartment and TCP/IP communication stack is that the layers of compartment stack are loosely coupled and have local addressing scheme. That is, a compartment need not necessarily implement functionalities of a whole communication stack with layer Y+1 on top of layer Y but parts which could be combined with other partial communication stack compartments to achieve complete communication stack. Compartments do not necessarily operate end-to-end communication as in TCP/IP network. That is compartments can be found in any part of the network (end nodes or intermediate nodes) with the same or similar operational capabilities.

Elements of compartments are Functional Blocks (FBs), Information Dispatch Point (IDP) and Information Channel (IC). These building blocks operate in accordance with the policies governing the compartment. Compartments are complete autonomous or discrete entities with no central control, they are governed by their own set of policies, have their own internal addressing scheme, protocols etc. Compartments are also found in ANA nodes, this is called node compartments. Node compartments have all the functionalities as regular network compartment.

A Functional Block (FB) is a loosely coupled information processing functions which can generate, process, forward or consume information. For example, a functional block may add protocol header or perform checksum operation on the protocol header, a whole communication stack and so on. Functional blocks could be seen as services offered by the compartment.

ICs are represented by two forms of abstractions - imported ICs (physical) and exported ICs (logical). The imported ICs are the abstraction of the services provided by the underlying system to the compartment. They are not accessible to external entities. Exported ICs are the abstraction of the communication services provided by the compartment or the clients of the compartment to the outside world for possible members of the compartment to subscribe. These services include membership registration services, which check if a node conforms to the policy of the compartment before it is allowed to register or join the compartment. The imported ICs can be the abstraction of real communication channel such as wireless medium and wired cable that carries the message from one node or compartment to another. It can also be the abstraction of operating system functions that support the compartment or the ANA software.

Information Channel (IC) can be unicast, multicast, anycast or concast. Unicast IC is similar to regular Internet traffic flow where a packet is sent from one node to one and only one node. Unicast IC in ANA allows message to be sent from one node to one and only one node. In ANA network a node could be a client, a module or a router. Multicast IC allows message to be sent from one ANA node to a group of

nodes. Anycast allows message to be sent to a group of node but the node which is closest to the source will receive on behave of the group. Concast is the opposite of multicast, which implies multiple senders to one receiver.

In order to increase the flexibility of ICs and FBs, Information Dispatch Point (IDP) is introduced to serve as an entry point to IC or FB. This idea is similar to mobile IPv6 where location is decomposed from address to ensure flexibility and client mobility. In ANA IDPs are needed to ensure dynamic rebinding. IDPs are also functional blocks with autonomic binding ability. It is simply a binding element and serves as the entry point to an IC or FB. A compartment has three views - exported view, structure view and imported view. The exported view consists of the services the compartment and its possible clients can offer to the external world. An exported service can become imported service of another compartment. The external clients see the compartment as a set of IDPs and logical ICs by which they can communicate with. Imported view represents the abstraction of the services offered by the underlying system (such as the operating system, physical medium which are not the functionalities of the compartment) which host or support the ANA compartment. It represents the abstraction of the low-level functionalities of the supporting systems. ANA software will be installed on an operating system which will provide support for the ANA software by exposing its internal functionalities in the imported view of ANA compartment. The structural view shows the functional blocks that perform the communication operations such as routing in the compartment (see figure 2.1).

Membership registration is a key concept in ANA network (see figure 2.2). Registration of a node **A** means checking if the node conform to the policy of the compartment and assigning a label to it, it is similar to assigning IP address to a node in IP network, however, ANA labels are local and have no global reach. ANA label can be any identifier such as MAC address, IP address etc. Compartment registration can be open or close. Close registration requires authentication before registration is allowed. In open registration, a host which conforms to the policy of the compartment is allowed to join. Equally, a compartment provides host deregistration functionalities for members that leave the compartment.

Communication begins with a certain node requesting the resolution of a member **A** say (see figure 2.2). This is equivalent to performing routing table lookup to find the path to a certain node **A** (as in routing tables it is possible to find more than one path to A). The end result of the resolution process is a label **a** to identify the IDP of the path or IC to A. That is, **a** identifies the entry point to the path (IC) to node **A**. Communication could occur directly between nodes using a labeled IC without IDP. However, the problem with this is that it would not be flexible. Suppose that there are two routes or paths (ICs) to node **A**. If communication were allowed to happen between the nodes and the initial IC breaks down you may not be able to forward packets through the second route without delay. For this reason, ANA attaches Information Dispatch Point (IDP) to the IC to make the communication autonomic with the ability to forward the

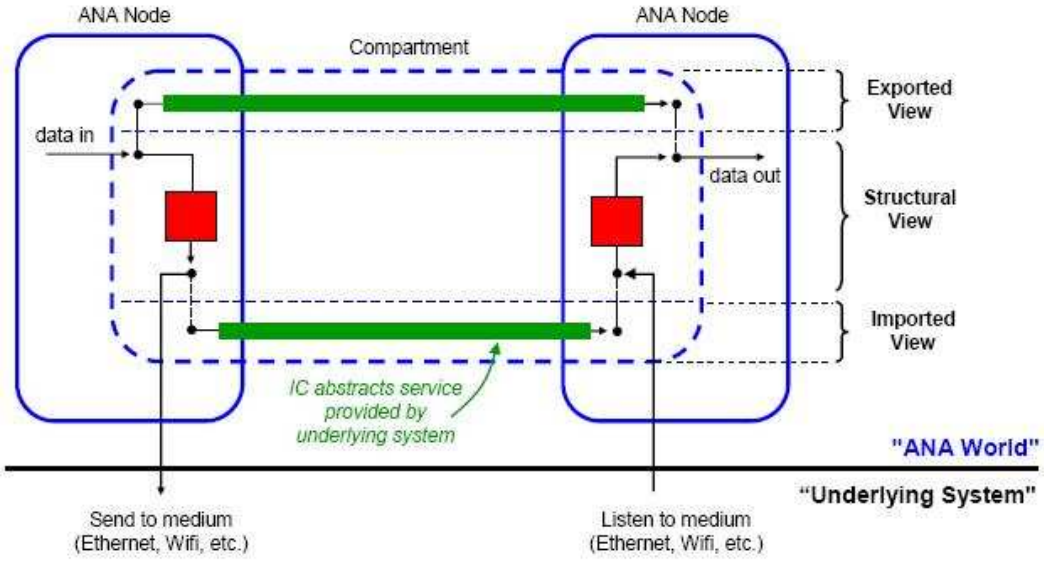

Figure 2.1: *figure 2.1 different views of ANA compartment, the red square blocks are the FB, there is only one FB in this diagram but in practice there could be more FBs in a single compartment. The solid green bars are the logical ICs. The bottom logical IC represent the chain of functions or services provided by the underlying system (OS). The top logical IC represent the chain of services or functionalities provided by the compartment to the outside world. The solid black lines represents he physical IC or medium through which information travels within the compartment. The solid black dot represents IDP. [1]*

packets through more suitable path in case the initial IC or path encounters problems. IDP is also attached to a functional block to perform similar functions. Considering figure 2.2 again, IDP **a** is the entry point to IC to node **A**. IDP and the IC are decoupled.

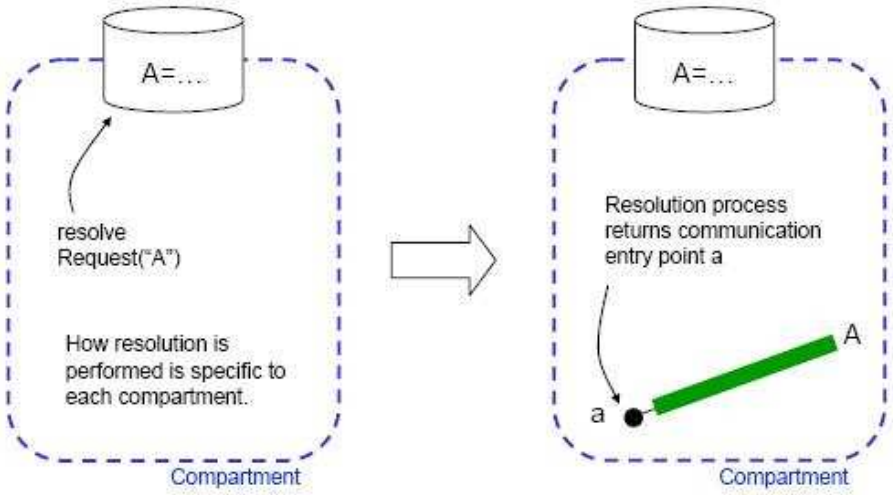

Figure 2.2: *Internal resolution of compartment member*[1]

2.1.1 Communication Between Clients

Each client has membership database in its node compartment, which keeps information about the accessible network compartment and the services offered by both the network compartment and other clients who are connected to the compartment. It also contains FBs of ANA software in particular encryption, data compression and transmission rate control FBs. The access object to the network compartment is located outside this database. The information in the database is access via node compartment access object or access functional block. Figure 2.3 shows how ANA nodes are organized to communicate over ethernet compartment. Compartments have all the routing protocols and the functionalities needed to communicate using ethernet protocol.

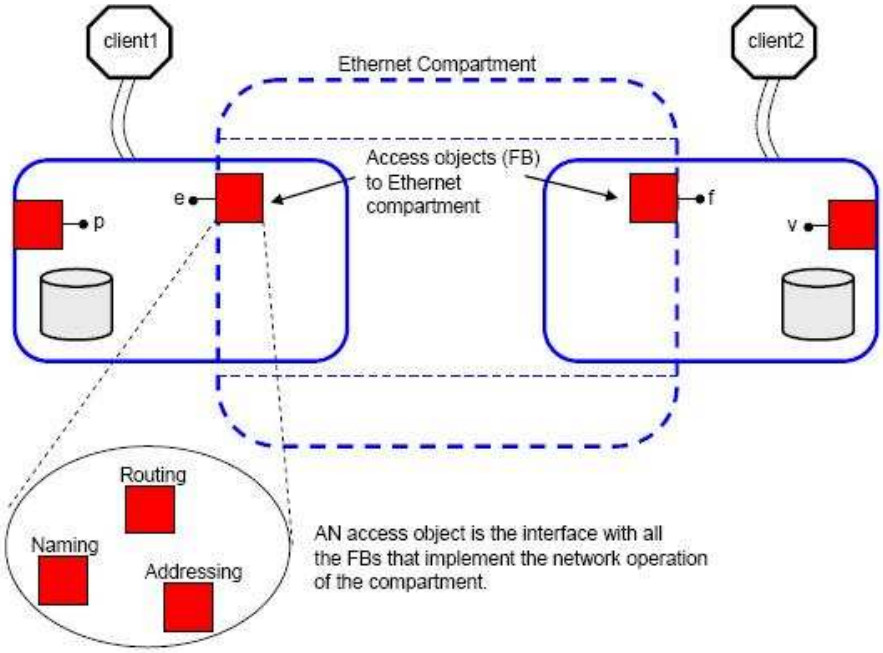

Figure 2.3: *Client communication*[1]

Figure 2.4 shows the rest of the communication. The node compartment access object for *client1* and *client2* can be reached through IDP labeled **p** and **v** respectively. Similar to the node compartment, the network compartment also has access object. For the two clients (*client1 and client2*) to access the network compartment, they need to call or instantiate the network compartment access object via their respective node compartment access objects (through IDP **p** and **v** respectively). This returns IDP **e** for client1 and IDP **f** for client2. This means the network compartment can be reached through IDPs **e** and **f** by client1 and client2 respectively. In practice, there could be the same network compartment access object for both clients. After these processes the two clients now know the IDPs to network compartment. The next step is that both clients have to be visible in the external view of the ethernet compartment. Both clients request their respective network compartment access object to make them visible in the network compartment. This is done by registering both clients in the network compartment's database. *client1* registers with an identifier **A** and *client2* registers with an identifier B. The B in the database means *client2* is reachable via a certain identifier **B**. A functional block with IDP **b** which will process the incoming data for **B** is instantiated and is mapped in the external view of the network compartment to enable other clients to reference it. The same is repeated for *client1*. The two FBs instantiated as a result of the registration are connected by a physical link or medium which is abstracted in the imported view of the compartment represented by IDPs **s** and **i**. The IC **s;i** is the physical abstraction with no ANA functionality but IC **a;b** has ANA functionalities. With this setup, the two clients can now communicate by using their

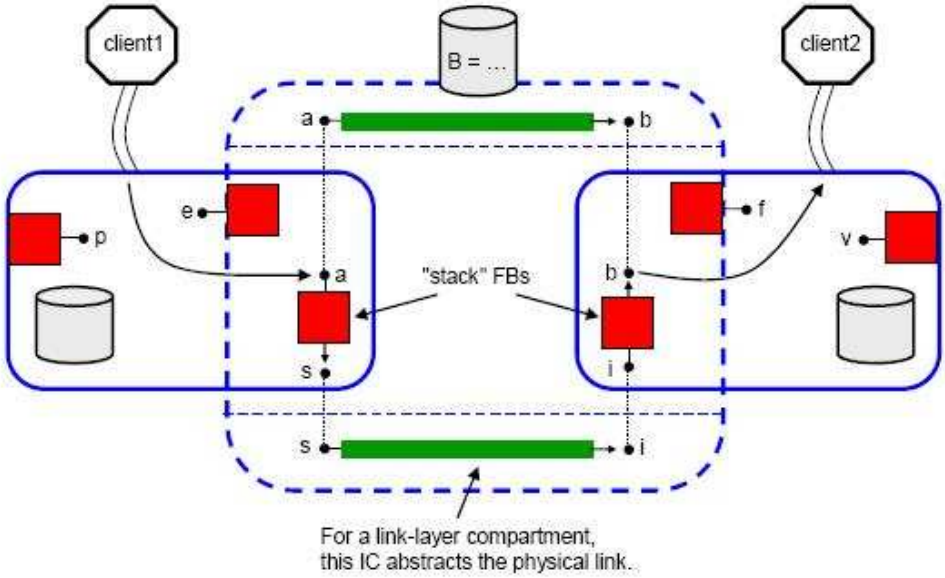

Figure 2.4: Client communication [1]

respective external IDPs as shown in figure 2.4.

2.2 Autonomic theory

The quest for self-governing computing systems has long been the focus of research. Since the days of object oriented concepts, the idea of self-governing computing entities has been extended even further to embrace the contemporary demand for autonomic systems. The idea of autonomic computing inspired by the operations of autonomic nerve system in human may be rather a simplistic view of autonomic computing [16]. The scope of autonomic computing specification by IBM is wide, and it includes everything, ranging from host, network, software to algorithms. In spite of this wide scope, the autonomic computing specification given by IBM focuses more on minimizing the interventions of system administrators in day-to-day running of software systems to produce software that is able to patch themselves, automatically seek updates, and better configurations that will give them better performance [7].

Like the evolution of telephone switchboard in 1920s there were serious concerns whether there would be enough operators to man the manual switchboards. Analysts predicted that if growth continues by 1980s, half of the population in USA would have to work as telephone operators to meet the demand [17]. Similarly, the demand for IT workers is expected to increase by over 100% in the next five years in the USA. The

current trend of using administrator to troubleshoot and fix almost every IT problem is quite manual and needs a paradigm shift.

IBM defined the following four self * attributes to describe autonomic computing

- Self-healing: the ability to discover and repair potential problems to ensure that the system runs smoothly;
- Self-protection: the ability to identify threats and take protective actions;
- Self-configuration: the ability to install and set up applications/patches/updates automatically, verify compliance with the specified service levels, optimize configuration of applications using adaptive algorithms;
- Self-optimisation: the ability to monitor predefined system goals and performance levels to ensure that all systems are running at optimum levels [16, 18].

These specifications do not explicitly capture certain requirements of autonomic systems such as the ability to evolve, adaptability, dynamic protocol switching (in autonomic networks) and so on. Shifting the focus from autonomic computing to autonomic systems would put more emphasis on systems such as host, network or complete infrastructure instead of computing elements. This would set the stage for a more focused research into autonomic computing.

Up to now, what is autonomic system is still not clear and whether autonomic systems already exist or not is controversial. There are many systems that implement certain attributes of autonomic systems but not all. An important issue is, how many autonomic attributes must a system exhibit in order to be classified as an autonomic system ? Should a system exhibit all autonomic attributes or some of them in order to be classified as an autonomic system. Currently, there are systems such as configuration engine (cfEengine) which exhibit some attributes of autonomic systems yet they may not be fully autonomic by definition. What about the controversy of resilience versus hibernation or self-destructive systems? Resilience in an autonomic system requires a system to be robust to attack or should be always on. Certain systems self-destruct or hibernate to conserve energy under critical or certain conditions. Such systems could not be described as autonomic since they fail to stay on all the time.

Should autonomic system be similar to a particular biological system? Computer Immunology [19] one of the early papers that suggested autonomic computing agrees that autonomic system maintenance should be similar to biological immune system. Such immune system could detect problem conditions and mobilize resources to deal with the problem automatically. All these examples are numerous attempts to define autonomic system.

Another question is, should autonomic system have the same capabilities of Artificial Intelligent (AI) ? The answer is no. Autonomic systems cannot be said to be the

same as Artificial Intelligent (AI) but it includes certain properties of AI [20].

Clearly, it may not be possible to have a system which exhibits all these numerous autonomic attributes. Hence autonomic system should implement as many autonomic attributes as possible.

ANA network classifies autonomic capabilities into three groups, namely, self-management, network management and resilience. Self-management consists of self-configuration, self-optimization, self-healing, and self-protection. This is achieved through autonomic network monitoring component (see figure 2.5). Autonomic network management includes Fault-management, Configuration-management, Accounting-management, Performance-management and Security-management. Resilience requires self-organizing, self-learning, auto-configuration, self-managing, self-diagnosing, self-repair and self-optimizing [1].

2.3 ANA Self-Management Concepts

Information gathering and analysis is key to achieving autonomicity in ANA. Configuration information, signalling information and monitoring information are key information needed to achieve autonomicity. Information is required to achieve self-management capabilities. Self-management is achieved through continual or on-demand monitoring, decision making and execution of the decision (see figure 2.5).

The decision makers issue data collection requests to collect the data and events from the data collection point (FB or sensing node) in a node or a compartment. The data collected is pushed to the decision maker for decision to be taken and sent for execution. This processes are recursive. Monitoring is the element responsible for measuring the parameters such as current load for performance optimization, link availability for fault tolerance, etc. needed for decision making [1].

Monitoring in ANA network is dynamic and adaptive implemented as generic functional block and instantiated when required. If additional rules are required for anomaly detection, the rules database can be updated ”on-the-fly”. Each functional block exposes part of it internal information which is collected and assemble for decision making. The decision maker (decision) can trigger monitoring on a functional block or a node when needed.

There are three levels of monitoring namely, node level monitoring, compartment level monitoring and inter-compartment level monitoring. Node level monitoring is executed by the collaboration of functional blocks. The monitoring operations are the same as described above but the focus of the monitoring is on the functional blocks and system parameters. It is an example of system monitoring. The decision maker, execution and monitoring are all represented by functional blocks within the node.

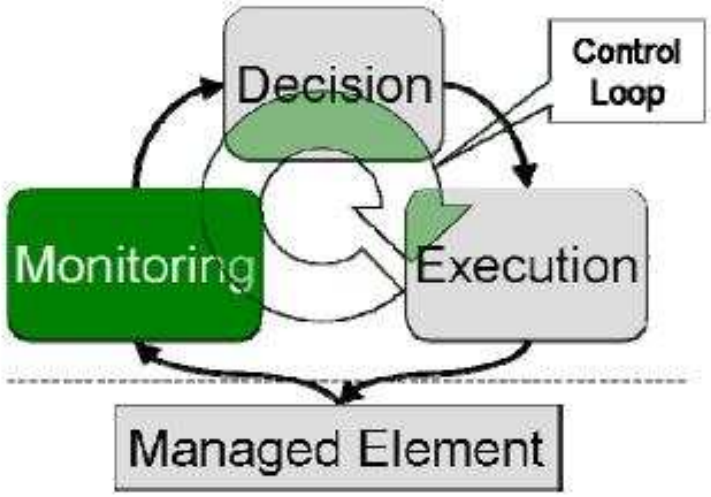

Figure 2.5: *Network monitoring* [1]

When the decision maker needs to take a decision about the node or the system it can trigger the functional block it requires information from, to supply the data for decision making.

Compartment level monitoring is responsible for monitoring the infrastructure of the entire compartment. Here, the decision makers and data collectors are not conceptualized as functional blocks but as nodes. The operations are the same as describe above.

Inter-compartment level monitoring is responsible for monitoring traffic flowing between two or more compartments to ensure quality of service guarantees, security guarantees etc.

ANA network seeks to use distributed collection points for monitoring data to ease the burden on a single or few data collection point. These collection points may collaborate by sharing monitoring information to ensure successful self-management.

2.4 Service Oriented Architecture

The significance of service-oriented architecture cannot be over emphasized in accessing modern architecture since modern technology operates in dynamic environments. It is therefore necessary to ask whether ANA network is service-oriented ?

Service orientation is a new way of building loosely coupled application. This is necessary in this era of globalization and rapid technological change. The idea of cooperation is gradually giving way to capital market. Capital market is built on

discontinuity but cooperation is built on continuity [21]. The focus of capital market is on creation and destruction as means of wealth creation. Technological change and public-policy shift has accelerated the rate of change in capital market, hence companies need to be more dynamic in order to survive. By extension technology also need to be dynamic in order to survive this rapid change.

Efficient integration is needed to meet the demands of capital market. Businesses undergo almost daily transitions which require rearrangement or realignment of business processes. For example when two companies merge their business processes need to be merged together to allow proper integration. Outsourcing is another example that needs change in business process.

From a technological point of view, new technologies are being developed almost everyday (Moore's Law). Regular technological update requires application update or changes. Service-oriented architecture is a way of designing systems which would be immune to these rapid changes or transitions. Norbert Sanjay and co. [21] defines service-oriented architecture as " *a framework for integrating business processes and supporting IT infrastructure as secure, standardized components-services-that can be reused and combined to address changing business priorities* ".

Additionally, service orientation is a way to bridge the gap between business operations and IT or technical operations to allow both of them to speak similar language thereby providing good understanding and also preventing process-integration failure. Information technology Infrastructure Library (ITIL) solves the language barrier problem by describing business as a collection of services. For example IT department provides IT services to the business, financial department provides financial services to the business. Departments are the owners of the services, and employees are the subscribers of these services. By using services to describe a set of loosely coupled processes own and manage by a department allows both non-technical and technical people in the organization to speak similar language.

Service is related to ownership of cost and risk. A service is define by OGC as means of "*delivering value by facilitating outcomes customers want to achieve without the ownership of cost and risk* ". The ownership of cost and risk is key in defining services. Service orientation must also provide a way of managing and measuring the business value of the services.

ITIL provides ways by which these services can be managed. ITILv3 provides three main ways of managing services namely, service transition, service operation and service design. Service operation provides prescriptions for managing incidences and problems that may arise as a result of running the service. Service transition prescribes change management principles such as release and update management. Similarly, service design prescribes solutions such as service level management and availability management.

The building blocks of ANA network include the so called functional blocks. Functional blocks are loosely coupled functions or processes that can be combined in any possible manner to make communication possible. Unlike IP network, what is unique about ANA network is that, policies, protocols and autonomic functionalities may all be implemented by loosely coupled functional blocks. The capabilities of compartments will be advertised as service through the external views of the compartment for nodes who conform to the policies of the compartment to subscribe. Service level requirements verifications will be implemented using functional blocks (FB).

2.5 Promise Theory and ANA

Promise Theory is an attempt to break away from the hierarchical way of modeling to a new realization that independent objects or entities interact not because one is superior to the other but because they offers services that each other needs. It represents hierarchical modeling as a set of autonomous entities which interact with each other through voluntary cooperation.

The elements of the discrete or autonomous entity form the services the entity provides. The policy governing the subscription or the use of these services are depicted as directed graph from one agent to another with a certain probability of honoring the service. This means the services are not honored all the time since the probability of honoring the service is mostly less than one.

This is true in real life since sometimes the survival of the service provider is threatened and needs to halt production in order to prevent extensive damage to the server or total collapse. For instance, a program can have severe exception during operation which forces it to suspend operation, a service provider may fall sick or become busy as result of flush crowd during service provision etc. In these cases, service production have to be prioritized or halted to recover before resuming the services.

When the probability of service provision is assumed to be one all the time, the system may not be considered as operating under the principle of voluntary cooperation. In other words, any policy that forces the autonomous entity to function against its will (e.g system must function even when the provider is suffering from exception or error) violates the principle of voluntary cooperation.

This is similar to the difference between deterministic system (implemented using finite state machine (FSM)) and stochastic systems. In FSM every state transition has the probability of one but in stochastic systems the probability is mostly less than one. The reality is that most systems are stochastic in nature.

The policies governing the interaction is expressed in the policy body of the promise which contains the promise constraint. This is not ”design” but realism. Promise The-

ory is more significant to resolve policy conflicts in ad hoc networks where there is no central authority that regulate the interaction.

The main goal of Promise Theory is to describe the policy that governs services subscription in discrete or autonomous entities, which assist each other through voluntary cooperation. This is done with the goal of eliminating policy conflicts [22]. In Promise Theory, there is no central authority that regulates the behaviors of other autonomous or the discrete entities.

The elements of ANA such are functional blocks, nodes, information dispatch points etc are completely autonomous entities governed by the policies of the compartment. They can therefore be described as autonomous policy objects otherwise known as autonomous agents in Promise Theory. For ANA to conform to the idea of voluntary cooperation the components of compartments should not be forced by any central authority to perform any function against their will, but function through voluntary cooperation.

Services in ANA compartments are exhibited as logical information channel IC governed by a certain policy. The policies themselves could be described as services represented by functional blocks. In essence, the policy of compartments can be described by Promise Theory, this way any hidden conflicts could be revealed to avoid conflicts of policies.

Promise Theory attempts to define policy as "*the ability to assert arbitrary constraints of the behavior of actors or effectors in a network*" [12]. Services are accessed, provided that policies are obeyed for closed compartments. For open compartments such conditions are not needed, every peer client is free to access the services offered by the compartment. Open compartments fall under basic services promise and close compartments fall under conditional services promise.

The three common type of promises are as follows:

- $c1 \xrightarrow{b} c2$, which means **c1** promises **c2** a certain service with promise body **b** which contains the promise type and the policy constraints. For open promise there is no such constraints.
- $c1 \xrightarrow{b/a} c2$, conditional promise which means **c1** promises **c2** a certain service containing the constraint **b** provided someone would promise **c1** with constraint **a**.
- $c1 \xrightarrow{c(b)} c2$, cooperative promise, **c1** and **c2** will cooperative with **b** as their guiding constraint or policy.

2.6 Network Tomography Monitoring Concepts

The term tomography is explained in [23] as the "*word used to link the field, in concept, to other processes that infer the internal characteristics of an object from external observation, as is done in magnetic resonance imaging or positron emission tomography*". Network Tomography is a powerful concepts and the merit of implementing it in any network cannot be overemphasized.

Active measurement has generally been used to measure the network infrastructure. There has been several approaches to doing this, ranging from complex queue theory approach to the more recent network tomography approach. Many differences have existed among the various approaches such as the need for protocol support, the amount of load or perturbations introduced to the network, the computational burden, internal or end-to-end approach.

The standard diagnostic tools for IP network, ping and traceroute fall under active measurement approach with protocol support. Ping and traceroute are supported by Internet Control Message Protocol (ICMP). Ping returns the round trip time for estimatation of network delay. Traceroute manipulates the time-to-live (TTL) field of the probe packet in order to estimate packet delay. The drawback to these approaches is their potential to congest the network if intensive probing techniques is adopted. Ping and traceroute are also based on unicast approach which has been found to exponentially increase the growth of network load [9]. Unlike the unicast approach, the multicast approach has been found to be more efficient in growth of network load. The main problem with multicast approach is wide spread protocol support. Multicast network tomography approach involves sending periodic probe packet through the network and making the corresponding measurement at the end nodes. The internal network characteristics are then inferred by examining the data collected from the multicast receivers end nodes [24]. *tcpdump* is one of the passive network measurement tools. It is usually used at the end node to collect packet traces which can be used to analyze the network. *Bro* is another passive network monitoring tool usually used for intrusion detection.

Network monitoring can also be done from an end node or the internal nodes. The internal approach requires extensive measurement of all relevant elements or nodes on the path of interest in order to determine the end-to-end performance [9]. This means,large amount of data will be collected and the issue of merging data, scalability and cooperation from the internal node will surface. The external approach requires little or no cooperation from the internal elements on the path of interest. It is solely based on the data collected at the end points. This has the potential of reducing the network load. In a way, this in contrast to the principle of ANA network which is not necessary an end-to-end network and requires the cooperation of nodes both internal and external in network measurement.

Network tomography inference using active data source with multicast protocol support is said to be promising [10]. There are several approaches and challenges confronting network tomography. Some of these challenges are identification of network topology, computational burden and in direct linear approach. The direct linear approach to inference in network tomography requires the identification of network topology and extensive matrix computations. The network tomography linear equation is given by the equation below:

$$\mathbf{Y}_t = \mathbf{A}\mathbf{X}_t + \xi \tag{2.1}$$

where $\mathbf{Y}_t$ is a vector of measurement such as packet counts or end-to-end observed delay recorded at a given time interval **t** at a number of different sites. Mostly, **A** stands for the (**i x j**) adjacency matrix of the network topology, usually, it has the value of 1 for connected nodes and 0 for unconnected nodes. The **i** represent the number of end node receivers and **j** represents the number of internal links. $\mathbf{X}_t$ is a vector of time-dependent packet parameters such as logarithms of packet transmission probabilities over a link, unobserved mean delays etc, ξ is the margin of error vector. ξ, depends on the bin size used in digitalizing the values of $\mathbf{X}_t$. It takes care of the error incurred as a results of the digitalization. ξ is usually, dropped to simplify equation (2.1). The simplified model is given by the following equation:

$$\mathbf{Y}_t = \mathbf{A}\mathbf{X}_t \tag{2.2}$$

([14])

In some cases $\mathbf{X}_t$ is a random vector with distribution approximately equals to $f(\mathbf{X}_t|\phi_t)$ and it is the parameter ϕ_t is the object of interest. The principal goal of the internal delay distribution is to estimate the values of ϕ_t. The dimensions of **X** and **Y** is **j** and **i** respectively.

The problem with this model is the potential size of the matrix $\mathbf{A}_t$ which may span from a few number of nodes to thousands of nodes. When $\mathbf{A}_t$ is in hundreds or thousands (e.g 100 x 500 matrix) finding the inverse of $\mathbf{A}_t$ in order to estimate the values of $\mathbf{X}_t$ becomes nearly impossible and generates immense computational burden. This is a typical example of inverse problem. The solution method for such inverse problem depends on the nature of ξ and $\mathbf{A}_t$. Mostly, the parameter ξ is assumed to be either approximately Gaussian, Poisson, Binomial or multinomial distributed. When ξ is Gaussian distributed the method such as recursive algorithm, linear least squares and other iterative equations solvers can be implemented. When the error margin is Poisson, Binomial or multinomial distributed statistical method such as reweighted nonlinear least squares, maximum likelihood via expectation maximization (EM) and

maximum a posteriori via Markov chain Monte Carlo (MCMC) algorithms, could be used. When ξ is dropped then how to estimate ξ is no longer necessary. The attention is now focussed on how to estimate ϕ_t which also requires the above solution methods.

The computational burden imposed by this model needs critical look. Various methods have been developed to solve this problem. Often times there is a trade-off between computational overhead and accuracy. Less accurate with less computational overhead methods include fast recursive algorithm for link delay distribution inference in a multicast framework and a method-of-moments approach for origin-destination matrix inference. More accurate but high computational overhead method includes maximum-likelihood methods. Unified pseudo-likelihood approach has been found to be accurate and produce less computational overhead in estimation of ϕ_t.

The main idea behind the pseudo-likelihood approach is to reduce the original model into a series of simpler sub-models by selecting pairs of rows from the adjacency matrix **A** and to form the pseudo-likelihood function by multiplying the marginal likelihoods of such sub-models. Equation 2.2 then becomes the combination of

$$Y^s = A^s X^s \tag{2.3}$$

where $s \varepsilon S$. S is a set of sub-model obtained by selecting all possible pairs of rows from the adjacency matrix. Another problem with the network tomography is how to find the matrix **A** which represent the network topology. Traceroute has been the traditional tool for determine the topology of a network however, it require the cooperation of the internal network devices and therefore may fail to identify certain devices. End-to-end measurement could be used to estimate the logical topology of the network. The difference between the logical and physical topology is that logical topology does not include the intermediate nodes. It is the subset of the physical topology. We can use the same end-to-end inference argument to estimate the logical topology.

Equally, network tomography model parameter can be estimated with different statistical model such Bernoulli loss model [9], Baysian approach and so on.

2.7 Testing ANA Network

The main focus of this work is to investigate possibility of implementing network tomography in ANA network. Specifically, to examine how passive measurement could be achieved in network tomography. Since there is no existing ANA network or testbed, testing anything ANA may be difficult due to relatively young life of the proposed network. [16] outlines how autonomic network can be tested:

- separate test could be used to access the behavior of autonomic system in differ-

ent scenario.

- adopting turning-like test where system is seen as black box with external view as the only verifiable.
- conducting continuous run-time or online test with a test function that will ensure the compliance of the system.

These recommendations provide a way out in the current situation where there is no existing network for testing. Successful experiments conducted using this recommendations on other networks could explicitly or implicitly apply in ANA network. This really, emphasizes the fact that ANA is mostly going to be the reusability of existing principles and works on other networks such as IP network.

2.8 Related Work

Similar work has been done by Y. Tsang and others ([10, 20, 9, 13]) to infer link delay distribution but what runs through many of these projects is a common focus on either algorithm to reduce the computational burden or inference technique (quantization methods, multicast probing or unicast probing etc) suitable to uncover the link delay distribution. Usually, the internal link delay distribution is estimated from digitalized end-to-end delay using algorithms such as expectation maximization. The focus of this thesis is however different. It is not about which algorithm or probing technique is more suitable but how to infer internal link delay distribution without any form of probing. That is, to infer internal link delay distribution solely from the packets that emerge from user activities.

This will be consistent with the demands of ANA network and would open the way for possible implementation of this new approach in both ANA and IP network. The thesis uses the least squares minimum norm method for the deconvolution of the internal delay for the subsequence estimation. Deconvolution means a process that reverses the effects of aggregation on recorded data. Unlike other works [25] the end-to-end delay is not digitalized from the onset. We use the continues end-to-end delay to compute $\mathbf{X}$ in the equation

$$\mathbf{Y}_t = \mathbf{A}\mathbf{X}_t \tag{2.4}$$

before digitalization of $\mathbf{X}$ to obtain the link delay distribution.

Chapter 3

Model and Methodology

3.1 Experimental Goal

This experiment is designed to test how to achieve best possible result in network tomography using passive monitoring and by extension show the feasibility of network tomography in ANA network. We define passive monitoring as monitoring with no probing packets. The main approach is to design an experiment that will allow the comparison of the propose passive approach against the established multicast and unicast methods. This will be consistent with the general practice specified in [13].

3.2 The Experimental Model and Analysis

3.2.1 Statistical Inference

There are certain distributions that approximately represent the behaviors of certain populations or generated sets of data. Such distributions are identified by their parameters and they are known as restricted family of distributions or models. The normal distribution is a model identified by the mean and the variance, while poisson distribution is identified by the expectation of the population. Sometimes a statistical model may not necessarily be a member of the restricted family of distributions but an equation of a theory. Once the kind of distribution or model for a particular population is identified or chosen, the parameters of the model need to be estimated in order to verify the goodness of fit of the chosen model, that is, how well the model represent the experimental data [26]. Statistical inference is about choosing a distribution from the well known family of distributions or models that represents a population and estimating the properties or parameters of that distribution. Once this is done inference

can be drawn from the distribution.

Parameters regulate the behavior of the statistical models, therefore the estimation of parameters are very vital in determining the nature of the distribution. Statistical inference provides various methods (such as maximum likelihood approach, least square, Bayes estimator) of estimating the parameter of a chosen model.

Parameter estimation is associated with the problem of whether the sample data is observed directly or estimated. Parameter estimation of observed data from a population is quite straight forward but it is more complicated when we have to use the observed data to estimate the parameter of another population. This is known as statistical inverse problem. Statistical inverse problem is about using information obtained outside a black box to estimate the information inside the box. The information outside the box may be directly or indirectly related to the unknown information. Typically, the unknown is infinite-dimensional.

Network tomography is a direct application of statistical inference. It is mainly about how to use the end-to-end network measurement to estimate the parameter that will yield similar properties as the true internal network dynamics. The internal network dynamics and the end-to-end data are related or connected by the universal equation $Y = AX$ where $\mathbf{Y}$ is a vector representing the end-to-end delay and $\mathbf{X}$ is a vector representing the internal network dynamics.

3.2.2 Network Tomography

Multicast probing for inferring network delay is known to be successful [9]. The main characteristic of multicast probing packets is that they experience the same delay on common links or links but different delay on uncommon links. This means the differences in end-to-end delay of multicast packet pairs will be caused by the uncommon links on which a pair of packet travels. This is key to resolving the delay on link by link basis [27].

The unicast approach of delay measurement tries to mimic the multicast approach by constructing packet pairs that are closely spaced in time. The idea is that when two packets are closely spaced in time they are likely to experience the same delay on a shared links and hence assumes the properties of multicast probing packets. However, in practice a packet pair is likely to experience different delays on shared links due to the fact that one packet precedes the other in the queues, additional packets are likely to intervene between the two. In effect, the differences between the delays on shared links results in measurement error in a set of end-to-end delay measurements. Nevertheless, the unicast approach is still able to estimate the average link delay since the errors are zero mean.

We extend the unicast idea by employing passive measurement approach for the

estimation of aggregated path-level delay. That is, the end-to-end delay is not determined by closely spaced probing packets but packets from normal user activities. The implication of this is that the requirement of packets experiencing the same delay on a common link will not be observed, yet we aspire to achieve similar results as unicast or even multicast probing results. The merit of this is to ensure that no additional packet is introduced into the network. It will also ensure that the packets for measuring internal link delay are truly independent. Above all, it may show the feasibility of using the network tomography in ANA network.

We propose two approaches that may neutralized the noise introduced into the path-level delay as a results of the discrepancies in shared link delay. The first approach is to model the path a packet travels as a queue. This may account for the noise introduced in the measurement as a results of the discrepancies. We assume that packets sent by users on a path are poisson distributed and therefore their end-to-end delay can be determined with better accuracy comparable to that of multicast if we model the path as M/M/1 queue. We estimate the aggregated end-to-end delay without the need for any form of coordination between the packets that transverse the path.

The second approach is to use the adaptive learning approach used by cfengine (configuration management software). The adaptive approach is an approximation to Bayesian thinking which is cheap to compute and does not rely on the assumption of a stable end-distribution [28].

3.2.3 Model Analysis

The network tomography simplified linear model is given by the equation

$$\mathbf{Y}_t = \mathbf{A}\mathbf{X}_t \tag{3.1}$$

In order to use this model, two of the three variables must be known to find the other. In the formula $\mathbf{Y}_t$ represents the end-to-end delay, $\mathbf{A}$ is the network topology and $\mathbf{X}_t$ is the hidden internal delay. The end-to-end delay of a path is how long it takes for a packet to move from one end of a path to another end. The end-to-end delay consists of transmission delay, processing delay, queue delay and propagation delay. Transmission delay is the time taken for a packet to move from one end of a communication link to another. The processing delay is the time when a packet is received by a node and the time the packet was transmitted. The propagation delay is the delay cause by the property of the transmission media. Queue delay is the time difference between when a packet enter a queue and when it leaves the queue. We put the end-to-end delay into two components, propagation delay and servicing delay. Servicing delay consists of processing delay, transmission delay and queue delay.

The end-to-end delay is needed for the estimation of the internal link delay distribution. There are two main approaches to finding the values of $\mathbf{Y}_t$, multicast and

unicast. In both cases however, probing is required to achieve results. The new approach here is to achieve similar results as the multicast approach without the need for any form of probing.

We reiterated that, the network tomography fails when probing packets travel at different rates since passive end-to-end delay measurement does not take into consideration the randomness associated with the movement of a packet traveling on a path.

We design an experiment that aims at solving this problem.

There are four main assumptions in this methodology,

- packets generated by users from one end of a path is poisson distributed
- there will be enough packets that will transverse the path during the short period of measurement
- the network topology is a tree structure
- minimum norm least squares solution is the correct estimate of the internal link delay

When the link delays along a path are statistically independent, the end-to-end delay densities are related to the link delay densities through a convolution or combination of the individual link delay. Convolution methods are used to estimate the internal delay densities from the end-to-end delay. The idea of convolution can be compared to fourier transformation where signals can be combined to pass through a communication medial and split at the end of a communication channel. Some of the convolution methods are least squares, transformation of the convolution into more tractable matrix operator via digitalization of the delays, expectation maximization algorithm (EM),iterative proportional fitting algorithm [29], estimation of the link delay cumulant generating function CGF from the end-to-end delay cumulant generating function CGF [30].

The independence of link delay assumption does not hold strictly in a real network due to the temporal correlations between network traffic. The use of disjoint passive packets could minimize the dependencies in packets use to estimate internal delay distribution and hence make the model more realistic. The assumption of temporal independence is feasible as long as the interval between probes is large enough [31]. This highlights one of the merits of our proposed approach.

The linear equation **Y = AX** is "under-determined" since $i < j$ for the i by j matrix A. Under-determined linear equation has no or infinite number of solutions therefore algorithms are required to choose the best-fit solution. Among the infinite values of **X**, the least squares approach adopt the minimum norm as the best-fit solution. The minimum norm is the unique solution **x** that minimizes $\|x\|_2$.

The internal delay distributions could be estimated, directly or indirectly. The indirect approach estimates the internal distributions without directly solving the linear equation for all the values of **X**. The direct approach solves the linear equation in order to obtained the distribution of **X**.

The accuracy of inference depends on which solution method is use to estimate the parameter or maximize the distributions of **X**. As mentioned before, there are several algorithms that can be used to estimate the distribution of **X**. Maximum likelihood estimator, least squares, pseudo-likelihood and so on.

The maximum likelihood estimator (MLE) is a better parameter estimator than the least squares estimator [26], usually, MLE is consistent with the true parameter value that generated the data. MLE is expensive because it requires the computation of all the values of **X** as in direct approach. The goal here is not to find out which estimator achieves the best results but to determine how passive network tomography approach may perform compare to multicast and unicast correlation properties.

We will use the MLE method to estimate the internal delay distribution. This means we will estimate the values of **X** to obtain the internal link delay distribution. MLE will be computed directly from the data without any algorithm.

3.2.4 The Queue Model

A queue is a generic processing model where jobs arrive as event and wait for processing. Queue models are statistical models that describe the steady-state properties of stochastic task system [32]. Queue models consist of a number of parameters including the following:

- Arrival time: defines the rate at which event occur for processing
- Service time: defines rate at which event is processed
- Number of servers: defines number of entities (computers, human etc) responsible for processing
- Scheduling policies: defines the algorithm used to select a job or event for processing

Queues are denoted in Kendall notation of the form A/S/c(B/K/P), where A is the inter-arrival distribution that usually takes one of the following values:

- M Memoryless (exponential/poisson)
- E_k Erlang with parameter k

- H_k Hyper-exponential with parameter k
- D Deterministic
- G General

A deterministic distribution has a constant inter-arrival rate. General, means the model's results can apply for all distribution. Memoryless means that the current state of the arrival distribution does not depend on the past. S is the processing time distribution, c is the number of servers, B is the number of buffers. K is the maximum population size and P is the policy.

One of the queue models is the M/M/1 queue model. The M/M/1 system is made of a Poisson arrival, one server, first in first out (FIFO) queue of unlimited capacity and unlimited customer population. The M/M/1 queue processes packets at a certain rate μ (service rate), with packets arriving at a rate λ (inter-arrival rate). Service rate μ is how long it takes for a packet to travel through a network path. The inter-arrival rate λ is how often packets arrive at the source. The delay in the queue observed at the end node (observed delay) is given by

$$W_t = 1/(\mu - \lambda) \tag{3.2}$$

λ is given by 1/E(X) where X is the expectation of the random sample $x_1, x_2 ... x_n$ representing the inter-arrival time. Similarly, μ is given by 1/E(Y) where E(Y) is the expectation of the random sample $y_1, y_2 ... y_n$ representing the service rate [33].

Poisson distribution is a discrete probability distribution of random variable X (X takes the values 0,1,2 ...). It is one of the restricted family of distribution with parameter $\lambda > 0$. The poisson distribution is given by the formula

$$P(X = k) = e^{-}\lambda \frac{\lambda^k}{k!} \tag{3.3}$$

where k=0,1,2 ...

3.2.5 Estimators

Parametric distributions such as Poisson distribution, depends on the values of the parameter λ. Different values of λ gives rise to different distributions. MLE provides a way of estimating the parameter of a statistical model to some data. It helps to fine tune the parameter of a distribution to provide optimum performance. For poisson distribution the MLE is determined as follows: if $x_1, ..., x_n \sim$ Poisson with parameter λ.

Then $\log l(\lambda) = \sum_i \log(e^{-}\lambda \frac{\lambda^{x_i}}{x_i!}) = -n\lambda - n\bar{x}\log\lambda - \sum_i \log(x_i!)$

$\Rightarrow f'(\log l(\lambda)) = -n + \frac{n\bar{x}}{\lambda} = 0 \Rightarrow \hat{\lambda} = \bar{x}$

Therefore MLE of λ is the expectation of x_i. This is true for univariate distributions. Multivariate distributions are more complex than this.

The use of parametric equations in network tomography is not encouraged by some practitioners [31] since it leads to less accurate results. Some prefer nonparametric estimation. The discrete model uses non-parametric approach to estimate the delay distribution. It begins with the digitalization of the delay measurements followed by the estimation of the parameter that will maximize the delay distributions with a given algorithms such as estimation maximization algorithm. The accuracy of the maximized distributions depends on the kind of algorithm used for the maximization.

Furthermore, if the frequencies of the elements of each link **X** is known then the MLE is given by

$$\widehat{P_i, j} = \frac{m_i, j}{\sum_{i=1}^{L} m_{i,j}} \tag{3.4}$$

where i is the link number, j is the digitalized delay, L is the number of bins and $m_{i,j}$ is the number of packets that experienced the digitalized delay j on link i. $\sum_{i=1}^{L} m_{i,j}$ is the total number of the packets experienced on link i [27].

X will be estimated using the ordinary least squares which employ the minimum norm solution with octave software, digitalized the elements of each **X** and count the frequency of each element of **X** and divide by total number of packets experienced on a link in order to estimate $\widehat{P_i, j}$. The ordinary least squares approach by octave forms linear relationship between the dependent variable **Y** and the independent variable **X**. The elements of **X** are then estimated from this linear combinations.

3.2.6 Adaptive Learning Model

Adaptive learning is an iterative method for expectation estimation that degrades the importance of data over time. It is motivated by Bayesian theory. The current expectation E_{n+1} is given by

$$\alpha q_{n+1} + \beta E_n \tag{3.5}$$

. Where α and β are constants with $\alpha + \beta = 1$. β defines the significance or contribution of the old measurements while α determines the significance of the current measurement, E_n is the previous expectation. In this experiment, we choose α to be 0.7 and β to be 0.3. The initial expectation is the first end-to-end measurement of each path. Adaptive learning approach will be used to minimize the noise in the path-level delay cause by the different delay on shared link.

The merit of adaptive learning is that it is cheap to compute. Adaptive learning formula is used in cfengine software (See [28]) for more information on adaptive learning).

3.3 Experimental Design

Two sets of similar experiments, passive/multicast, unicast/multicast are design and their correlation properties compared. The closer the correlation properties the better the results [13]. The unicast/multicast experiment will serve as a benchmark experiment for the passive/multicast experiment. The experiment was implemented using ns-2 simulator.

3.3.1 Simulators

The use of test beds and real lab for network is likely to produce accurate results, however, building test beds and labs is expensive. They are inflexible and difficult to reconfigure. For certain networks such as wireless networks reproducing results is difficult. Simulator are good for protocol design and also to help to avoid the constraints above.

ns-2 stands for network simulator [34]. It is one of the main network simulator used in academia for network experiment. It can be used to test new protocols etc. It uses two main languages, Tool Command Language (TCL) or Object Tool Command Language (OTCL, the object oriented extension of TCL) and C++. The OTCL is the front end language and C++ is the back end language. It is used for detailed protocol modeling. OTCL runs slower but can be changed faster than C++. It also support a visualization tool called NAM. Network Animator Tool (NAM) helps to visualize the process of the network simulation.

There are several simulators including Matlab, OMNeT++ and ns-2. While Matlab is suitable for mathematical model simulation, OMNet++ and ns-2 are both network simulators. ns-2 is free for both commercial and educational purposes but OMNeT++ requires licence for commercial used. OMNeT++ is more flexible than ns-2 but for TCP/IP simulation ns-2 may be superior to OMNeT++.

3.3.2 Tools

- octave: is used for mathematical calculation
- excel: calculation and plot

- awk: for text processing
- grep: for text processing
- ns-2: the network simulator for simulation
- lenovo 3000 c200, Intel CPU, T2080 at 1.73GHz, 0.99Gb RAM
- OS : Ubuntu version 7.04 Feisty

3.4 Experimental Setup

The topology of the network is shown in figure 3.1. The numbers 0,1,2,3,4,5,6,7 represents the node in the network. Each of the internal links (link1-2 and link1-3) have propagation delay of 50ms and bandwidth of 5Mb. The outer links (link0-1, link2-4, link2-5, link3-6, link3-7) have propagation delay of 10ms and bandwidth of 1Mb. In order to evaluate the effectiveness of the proposed passive method we conduct ns-2 simulation using the network topology in figure 3.1. The topology is chosen to allow for easy computation of the values of vector **X**. Each node uses FIFO queue (Droptail in ns2) with a 10-packet capacity. The link between each two nodes is full duplex. The background traffic consists of two TCP streams with random packet size and transmission rate generated respectively by Pareto and Exponential distribution. One from node 0 to 4 and the other from node 5 to 6. The third background traffic is a poisson distributed udp streams from node 0 to 7. The streams have the burst time of 0 packet size of 1000 bytes, idle time of 2ms and transmitting at the rate of 0.5Mb. When the burst time is zero and the transmission rate is high the packet transmission distribution turns to poisson.

Exponential multicast probing packets of size 40b, burst time set to 0, idle time of 18ms and rate of 20Kb is sent to all end nodes. This specifications are the same for both experiments. For the multicast/unicast experiment, we design a closely related packet pairs for each of the two multicast tree paths, with each pair separated by 0.1ms. The pairs from node 0 to node 4 and 5 are exponential on/off traffic distribution with packet size of 20Kb, burst time of 700, idle time of 8ms and rate 6Kb. The codec_selector_ variable is set to zero to make the transmission rate constant, this is to ensure that the two packet pairs are always tied together. Similarly, packet pairs from node 0 to 6 and 7 have the same specification except that the transmission rate is 10Kb to ensure that the two set of packets pairs travel at different times on link 1. (see the script at Appendix D).

For passive experiment, we choose random traffic generators which are poisson distributed. The poisson generator is achieved by setting the burst time of the Exponential On/Off generator to 0 and the rate of transmission to some big value. The

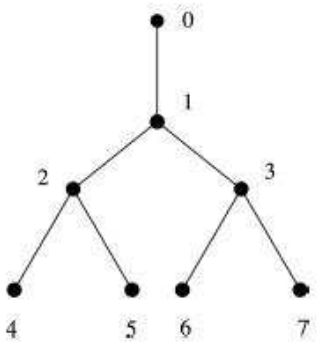

Figure 3.1: *topology of the experiment*

inter-arrival time for each pair of generators is 8ms and 72ms. The packets transmission rate is 20Kb/s for each generator. The random_ variable and the codec_selector_ of the generator is set to true and 1 respectively to introduce more randomness. The mean inter-arrival time differences is to enable each pair of generators travel independently. Unlike the unicast approach, the traffic generators here are separated by a period of 64ms which is more than half of the average time a packet spends in traveling a whole path. This means a pair of passive packets is likely to encounter different delays on a shared link.

3.4.1 Test Plan

The correlation properties among multicast, unicast and the passive experiments for each approach will be compared. The end-to-end delay will be correlated in both experiments. For the convergence test, 400 probes of both passive and unicast measurement will be compared to their respective multicast results to find which of the two converges faster to their respective multicast results. The multicast experiment will still maintain the 1000 i.i.d results.

3.4.2 How to Obtain Results

The experiment is run for 200 seconds and the end-to-end delay is recorded. The simulation consists of custom scripts obtained from [35]. The script is able to compute the delay on a link or a path as packets are being transmitted. It also computes the start time and arrival time of the packet. We attach multicast receivers at each node to record each link delay as well as the end-to-end delay using the delay script. We also attach other receivers to record the end-to-end delay of both passive packets sources and unicast packets sources in each of the two experiment.

Seven files which contain path level delays, the start time and the arrival time at each node are created during the simulation by the multicast probes in addition to a single source file containing the starting period of each multicast probe. Both the passive and unicast sources create eight files, four sources files containing the start periods and other four containing the end-to-end delay in each separate experiment.

The first 1000 probes is collected from each experiment. When the end-to-end delay for each probe is obtained, the value of the vector **X** is computed from the equation **Y=AX** using octave software (see script in Appendix E). If the system of equation has infinite number of solutions, Octave uses the minimum norm solution to determine the minimum solution. The end-to-end delay becomes the values of **Y**. We use the direct approach to obtain the distributions of **X**. **A** is the adjacency matrix representing the network topology. It is a matrix of connected paths and obtain as shown in table 3.1.

nodes/links	1	2	3	4	5	6	7
4	1	1	0	1	0	0	0
5	1	1	0	0	1	0	0
6	1	0	1	0	0	1	0
7	1	0	1	0	0	0	1

Table 3.1: *how to derive topology matrix*

The column of the table represents the individual links and the rows represents the end nodes. If there is a direct path or link from an end node to a link then the value of the matrix is 1 otherwise 0. Each end node is directly connected to itself.

The values of **X** obtained from the matrix computation is counterbalanced to obtain the estimated transmission delay on each link for each of the 1000 packets (we assume that at least one probe has experienced no queuing delay along the path). We do this by subtracting the propagation delay (which is 10ms for the outer links and 50ms for the inner links) of each individual link from the corresponding value of **X**. We then take the absolute value of the resultant vector (see [20]). We assume that the propagation delay is the same as the minimum delay of each link. The set of all such resultant vectors become the sample distribution for each corresponding links from which the probability of each element of X_i can be estimated using MLE formula stated in section 3.2.5 above.

The next step in the estimation is digitalization using a chosen bin size. We group the values of each **X** using a given bin size and assigning a number to each group of values. The bin size (q) determines how to group each set of **X** values. The bin size of one means the range of a group is 1. The bin size also determines the accuracy of results and the computational cost. The smaller the bin size the higher the accuracy of the parameter estimation but higher the computational cost. It is related to the error (in equation 2.2). The smaller the bin size the smaller the error.

It is also required to choose the maximum number of bins ($m > 0$). Any value greater than mq is considered as lost packets. We illustrate this with an example. Suppose X_1 values for link1 is 1.1,1,1,1,2.2,2,2,4,5,6,6,10,10,11,11. Suppose we choose bin size of 1 and the maximum number of bin is 8 so that mq=1x8=8. We group the sample distribution according to the bin size and digitalize the results is shown in table 3.2.

Range	0-1	1.1-2	2.1-3	3.1-4	4.1-5	5.1-6	6.1-7	7.1-8	∞
Digits	0	1	2	3	4	5	6	7	∞
Frequency	4	3	0	2	1	1	2	0	4

Table 3.2: *digitalization table*

0-1 means greater than zero but less than or equal 1 and so on. We then find the probability of the occurrence of each of the values and use them to estimate the link delay distribution. ∞ in this case means any value greater than 8. The value of infinity in the above example is 4 meaning four packets were lost during the probe.

In general X_j takes finite possible values 0, q, 2q, ...,mq, ∞, where q is the bin size or width and m is a chosen constant. That is each X_j is a discrete random variable whose possible values are 0, q, 2q, ...,mq, ∞ with respective probability Pi=Pi0,Pi1, ..., Pim, ∞_j. The main idea behind network tomography link delay distribution is to estimate The Pi's. When the delay is infinite it implies packets are lost during the transmission or the values of the digitalized delay is greater than the maximum delay mq.

In this work each X_j has 16 measurements, from 0 to 15. The number 15 is the infinite. Digitalization is performed on each of the set of values of **X** to obtained the discrete distribution.

After this process the results on each link for both multicast estimation and the passive or unicast estimation is plotted and their correlations compared. The end-to-end delays are also compared.

3.4.3 How to Obtain The End-to-end Delay of Passive Data Source

The end-to-end delay of the passive method is determined using the queuing model discussed in section 3.2.4. The E(X) is the running average of the arrival rate. We find E(X) by subtracting the starting time from arrival time of each packet transferred and averaging them over successive "probes" start from the first probe. The starting time and arrival time are recorded by the delay script used in the ns-2 simulation. Similarly, we obtain second set of the end-to-end using the adaptive formula discussed in [28].

We cannot use the end-to-end passive data source directly for the estimation since passive packets are likely to experience different delays on shared links. We need to minimize this uncertainty in order to arrive at a result similar the unicast method. So we propose two models which we hope could minimize the uncertainties in the end-to-end delay measurements. The two models are the M/M/1 queue model and the adaptive learning model. First of all we collect the passive end-to-end delay and process it with the two models. This is done as follows:

Suppose we send packets from node 0 to each of the end nodes (figure 3.1). Let the delays experienced by the end nodes be equal to those shown in the table 3.3.

Node 4	Node 5	Node 6	Node 7
70	71	80	72
72	71	74	72
70	76	75	72
70	71	80	72

Table 3.3: *end-to-end delays*

Suppose the corresponding observed inter-arrival time is shown in the table 3.4

Node 4	Node 5	Node 6	Node 7
30	30	10	22
30	30	11	22
30	30	22	22
20	40	40	22

Table 3.4: *inter-arrival time*

The end-to-end delays are from passive data source and contain some uncertainties since they experienced different delay on a shared link. We apply the queue and the adaptive model to the above delay measurements with the hope of reducing the uncertainties. The estimation is done as follows:

Queue Model
Using the equation

$$W_t = 1/(\mu - \lambda) \tag{3.6}$$

λ is given by 1/E(X) where X is the expectation of the random sample $x_1, x_2 ... x_n$ representing the inter-arrival time. Similarly, μ is given by 1/E(Y) where E(Y) is the expectation of the random sample $y_1, y_2 ... y_n$ representing the service rate (see section 3.2.4 for more delays). We show how to estimate the end-to-end delays for node 4 as follows:

λ	μ	W_t	Delay(ms)
70	30	$\frac{1}{1/30-1/70}$	52.5
72	30	$\frac{1}{1/(60/2)-1/(142/2)}$	52
70	30	$\frac{1}{1/(90/3)-1/(212/3)}$	52.1
70	20	$\frac{1}{1/(110/4)-1/(282/4)}$	45.1

Table 3.5: *end-to-end delay estimation with queue model*

The adaptive model is given by

$$\alpha q_{n+1} + \beta E_n \tag{3.7}$$

. Where α and β are constants with $\alpha + \beta = 1$. β defines the significance or contribution of the old measurements while α determines the significance of the current measurement, E_n is the previous expectation. We choose α to be 0.7 and β to be 0.3 and recompute the delay with adaptive learning model as follows:

Delay	Formula	E(Y)
70	70	70
72	$0.7*72+0.3*70$	71.4
70	$0.7*70+0.3*71.4$	70.42
70	$0.7*70+0.3*70.42$	70.13

Table 3.6: *end-to-end delay estimation with queue model*

The results of the calculations are used to estimate the internal delay from the equation $Y = AX$. Where **Y** values for node 4 in this case are (70,71.4,70.42,70.13) for adaptive learning. For the queue model **Y** is (52.5,52.1,45.1) for node 4. Since the routing matrix is known already (see section 3.4.2) we can calculate the internal link delay by solving the equation $Y = AX$ using Octave software. After solving the equation we get seven values with each X value representing a link delay. Let X1 represent the set of delays on link 1. We subtract the propagation delay from each of the values of X1. Sometimes the results may be negative. When a value is negative we take the absolute value since delay cannot be negative. After obtaining the X1 values for each probe, we normalize the values of X by subtracting the smallest value of X1 from the each of the values of X1. The results is digitalized as shown in (section 3.4.2). This process is repeated for the rest of the links.

For the multicast and the unicast measurements, we do not perform any computation on them. We use the raw observed delay to find the values of X in the equation $Y = AX$ and repeat the processes discussed above for each X_i .

For example, suppose the first probe resulted in y4=72, y5=71, y6=74 and y7=72, where y_i are the end-to-end delay on node 4, node 5, node 6 and node 7 respectively. We will have the following:

The matrix **A** is given by

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

The vector **Y** will be:

$$Y = \begin{pmatrix} 72 \\ 71 \\ 74 \\ 72 \end{pmatrix}$$

We solve the equation $Y = AX$ for **X** using Octave (see the script in appendix E).

Suppose after solving the matrix equation $Y = AX$ we got x_i to be 40,30,20,8,10.2,20,12 where 40 is the delay on link 1, 30 is the delay on link 2, 20 is the delay on link 3, 8 is the delay on link 4, 10.2 is the delay on link 5, 20 is the delay on link 6, 12 is the delay on link 7. This delay includes the propagation delay so we subtract the propagation delay from each results and take the absolute value. In this experiments the propagation delays are 10,50,50,10,10,10,10 for links 1,2,3,4,5,6,7 respectively. The resultant delay will therefore be 30,20,30,2,0.2,10 and 2 for link 1,2,3,4,5,6 and 7 respectively. We repeat this operation for every probe to obtain the continuous delay for digitalization.

3.4.4 Performance Metrics

The following performance metrics will be used to compare results:

- correlation coefficient R
- relative L1 error norm
- uncertainty
- convergence
- effect of long term probes

3.4.5 Correlation

Residual **r** can be used to access the effectiveness of prediction. It is the difference between the observed or actual value and the predicted value. It is given by the equation

$$r(i) = Y_{actual(i)} - Y_{predicted(i)} \tag{3.8}$$

, where r(i) is the residual of the point i , $Y_{actual(i)}$ is the actual value of **Y** and the $Y_{predicted(i)}$ is the predicted values of **Y**. Small residual indicate better prediction.

Correlation is the direct results of the residual idea. Correlation **R** is a measure of model prediction. When all the residuals is zero then, we have a perfect correlation. It means there is no error in the prediction. The Pearson correlation coefficient, **R**, measures the extent to which two variables (predicted and actual) are related. It is given by the equation

$$\Sigma \frac{(x_i - \bar{x})(y_i - \bar{y})}{(n-1)s_x s_y} \tag{3.9}$$

where i=1...n pairs, s_x, s_y are the variances of the two variables.

$$-1 < R < 1 \tag{3.10}$$

. When R=1 or R= -1 we have perfect positive and negative correlation respectively. In general when $R = 0.7$ and above we have strong positive correlation, when $R = -0.7$ and below we have strong negative correlation. When $R = 0$ we have no correlation.

Another correlation metric is R^2. It determines the percentage of the variation in **Y** explained by the variation in **X** over the range of **X**. R^2 is the square of correlation **R**. When R^2 is 100% it means all the variations in **Y** can be explained by **X**.

The unicast and multicast results are the bench mark results or the actual values of **Y** and the passive results are the predicted values of **Y**. We will therefore use correlation to access the effectiveness of the passive prediction using unicast and multicast results as the actual values.

The effect of large probes on the result can be determined by examining the correlation between a small number of probes and a large number of probes. We will examine the effect of 400 probes and 1000 probes experiments. The wide traffic and link diversity gives rise to small correlation for large probes as against small number of probes.

3.4.6 Uncertainty

Uncertainty determines the error in measurements. Standard deviation will be used to determine the uncertainty in the end-to-end delay measurements. The standard deviation is given by

$$\sigma = \frac{1}{N}\sum_{i=1}^{N}\sqrt{(x_i - \bar{x})^2} \tag{3.11}$$

where x_i are the elements of **X**, N is the number of elements, $\bar{x}$ is the mean of **X** and σ is the standard deviation. The variance is the square of the standard deviation.

The variance between the various measurements are compared using the largest absolute deviation of the multicast/passive and multicast/unicast experiments (see [25]). The smaller the value the better the estimation.

3.4.7 L_1 Relative Error Norm

Error norm is from the ideal of residual. It measures the difference between the exact solution and calculated solutions. L_1 error norm is the absolute relative error norm

over a certain number of elements. It is given by

$$\sigma = \frac{1}{N}\sum_{i=1}^{N}\frac{|E_i - C_i|}{|E_i|} \tag{3.12}$$

. Where E_i is the exact value over certain number of probes and C_i is the calculated or estimated value over certain number of probes. A higher value means the relative difference between the exact or actual values and the estimated hence large error.

Error norm helps to measure the relationship between the various probability mass functions (pmfs). The L_1 relative error norm of normalized delays of each link will be compared. The relative error norm is the average of error norms. The smaller the value the better the relationship.

3.4.8 Convergence

Convergence is how long it take to get a result. Two experiments may produce the same results but one may take longer than the other.

The variance determines the spread of a distribution and also shows how fast two distributions converges. The variance could be used to roughly determine the rate of convergence and effect of the long term probe on the inference results (see [25]). The variance of 400 probes of the unicast and passive experiments will be compared to 1000 independent and identically distributed (i.i.d) of their respective multicast results. The one with small difference is likely to converge faster.

3.4.9 Possible Experimental Error

The following are the possible errors that may affect the results:

- unicast probing depends on the time space between packet pairs. If this is not chosen carefully it may affect the correlation properties of the unicast packet pairs.
- M/M/1 queue requires unlimited queue size but the queue size used in this experiment is 10
- digitalization of results has the potential to degrade the accuracy of measurements, since it depends on the bin size.

3.4.10 The Kind of Data to Compare

For the internal delay inference, we are interested in comparing normalized data rather than the actual data itself [25](p.18). The inference data is normalized by subtracting the minimum delay of each set of data from all the elements of that particular set of data. After this is done the results is compared. The normalization tries to put the results under the same scale to enable possible comparison of the delay distributions. The results is rounded off to two decimal places. The comparison of the normalized data is done with L1 relative error norm, averaging over 50 probes.

We compare correlation properties of the digitalized distribution by finding the correlation coefficient R and R^2.

Chapter 4

Results

4.1 Description of Results and Test Procedures

We conducted two separate experiments and labeled them passive/multicast and unicast/multicast. The passive/multicast means the experimental setup has passive packet generator from node 0 to the end nodes 4,5,6 and 7. On the same network there is multicast data source which send probes to the same end nodes 4,5,6 and 7.

We used the multicast data source to estimate the true end-to-end delays so that we can compare it to the passive results. Similarly, the second experiment also has the same setup but the traffic generators are unicast packet pairs and multicast probing packets. Each experiment produces two set of end-to-end delay measurements. So we had four sets of data in the end. We then pair them as passive/multicast and unicast/multicast.

We compare the performance of the individual experiment and also the performance of the two separate experiment to ascertain whether passive data source approach is possible in network tomography. We know that the passive data source contains uncertainty since passive packets may not experience the same delay on a shared link. We therefore adopt two models that would minimize the uncertainty in the end-to-end passive delay measurements before they are used to infer the link delays. The models are the adaptive model and queue model. So the passive results has two end-to-end delay measurements one for adaptive model and the other for queue model. We refer to them as adaptive and queue respectively. We may also refer to unicast/multicast as ”unicast” and to passive/multicast as ”passive”.

We use the queue and the adaptive models to recalculate the end-to-end delay so that we can remove some of the uncertainties. The resultant estimated end-to-end delay is now used to compute the internal link delay distribution using the equation $Y = AX$. Similarly, observed unicast end-to-end delay and the multicast end-to-end delay are

also used to compute the internal link delay distribution.

The experimental results of the end-to-end delays and link delay distribution estimation are included below for both passive/multicast and unicast/multicast experiments.

Each subsection represents a particular set of experiments, their descriptions and summary statistics. It is a partial display of experimental results. The rest of the results is found in Appendixes A,B and C.

In order to show the differences in the observed end-to-end delay and the estimated delays we plot the observed unicast end-to-end measurement versus multicast end-to-end measurements. We also plot the end-to-end delay of the passive experiment, both the observed multicast end-to-end delay and the estimated adaptive and queue model delays.

Similarly, the link-level inferred delays are plotted for adaptive/multicast, queue/multicast for the passive experiment and unicast/multicast for the unicast experiment. The adaptive/multicast is comparing the link delay estimated by the adaptive model to the link delay estimated by the multicast technique. Also the queue/multicast is the comparison of the multicast link delay results and queue model link delay results. Finally the unicast/multicast is the comparison of the link delay estimated by the unicast technique and it multicast technique. The multicast probes were designed in such a way that its impact on the other test is minimal.

Each of the link delay plot is discrete but for easy visual comparison, we plot them with continuous curve. We also estimate the convergence rate and how long term probes affect the results. We do this by comparing the results of 400 probes of the unicast and passive measurements to the 1000 probes of their respective multicast measurements. We calculate the variance of the two results to find which one converges faster.

4.1.1 End-to-End Delay

The end-to-end delay of the passive experiment (passive/multicast) and unicast experiment (unicast/multicast) for selected end nodes is shown below. Each plot depicts the traffic flow on a path. We compare the unicast and the passive experiments. They come from 1000 probes of ns-2 simulation.

The figure 4.1 depicts the end-to-end delay experienced by node 4 from the unicast experiments. It shows the observed unicast and the multicast end-to-end delay of node 4. The plot shows that both unicast and multicast have similar distribution with comparable characteristics.

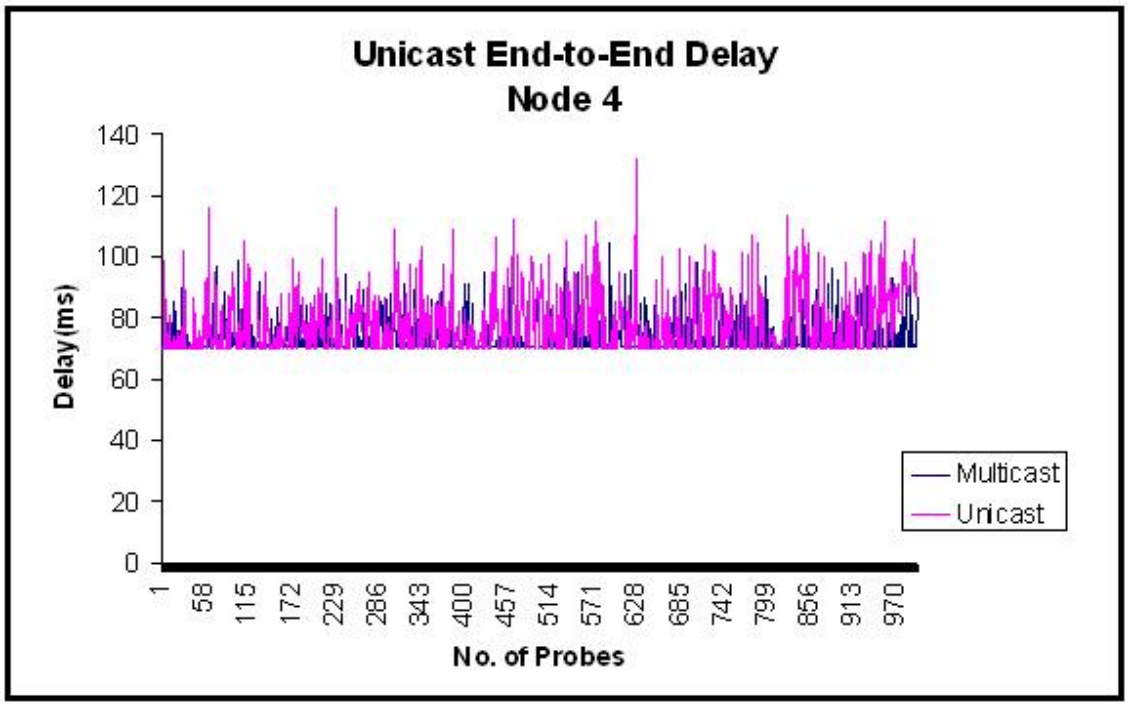

Figure 4.1: *Unicast End-to-End Delay for Node 4*

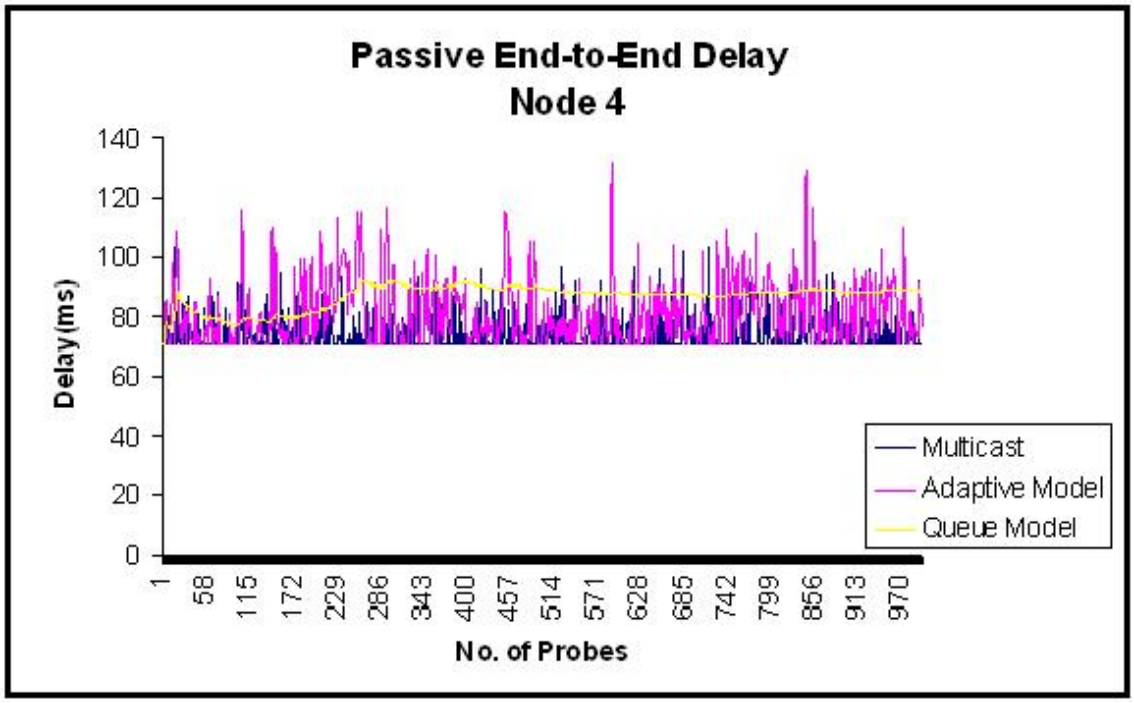

Figure 4.2: *Passive End-to-End Delay for Node 4*

Similarly, figure 4.2 shows the end-to-end delay experienced by node 4 from the passive/multicast experiment. It shows the estimated passive delay of node 4 using the adaptive and the queue model. The queue model is in the middle of the plot somehow depicting the average of the two results. The adaptive model and the multicast results are quite similar. There are no major sparks in the end-to-end queue delay measurements.

Further examination of the results is illustrated in table 4.1 which shows some of the summary statistics of End-to-End Delay of both experiments.

Absolute Deviation	$\|S_2^2 - S_1^2\|$	Answer	Node
Multicast/Adaptive	29.64-114.89	85.25	Node 4
Multicast/Queue	29.64-16.14	13.50	”
Multicast/Unicast	32.29-99.55	67.26	”
Multicast/Adaptive	12.82-54.49	41.67	Node 5
Multicast/Queue	12.82-0.66	12.16	”
Multicast/Unicast	13.46-67.47	54.01	”
Multicast/Adaptive	22.36-90.91	68.55	Node 6
Multicast/Queue	22.36-50.28	27.92	”
Multicast/Unicast	21.56-57.46	35.9	”
Multicast/Adaptive	35.77-58.17	22.4	Node 7
Multicast/Queue	35.77-0.79	34.98	”
Multicast/Unicast	35.37-74.44	39.07	”

Table 4.1: *absolute difference of variance for end nodes*

The table 4.1 depicts end-to-end absolute difference of variances for the various categories of experiment. The adaptive model and the unicast have bigger variances than the queue model. They also produce bigger absolute difference of variances.

4.1.2 Link Delays

The following depicts the delay distribution on individual links. The bin size of the plot is 1 and the number of bins is set to 16 with bin 15 representing large or infinite delay. We rounded off each link delay to zero decimal places and normalized by subtracting the smallest element from the rest of the elements. The results is then digitalized with bin size of 1 and plotted as shown below. Total number of packet used in this plot is 1000.

Figure 4.3 shows the estimated link delay on link 1 for both passive and unicast experiments.

Figure 4.3, ***(A)*** shows the estimated digitalized delay using the passive end-to-end measurements calculated with the queue formula and the delay estimated by the real multicast end-to-end delay in link 1. The queue formula distribution depicts normal distributions while the multicast distribution shows exponential decay distribution. The two lines show little correlation. The plots show two sets of results. The first set of results is the number packet transmitted and the second set is the number of loss packets. Any packet with delay greater than 14ms second is considered loss packet. A packet which spend more than the average time to travel a link is regarded as a loss packet. The multicast plots in all the three cases show how many packets were lost. The rest of the plot recorded zero packet losses.

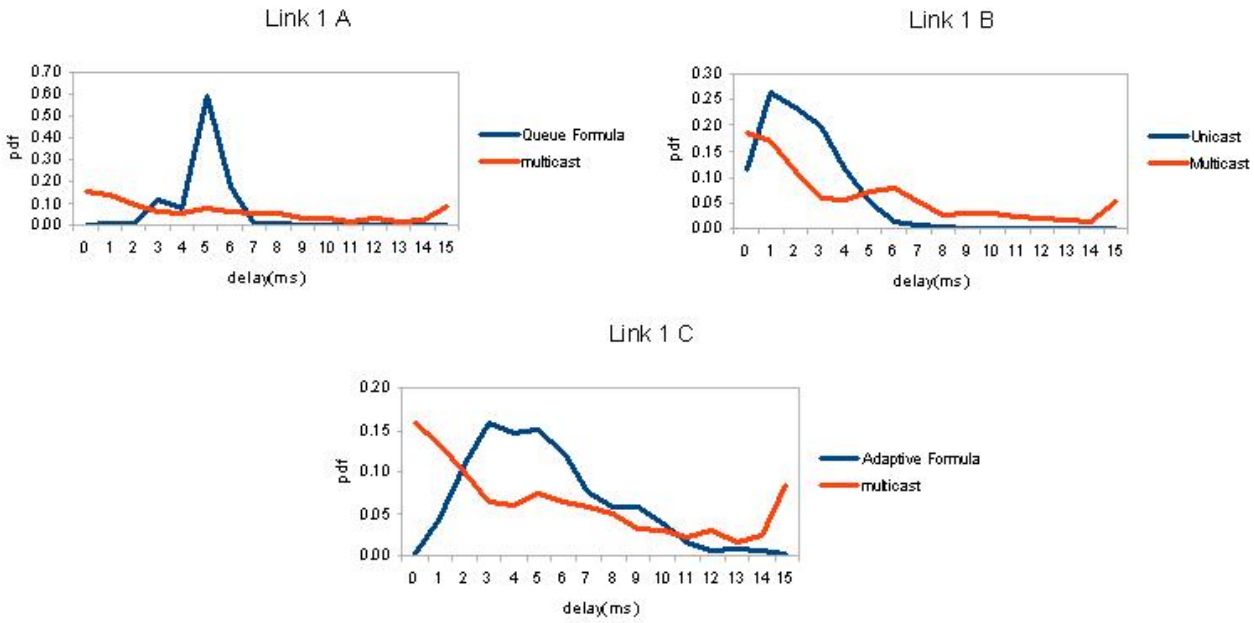

Figure 4.3: *Link 1 Delay*

Figure 4.3, ***(B)*** shows the estimated digitalized delay using the unicast end-to-end measurement and the multicast end-to-end delay on link 1 for the unicast/multicast experiment. Both distributions follow similar trends but their correlation turn to vanish as delay goes below 6ms. The multicast results also show more lost packets than the unicast results.

Figure 4.3, ***(C)*** depicts the estimated digitalized delay using the passive end-to-end measurement calculated with the adaptive formula and the delay estimated by the multicast probes for link 1. Both distributions start apart but converges as the delay goes beyond 6ms. The multicast plot shows more lost packets than the passive plot.

In general there seems to be no or little correlation between each pair of distribution on Link 1.

Figure 4.4 shows the estimated link delay on link 4 for both passive and unicast experiments.

Figure 4.4, ***(A)*** shows the estimated digitalized delay using the passive end-to-end measurement calculated with the queue formula and the delay estimated using the multicast end-to-end delay in link 4. The two distributions are dissimilar. While the delay on the link 4 captured by the multicast probes is exponential in nature the queue formula estimate depicts something similar to a normal distribution. Both plots registered no packet loss.

Figure 4.4, ***(B)*** shows the estimated digitalized delay using the unicast end-to-end

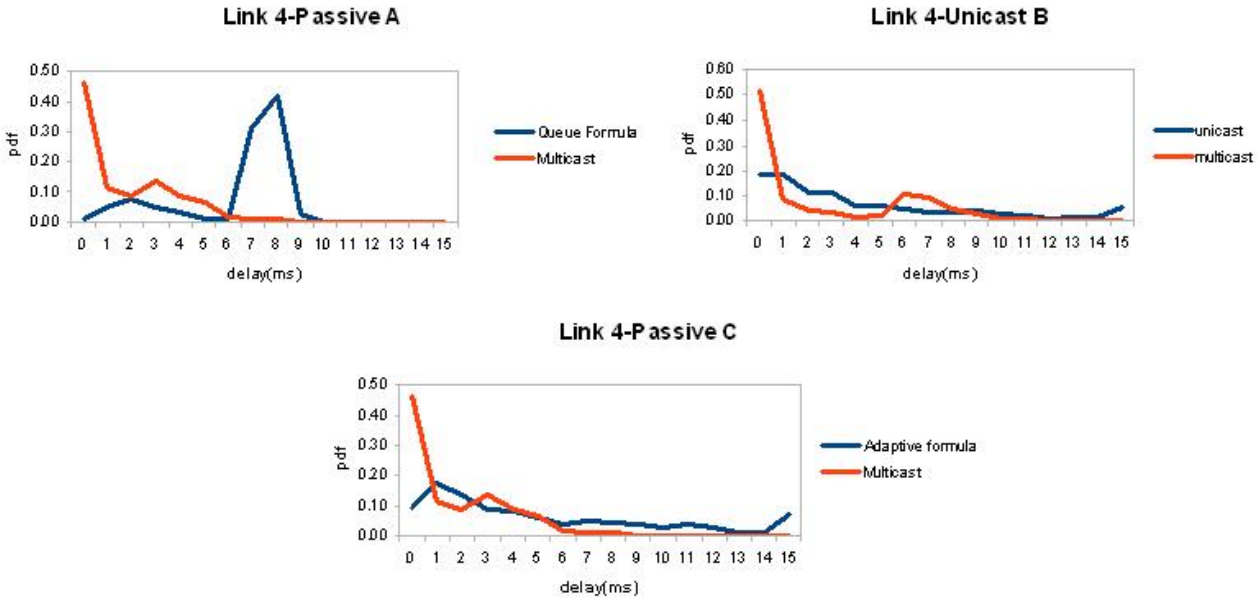

Figure 4.4: *Link 4 Delay*

measurement and the multicast end-to-end delay in link 4 for unicast experiment. The two distributions and their performance are comparable. The unicast shows more loss packets than the multicast and seems well correlated. They start apart at the initial stage but converges with time.

Figure 4.4, ***(C)*** shows the estimated digitalized delay using the passive end-to-end measurement calculated with the adaptive formula and the delay estimated with the multicast probes in link 4 for the passive experiment. The two distributions look similar. Whilst the adaptive model registered losses the multicast registered no packet loss.

In general, apart from figure 4.4 ***(A)*** the rest of the curves looks very much alike and well correlated.

The table 4.2 depicts further comparison of the link delay measurements using correlation **R**. The **R** measures the direction and the extent of correlation between a pair of results. Queue, Adaptive, unicast are predicting the multicast results. Generally, the correlations of link 1 is weaker than correlation of link 2. (see appendixes A,B,C for the rest of the results).

Link 1	R	$R^2(\%)$
Queue/multicast	0.09	0.81
Adaptive/multicast	0.11	1.21
unicast/multicast	0.74	54.76
Link 2		
Queue/multicast	-0.11	1.21
Adaptive/multicast	0.88	77.44
unicast/multicast	0.99	98.01

Table 4.2: *Correlation table of various link measurements*

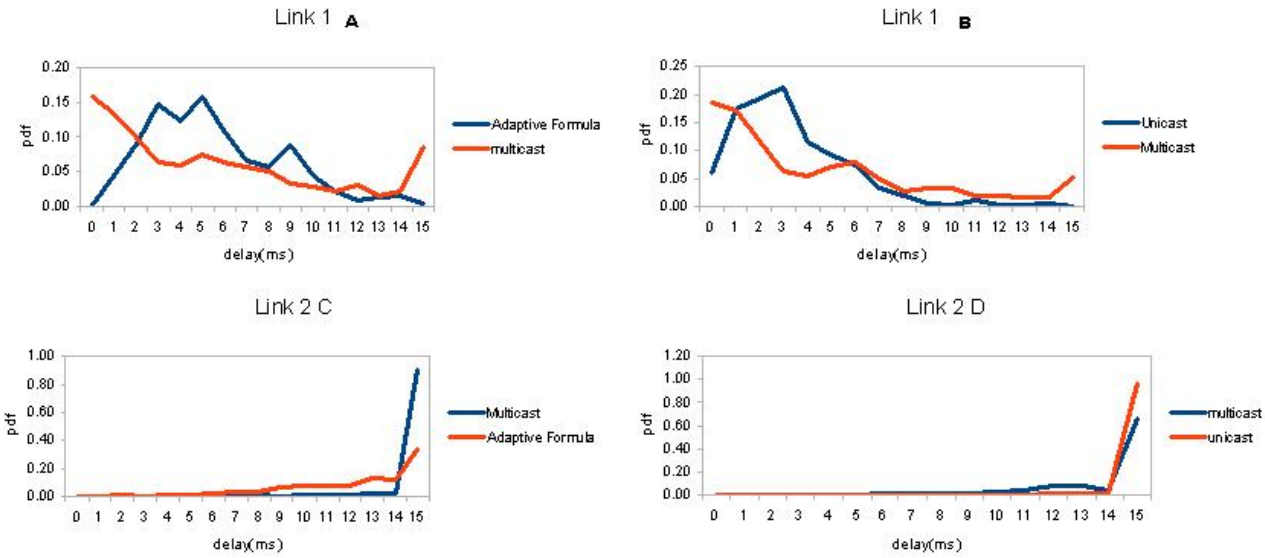

Figure 4.5: *Converge link 1 and 2*

4.1.3 Convergence

Figure 4.5 shows the estimated link delay on link 1 and 2 for both passive and unicast experiments. It depicts how fast the unicast and passive (adaptive model) converges to their respective multicast link delay distributions. We conducted the test using 400 i.i.d for the passive and unicast measure and 1000 probes for the multicast measurements. The goal is to find which of the two method converges fast to their respective multicast results and how the long term probes affect the results.

Figure 4.5, ***(A)*** shows the estimated digitalized delay using the passive end-to-end measurement calculated with the adaptive formula and the delay estimated with the multicast end-to-end delay on link 1. The plot looks less correlated as compare to **B**.

Figure 4.5, ***(B)*** shows the estimated digitalized delay using the unicast end-to-

end measurement and the multicast end-to-end delay on link 1 in the multicast/unicast experiment. It shows better correlation characteristics than **A**.

Figure 4.5, ***(C)*** shows the estimated digitalized delay using the passive end-to-end measurement calculated with the adaptive formula and the delay estimated with multicast end-to-end delay on link 2. The plot shows better correlation characteristics but the adaptive formula is unable to uncover all the losses.

Figure 4.5, ***(D)*** shows the estimated digitalized delay using the unicast end-to-end measurement and the multicast end-to-end delay on link 2 for the unicast experiment. Both plots are comparable and number of losses are close.

Further comparison is done using the absolute differences between variances of the distributions and the results are shown in the table 4.3.

Link 1	variance	Absolute Difference
Adaptive/multicast	0.00/0.00	0
unicast/multicast	0.01/0.00	0.01
Link 2		
Adaptive/multicast	0.01/0.05	0.04
unicast/multicast	0.06/0.03	0.03

Table 4.3: *Absolute difference of variances of various measurements for 400 probes*

The table 4.3 depicts the variances and the absolute difference of variance of adaptive model estimation and inferred multicast estimate for the passive experiment and that of unicast and it respective inferred multicast estimate for the unicast experiment. It compares the variances of 400 i.i.d of passive and unicast experiments to 1000 i.i.d of the respective multicast experiments. The passive delay was estimated with the adaptive formula. On link 1 the table shows better absolute difference of variance for adaptive/multicast experiment. Link 2 shows better absolute difference of variance for unicast/multicast

4.1.4 L_1 Relative Error Norm

The table 4.4 shows various L_1 relative norm measurements of the first three links computed over 50 probes. In Link 1, unicast/multicast pair shows the best L1 relative error norm. The adaptive/multicast pairs means we are comparing the passive delay estimated with adaptive model to the multicast measurement taken from the passive experiment. We computed the error norm using the continuous normalized results. So the L_1 relative norm measures the relationship between the pairs using the continuous data.

Link 1	
Queue/multicast	0.27
Adaptive/multicast	0.31
unicast/multicast	0.25
Link 2	
Queue/multicast	0.92
Adaptive/multicast	0.36
unicast/multicast	0.37

Table 4.4: *a table of various link 1 and 2 L_1 relative norm measurements*

Chapter 5

Discussion

The aim of network tomography is to determine an estimate of what goes on inside a network from the measurements obtained from the end nodes. In order to estimate link delay distribution (from end-to-end delay measurements or distributions) requires intelligent guesswork and algorithms such as maximum likelihood techniques. This chapter discusses the performance of this techniques. We aim to improve the results of network tomography when used together with passive monitoring.

The usual practice in network tomography is to use either unicast packet pairs or multicast probing technique in a tree topology to measure the end-to-end delay. The results of the end-to-end delay is then used to predict the distribution of the individual links of the network. The prediction or estimation of the internal link delay distribution from the end-to-end measurement is possible because we assume that the link delay distribution on a path is the aggregation of the distributions of each of the links that constitute the path. For example in figure 3.1 the aggregation of the distributions of link 0-1, 1-2 and 2-4 should be the same as the delay distribution of the end node 4. We can also say that the sum of the delays experienced by a packet traveling along the path on link 0-1, 1-2 and 2-4 should be the same as the delay experienced by the end node 4.

Besides, if we send a pair of probe packets from a source 0 of the multicast tree (figure 3.1) to node 4 and 5 for example, the packet pairs should experience the same delay on the shared link 0-1 and 1-2. This means the differences in the end-to-end delay of the packet pair is caused by the delay experienced on link 2-4 and 2-5 but not the shared links 0-1 and 1-2. If this condition is true then the internal delay distribution could be estimated from the end-to-end delay measurements or distributions.

The second condition for network tomography to hold is that the probing packets should be independent and identically distributed (i.i.d).

In practice however, only multicast packets are able to satisfy these conditions

fully. Since many networks do not support multicast probing, closely spaced packet pairs are used instead of multicast probing. This packet pair is known as unicast packet pair. The assumption in the unicast packet pair approach is that, when packets are closely spaced they can experience the same delay on a shared link since there will be no intermediate packet between them. In reality however, this is not always true since they can still be separated by intermediate packets. The unicast packet pairs may fail the i.i.d condition if they are not designed carefully. The unicast probing results is credible and accepted in network tomography even though it introduces uncertainty in the end-to-end delay measurements.

Our goal is to extend this unicast idea even further to include passive packet source. The reason for this is that active probing is giving way to passive measurement techniques and also new generation networks such as ANA network do not support probing. Neither unicast nor multicast technique would be suitable for such networks.

We cannot use the end-to-end passive data source directly for the estimation since passive packets are likely to experience different delays on shared links. We need to minimize this uncertainty in order to arrive at a result similar to the unicast method. So we propose two models which we hope could minimize the uncertainties in the end-to-end delay measurements. The two models are the M/M/1 queue model and the adaptive learning model. First of all we collect the passive end-to-end delay and process it with the two models. An example of how this processing is done can be found in section 3.4.3

5.1 Experimental Evaluation

In order to evaluate the performance of our passive method we conducted two separate experiments and labeled them passive/multicast and unicast/multicast. The passive multicast means the experimental setup has passive packet generator from node 0 to the end nodes 4,5,6 and 7. On the same network there is multicast data source which send probes to the same end nodes 4,5,6 and 7. The multicast data source is included to help in estimating the true end-to-end delays so that we can compare it to the passive results. We define passive packet as any packets that travel at random from one end of a network to another.

Similarly, the second experiment also has the same setup but the traffic generators are unicast packet pairs and multicast probing packets. We define unicast packets pairs or data source as a set of packet pairs that may experience the same or similar delay on a shared link. Each experiment produces two sets of end-to-end delay measurement. So we had four set of data in the end for the link delay distribution estimation. We then pair them as passive/multicast and unicast/multicast. Passive/multicast means the experiment has multicast traffic generator that measures the actual end-to-end delay

and at the same time a passive traffic generator that sends packets at random on the network in addition to the other background traffic.

"Unicast/multicast" means the experiment has multicast traffic generator that measures the actual end-to-end delay and at the same time we have an additional closely spaced packet pair traffic generator that sends packets at on the network in addition to the other background traffic.

The reason for combining multicast data source with each of the two experiment is to enable us measure the true end-to-end delay for comparison.

We compare the performance of the individual experiment to their respective multicast results. We also compare the performance of the two separate experiments to ascertain whether passive data source approach is possible. We will use "unicast" to refer to unicast/multicast experiment and "passive" for passive/multicast experiment.

The passive end-to-end delay is not used directly but we try to reduce the uncertainty in the measurement by applying the adaptive learning model and queue model on them. We will refer to these models as "adaptive" and "queue" respectively. This means the passive experiment will have three sets of results for comparison. They are the multicast results, adaptive results and the queue results. The unicast experiment consists of two results, they are the unicast packet pair results and multicast results.

5.2 Performance Evaluation

The internal delay is estimated using Maximum Likelihood Expectation (MLE) (see section 3.2.5). The MLE is a logarithmic function with either exponential decay or growth in nature. We expect our estimation to have the property of the likelihood function. When probing packets are not independent and identically distributed, the MLE estimator usually fails to exhibit the exponential properties [31]. MLE is also asymptotic meaning the curve gradually approaches either the x or the y axis but never crosses the line. Following this, we expect all the link delay distributions to be exponential in nature.

However, Link 1 shows a strange plot uncharacteristic of the MLE (see figure 4.3). The unicast, adaptive and the queue plots all depict tailed distributions instead of expected exponential distribution. This could be attributed to the fact that the probes were not i.i.d. Link 1 carried more traffic than any of links. Link 1 has seven traffic sources while the others have at most four traffic sources.

However, the multicast estimates show that only few packets were loss on link 1 (see figure 4.3). Specifically, only 52 packet losses were estimated by the multicast probe in the unicast/multicast experiments. For the passive/multicast experiments only 85 packets losses were estimated. This suggests that link 1 was not congested. On the

contrary over 600 packet losses were estimated in link 2 (see figure 6.1). Packet loss here means, a packet spent more than expected time to travel a link. So link 2 was more congested than link 1.

So the distortion in link 1's distributions might not have been caused by traffic congestion. Thus, congestion might not be the cause of the unusual behavior of the packets. It seems packets were flooded onto node 1 by node 0 at a rate such that the probing packets were dependent on each other. The propagation delay between node 0 and 1 is only 10ms. Since the rate was too much for node 1, it decided to drop a lot of packets.

We expect the multicast results on link 1 to be the same as the others since they all travel on the same link but the plot shows otherwise. The multicast plots follows the characteristics of MLE in all the plots in figure 4.3. The multicast probes seem unaffected by the possible flooding because only one packets travels at a time from node 0 to 1. In multicast probing only a single packet is sent from source and copied to a group of receivers when the links to the destination split. Since there is only one packet sent from node 0 to 1 at a time the packets would mostly be independent.

The shape of the unicast plot has better MLE characteristics than both queue and adaptive models but still not as good as the multicast plot. The multicast inference seems more robust than any of the methods. Apart from the multicast plot the rest are not likely to converge to the true link delay distribution. They are distorted by the activities of node 0.

We now compare the correlation results of link 1. Our expectation is to find a better correlation between the unicast and its respective multicast results than the adaptive and the queue model since the adaptive and queue may contain uncertainties. From the table 4.2 the R^2 value for unicast/multicast experiments is 54.76% representing the highest correlation score follow by the adaptive/multicast results. The queue/multicast shows virtually no correlation. The unicast and the adaptive produced expected results but queue model's result is quite unexpected. This may be due to the effect of end-to-end delay estimation. Considering the end-to-end delay plots in figure 4.3, the queue end-to-end delay estimation is quite odd. The queue end-to-end delay lies in the middle of the plot with almost no sparks. The queue model seems to be averaging the observed delays. However, the adaptive model is quite consistent with the multicast delay.

We also compare the convergence of the passive and unicast methods for link 1. Convergence determines the behavior of the link distribution in a short term against the long term behavior. We compare the link delay distribution of link 1 estimated with 400 probes to the multicast link delay distribution. The multicast delay was estimated with 1000 probes. We will do the same to the passive method and compare the two results. The ones with strong correlation or small absolute variance may converge faster than the other. Our expectation is that the unicast and the adaptive model would

have similar results.

From table (6.2) we found out that the passive and the unicast model have different coefficient of correlation. The adaptive model has the correlation coefficient of 0.04 whilst the unicast model has correlation coefficient of 0.6. This suggests that the unicast converges faster than the passive model. However, using the absolute variances for the convergence rate, we found out that the adaptive model has better absolute variance of 0 (see table 4.3) whilst the unicast model has absolute variance of 0.01. The absolute variance may not be a good estimator as compared to correlation coefficient so we may conclude that the unicast model may have better convergence rate than the passive model.

The next test for link 1 is the L_1 relative error norm test. The tables are found in (4.4, 6.4). The L_1 relative error norm test is used to compare distributions. The previous correlation coefficient comparison was computed with discrete or digitalized distributions. The digitalization introduces error in the measurement. We performed the L_1 relative error norm measurement on the continuous distributions of the various experiments in order to measure the correlation of the normalized continuous data. Also it would be able to roughly estimate the impact of the digitalization on the results.

Our expectation is that the correlation coefficient between a pair of models and their equivalent L_1 relative error norm to be similar or consistent. If they are different then the difference could be attributed to the effect of digitalization or quantization of the continuous data. When they are similar, it means no exact conclusion could be drawn except to say that the result is consistent with the correlation results. A small L_1 relative error norm is better than large value. We use "norm" to represent L_1 relative error.

From the table 4.4, link 1 shows that unicast/multicast pair has the best norm followed by queue/multicast, adaptive/multicast in that order. Our expectation is that the unicast and the adaptive model may perform better than the queue model since the queue exhibit poor correlation. However, the queue estimate unexpectedly performed better than the adaptive estimate. The queue/multicast is better than the adaptive/multicast and significantly close to the value of unicast/multicast. It is not immediately clear what might cause this behavior, however, considering the results of other links, the good norm value may be caused by a mere coincident rather than actual performance. The queue model produces different non-MLE distribution in all the links. This abnormal behavior could contribute to the abnormal results rather than the possible effect of digitalization.

The norm for unicast and the adaptive models are consistent with their respective correlation coefficient values (see figure 4.4). The norm value confirms that the adaptive/multicast pair may be less correlated than the unicast/multicast pair.

The link delay distribution plots shows two things. They are the probability of

packets that experienced various delays and the loss packets. Any packet with delay above 14ms is considered loss packets. We expect the plot to show similar number of loss packets in link 1 but apart from the multicast results, figure 4.3 shows that none of the models was able to estimate the loss packets on the link. They all show zero loss packets in link 1. The dependence between the packet pairs might have affected the efficiency of the other models.

Again in link 1 the maximum delay experienced by the unicast pair is 9ms. Most of the packets experienced a delay of 1ms. This means the unicast packet pairs traveled at a very high speed.

Now in link 2 our expectation is that the adaptive model and unicast model would have similar correlation properties with their corresponding multicast counterparts. The traffic situation on the link was estimated to be high (see figure 6.1). More packets were dropped on link 2 and 3 than any other link. This means during the period of measurement more packets including the passive packets were flooded to node 1 by node 0 causing congestion on link 2 and 3. Since passive packets were also flooded as discussed above, they might have been buffered on link 2 and 3 and sent together causing them to experience similar delays on the shared link 2 and 3. For this reason, our expectation is that the correction between the multicast and the adaptive model would be strong. The results of the queue model could go either ways. We don't expect the unicast model to perform too well since the congestion may introduce intervening packets causing a packet pair to experienced different delays.

We proceed by measuring the behavior of the two experiments. From figure 4.2 our finding shows that the queue/multicast exhibits weak correlation. The correlation for unicast/multicast in link 2 is better than that of adaptive/multicast. However, both exhibit strong correlation with their respective multicast results. The exceptional performance of the unicast model may be due to the closeness of the packet pairs. Since the time space between packet pairs is only 0.1ms it may not have allowed room for intervening packets in link 2.

We now consider the norm values. The norm values may give an idea about the effect of digitalization on the results. In nominal terms, we expect the norm values to be consistent with the correlation value of the discrete distribution. From table 4.2 the correlation coefficient of the unicast model is 0.99 and that of adaptive model is 0.88. However, their norm values are 0.37 and 0.36 respectively (see table 4.4). The difference might be the effect of digitalization. The error norm was computed on the continuous data while the correlation results comes from the discrete data. Since the norm and coefficient of correlation all measures the relationship between two distribution and they are similar in nominal terms, small difference could be attributed to the effect of digitalization. In both cases the queue model shows very poor results.

The difference of two variances could tell the extent to which the two distributions converge. If two distributions are similar then their variances are likely to be the same

or similar. The exact relationship between two variances is usually determined using standard statistical tables. Here we use the variances to roughly guess the rate of convergence of two distributions. The smaller the absolute difference between two variances the better the rate of the convergence.

We now compare the variance of 400 probes of the adaptive model to the variance of 1000 probes of the multicast model. We do the same to the unicast model and find which of them have smaller absolute difference. Absolute difference is the difference between two variances without a negative sign. The one with smaller difference converges faster than the one with larger difference. This is not an absolute rule but could help to roughly estimate the convergence rate.

Table 4.3 shows the absolute differences of variances of link 1 and link 2. Note that link 1 means the path from node 0 to node 1. The absolute difference value for unicast is 0.03 and that of adaptive is 0.04. This means for a short term probe, unicast method would converge faster to it multicast delay than the adaptive approach.

The queue model is dropped from this test because of it erratic behavior. When we examine the traffic build up on link 1 closely, we found out that most of the packets were sent within the first three time slots with delay between 0 and 2ms (see figure 4.3). This means the traffic build up was very high at the initial stages and gradually slowed down with time. Also similar number of packets where transmitted by the passive model within similar period. This suggests that the passive data source transmitted continuous stream of data within that short period before it became accustomed to it random pattern. The closeness of absolute difference of variances may be caused by design fault or the number of probes used in the measurements rather than an indication of rate of convergence. It may also be cause by the behavior of the random traffic generator. The exponential on/off could send continuous stream of packets for sometime before it goes off. When the passive packet generator generates constant stream of data the behavior of the packets would be similar to the behavior of the unicast packet pairs. This may explain why the results are quite close. It would also explain why the passive plot in link 1 was not exhibiting the characteristics of MLE. They were not i.i.d.

When we consider variance of 400 probes versus 1000 multicast probes for both experiment in table 4.3 we found out the the adaptive model is less correlated than the unicast model. So both the correlation coefficient and the variance are consistent.

We now consider the packet loss estimation on link 2. In figure 6.1 the plots show a similar packet loss estimation in the unicast/multicast experiment. However, the passive experiment has a different results. The adaptive method estimated 345 packet losses while its multicast counterpart estimated 912 packet losses on the same link. The difference between the two is over 50%. For the unicast model the difference in loss estimation is quite close. It was 655 for multicast and 928 for unicast. The packet loss estimation by the adaptive model was far from the true estimation by it multicast

model.

We found a similar situation in link 3. The congestion on the link is estimated to be high. The multicast probe for the passive experiment estimated a loss of 681 packets out of 1000 probes on link 3 (see figure 6.1). When we consider the correlation coefficient of the two experiments from table 4.2, the results show that the adaptive model has a better coefficient of correlation than the unicast model. It shows 100% correlation while the unicast model shows 77.44% correlation. The convergence rate between the two is also consistent with the correlation. The absolute difference of variance for adaptive model is 0 and that of unicast is 0.05 (see table 4.3). This suggests a better performance for the adaptive model. It may be due to the same reason that the congestion on link 3 might have contributed to the good performance.

We now turn our attention to the error norm of link 3 to find out how digitalization may affect the results. From the norm table (6.4), the error norm of adaptive model for link 3 is far smaller than the error norm of the unicast model. This means that the results of the error norm is consistent with correlation results.

Considering the convergence rate of link 3 from the tables 4.3 and 6.2, both the coefficient of correlation and the absolute variance of the adaptive model are better than the unicast model. So the adaptive model may converge faster than the unicast model.

For the rest of the links (4,5,6,7) the correlation of unicast/multicast dominates the other two pairs. In most cases the queue/multicast performs poorly. Links 4,5,6 and 7 are the outer links where the congestion is quite low. The multicast estimates in both experiments show that there were no packet losses on those links (see figure 4.4, 6.3, 6.4, 6.5). This means congestion might be less on those links. We observe that when congestion is less the unicast model out performs the adaptive model. The correlation coefficient of the adaptive model is quite low below the 0.7 threshold in link 4 and 5. The queue model performance was very poor and never above 0.5 coefficient of correlation (see table 6.1). Nevertheless, the performance of the adaptive model on link 6 and 7 was above the 0.7 threshold.

The adaptive model estimated higher packet loss on these four links than any of the models. The absolute differences show that the convergence rate of the links 4,5,6 and 7 are the same but their correlation tells quite a different story. The correlation table 6.2 shows that the adaptive model converges at a faster rate than the unicast model in links 4,5,6,7. We observed that the first 400 packets arrive within the first three time slots and declined slowly. In the adaptive estimate the first 400 probes arrived within the 0 and 3 ms. The rate of transmission might have caused the very good performance at the initial stage (see figure 4.4, 6.3, 6.4, 6.5).

The unicast model has a better L1 relative error norm in links 4,5,6,7 than the adaptive model. The queue model is trailing in most of these cases.

The queue/multicast has a very strong norm in link 7 than the rest of the pairs. Notwithstanding, their correlation coefficient is week compare to the other two remaining pairs.

We observed that in most cases where the loss rate of a link is high the adaptive performs much better than the unicast model. This is because congestion may have caused unicast packet pairs to experienced quite different delays on shared links.

We can use the convergence rate to also determine the effect of large number of probes. We do that by examining the correlation between the 400 probes and the 1000 probes experiments. In figure 6.2, 4.2, 6.1, the effect of short term probe were felt in link 2,3 and 7 for adaptive/multicast experiments and in link 2,4,5,6 and 7 for unicast/multicast experiments. The difference in link 3 is 0.01 which is quite insignificant. In most cases the short term probes have stronger correlations than the longer term probes.

From figure (6.1 and 6.2) the behavior of the shared links seems very related. The shared links 2 and 3 have more traffic but bigger capacity as compared to link 1. By looking at the plot they seems more correlated than the rest of the links. The adaptive model has 100% correlation with it multicast distribution. The unicast is also strongest on link 2. This suggests that both unicast and adaptive models may react in a similar manner to congestion. The adaptive model performs better when there is more traffic but not too effective when the traffic intensity is less. The unicast performance depends on how the unicast packet pairs are designed. The difference of 0.1ms between packet pairs in this experiment may be too closed to exhibit perfect characteristics of i.i.d packets, however it may boost it performance when there is a lot of congestion.

5.3 Performance unicast/multicast versus passive/multicast

Here we summarize the general performance between the two set of experiments. The unicast/multicast experiment out performs the adaptive/multicast experiment in the effect of short term probes test by scoring over 80% of the performance test.

The passive experiment was found to be susceptible to quite low traffic, the adaptive model performed quite creditably than the queue which failed most of the test.

We found a strong correlation in four out of the seven links for the adaptive/multicast experiment whilst the unicast/multicast had five strong correlation out of the seven links. In most cases the unicast/multicast has stronger correlation than adaptive/multicast experiment. Here strong correlation means $R >= 0.7$. The seemingly poor performance may be caused by the inability of the adaptive model to estimate similar end-to-end delay as in multicast probe. As shown in table 4.1 the absolute variances of the multicast/adaptive end-to-end measurement and multicast/unicast experiment are

inconsistent with each other. This is not a good sign for two results which are suppose to exhibit similar characteristics. The variances of the adaptive model is wide spread as compared to that of the multicast.

The individual variances of any of the pairs are not in any way close to each other. However, considering the the unicast variances and the adaptive variances we can observe quite similar variances. On the surface the absolute differences could tell how the two pairs compare in their variances. Though the absolute differences look similar there is still significant differences between the two pairs.

The queue model pairs are quite odd. Relatively, they have small variances but their absolute variances with the multicast are far from the rest of the two pairs. This reveals that the queue is not likely to exhibit good performance.

For the normalized continues data the L_1 relative error norm (see table 4.4, 6.4) was found to be small in most cases for the unicast/multicast experiment. The unicast/multicast has four smaller error norms than the adaptive/multicast experiment. It also performed better against the queue model.

Finally, the adaptive proves to be quite a better a model in inferring the link delay distribution using passive data source than the queue model but further investigation will be required to ascertain how network dynamics affect the model and whether we are able to infer the true internal delay distribution from this approach. The number of probes seems to be too small and long term probes could have produced better results.

Chapter 6

Conclusion and Future Work

The possibility of estimating link delay distributions on tree-shaped networks under the assumption of temporal and spatial independence was considered. The internal link delays were estimated with discrete approach. Our method estimates internal link delays with least square minimum norm approach using the continuous end-to-end delay. The results were rounded off and digitalized with a bin size of 1 and number of bins of 15 from zero to 14. The bin size 15 were considered to be the infinite, which represent the dropped packets.

We conducted two separate experiments passive/multicast and unicast/multicast experiments with the multicast/unicast serving as bench mark experiments. The end-to-end delay was estimated using the M/M/1 queue model and the Adaptive learning model.

We compared the correlation properties and the behavior of the link delay distributions. Our method used correlation coefficient as one of the performance metrics to measure the correlation between each of the passive/multicast and unicast/multicast pairs. We then compared results.

Our method introduced L1 relative error norm to examine the properties of the continuous link delay distribution data. We normalized the estimated link delays and the error norm computed without digitalization over 50 probes. This enabled us to observe how the continuous link delay distributions compared in unicast/multicast and passive/multicast pairs. We then evaluated the performance of the passive model using the L1 error norm and draw conclusions.

We analyzed the performance of each experiment and model in section (5.2, 5.3). From section (5.2, 5.3) we can make the following general conclusion, the performance of the passive experiment lies in the middle. It was able to pass some of the performance evaluation test and failed others. The unicast model outperformed the passive model. The queue model failed most of the test. We also observed that the

adaptive model performed well when the traffic intensity was high. We observed that short term probes results may be affected by the design error.

The goal of this work is to investigate the possible ways of extending network tomography in passive network such as ANA network and also initiate the discussion on how to use passive measurement to improve on the unicast packet pair results. One of the benefit of this work is to provide an insight into how passive network tomography could be achieved. It has evaluated the efficiency of adaptive learning approach and queue model on fairly balanced scale. The use of passive tomography may have direct application in vehicular traffic model. The delay observed at the end by passive means could help estimate the internal dynamics of a system where probing may not be applicable. It may help to reconsider the idea of using complex queue models which are expensive to compute in traffic theory.

The method is also useful for IP network since passive measurement is currently gaining root. We show that simple M/M/1 queue model is not likely to uncover the end-to-end delay measurements in a manner similar to unicast method in network tomography. We also found out that the adaptive model performance in network tomography delay inference is good when the traffic intensity of a link is high (see section 5.2).

Future work should investigate more on the adaptive learning model in the light of end-to-end delay estimation from passive data source. The best way to achieve better result and how network dynamics such as varying window size may affect adaptive learning algorithm. To investigate more on the effect of high and low traffic intensity affect adaptive learning model and also to consider real network instead of simulation.

The application of passive network tomography approach to traffic theory will be an interesting area of research. Above all, the feasibility of the passive network tomography in autonomic networks such as ANA may be an important area of research that need attention. Furthermore, how ANA network can estimate quality of service parameter such as link delay distribution need to be looked at.

Appendix A

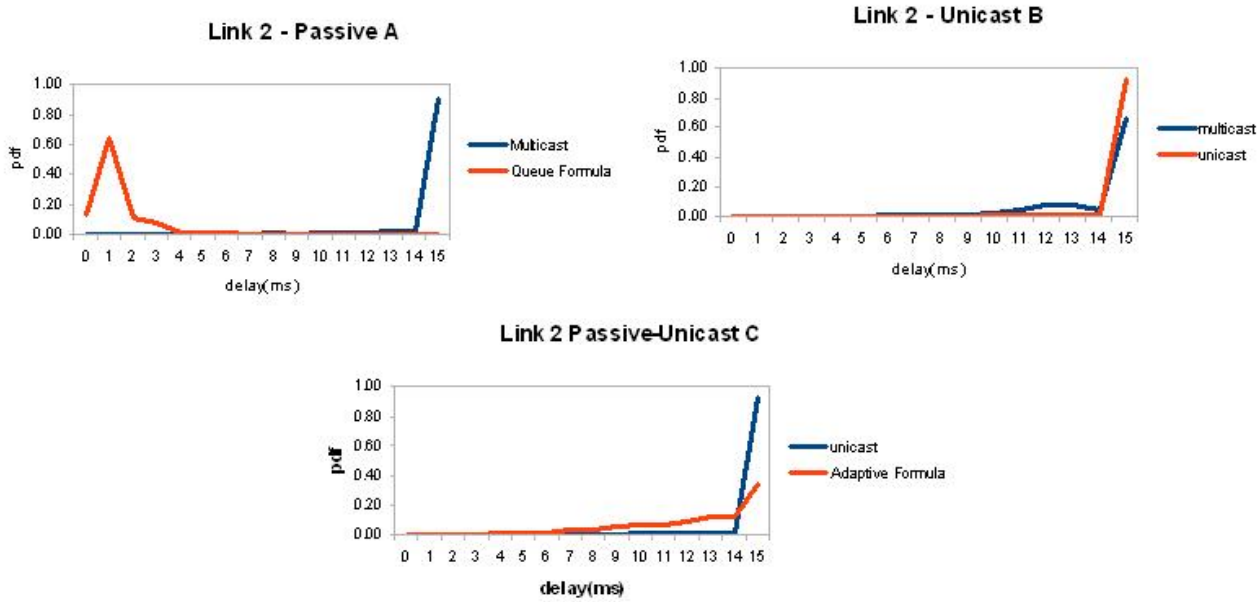

Figure 6.1: *Link 2*

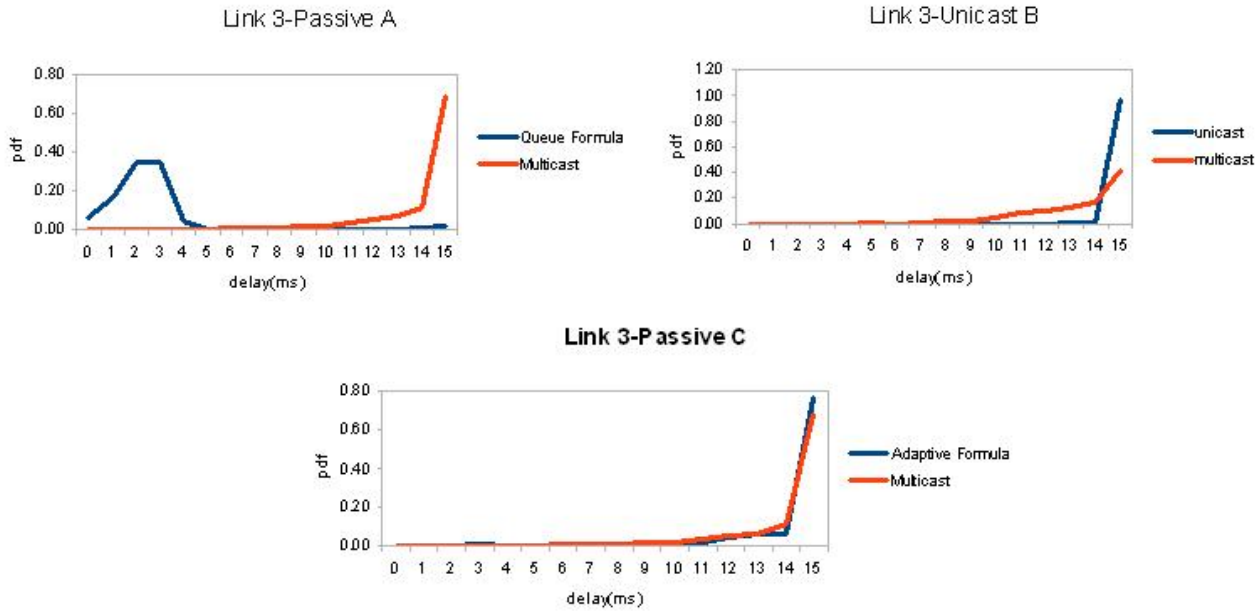

Figure 6.2: *Link 3*

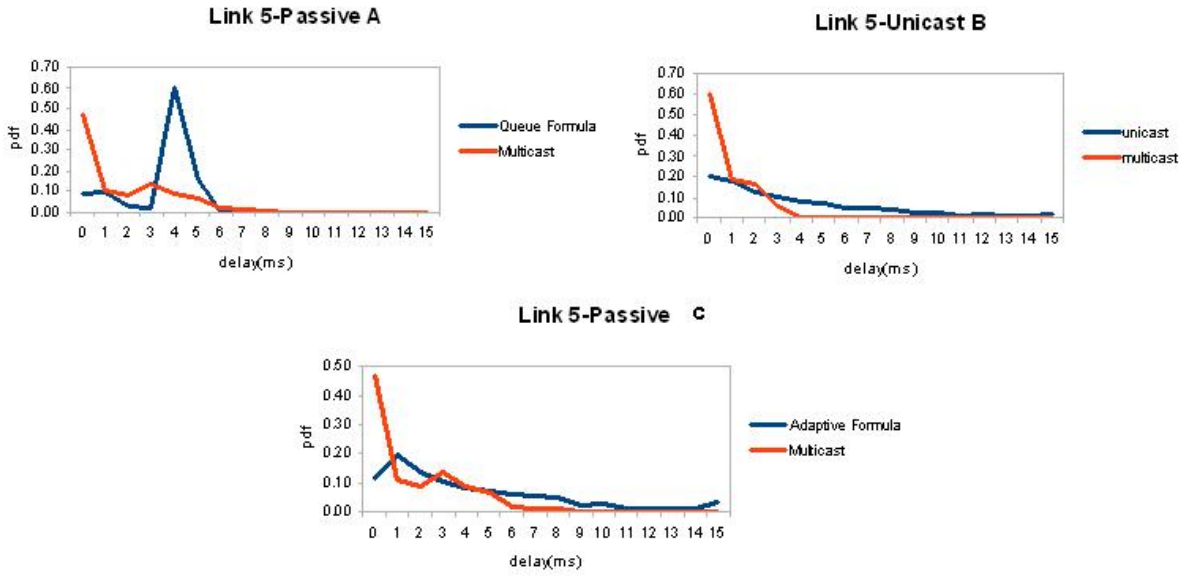

Figure 6.3: *Link 5*

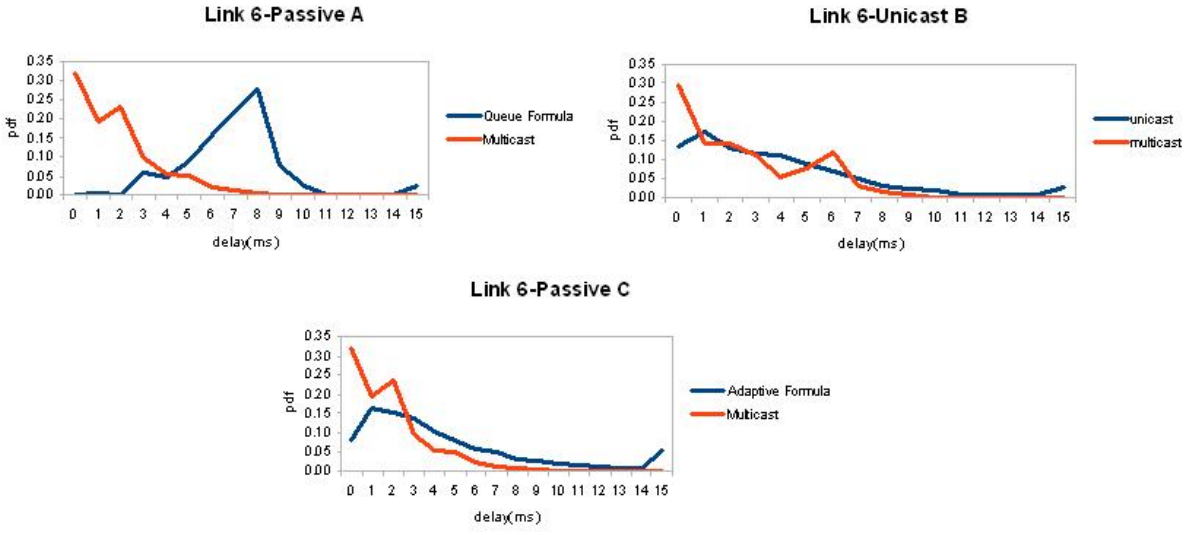

Figure 6.4: *Link 6*

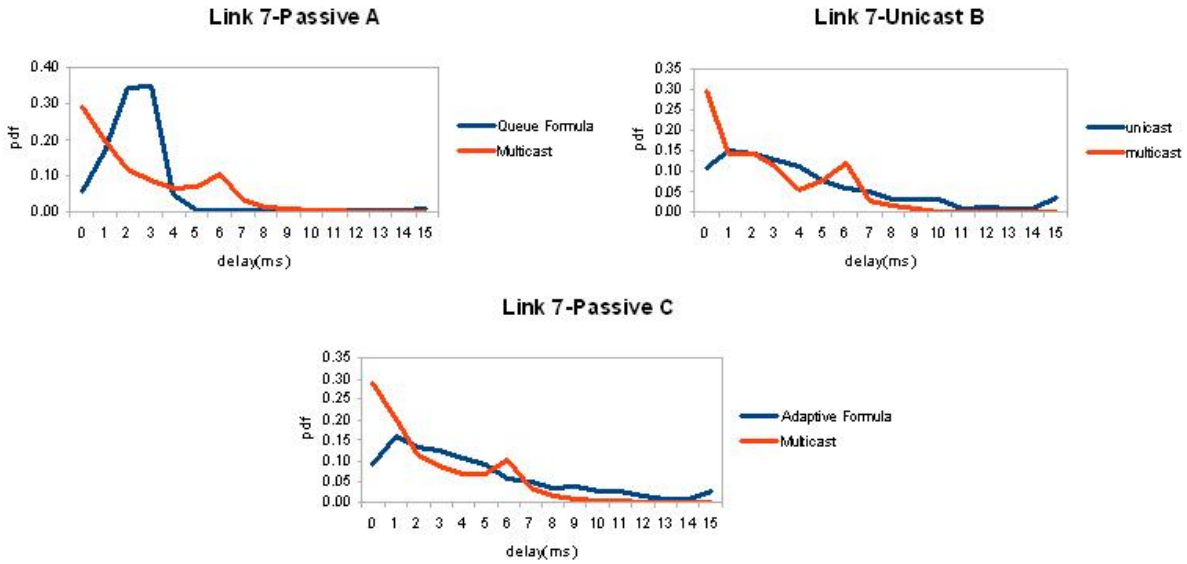

Figure 6.5: *Link 7*

Link 3	$R^2(\%)$	R
Queue/multicast	3.24	-0.18
Adaptive/multicast	100	1.00
unicast/multicast	77.44	0.88
Link 4	$R^2(\%)$	R
Queue/multicast	1.44	-0.12
Adaptive/multicast	22.09	0.47
unicast/multicast	43.56	0.66
Link 5	$R^2(\%)$	R
Queue/multicast	4.84	0.22
Adaptive/multicast	31.36	0.56
unicast/multicast	68.89	0.83
Link 6	$R^2(\%)$	R
Queue/multicast	9	-0.3
Adaptive/multicast	50.41	0.71
unicast/multicast	68.89	0.83
Link 7	$R^2(\%)$	R
Queue/multicast	17.64	0.42
Adaptive/multicast	53.29	0.73
unicast/multicast	57.76	0.76

Table 6.1: *Correlation table of various measurement for 1000 probes measurements*

Link 1	R
Adaptive/multicast	0.04
unicast/multicast	0.60
Link 2	R
Adaptive/multicast	0.89
unicast/multicast	0.99
Link 3	R
Adaptive/multicast	1
unicast/multicast	0.87
Link 4	R
Adaptive/multicast	0.36
unicast/multicast	0.71
Link 5	R
Adaptive/multicast	0.48
unicast/multicast	0.89
Link 6	R
Adaptive/multicast	0.69
unicast/multicast	0.86
Link 7	R
Adaptive/multicast	0.79
unicast/multicast	0.84

Table 6.2: *Correlation table of various for 400 probes measurements*

Appendix B

Figure 6.6: *Converge 3 and 4*

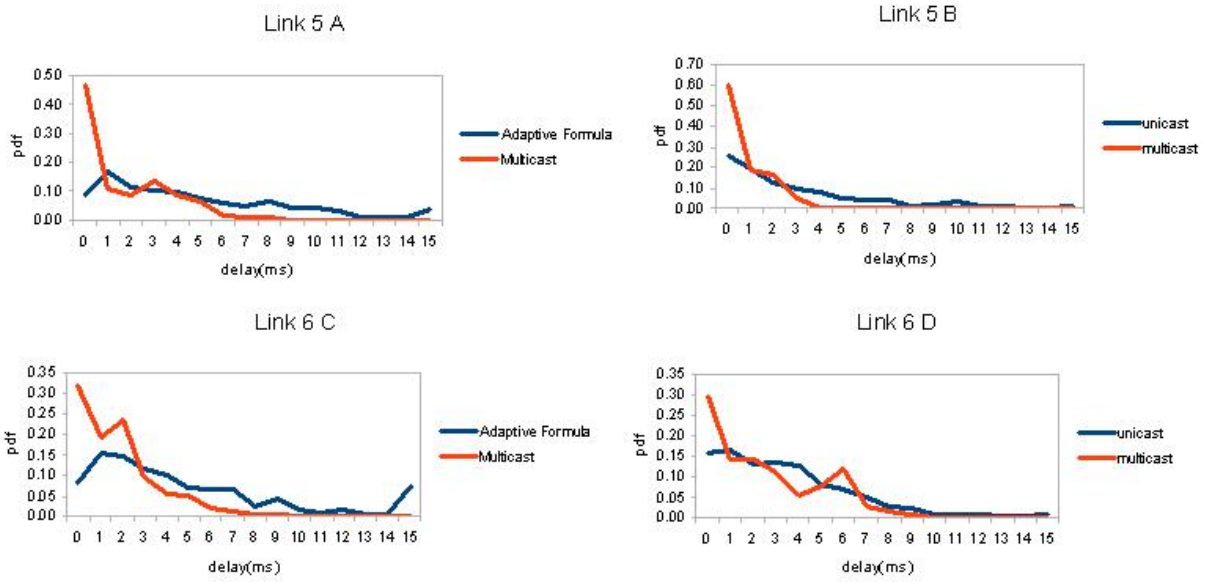

Figure 6.7: *converge 5 and 6*

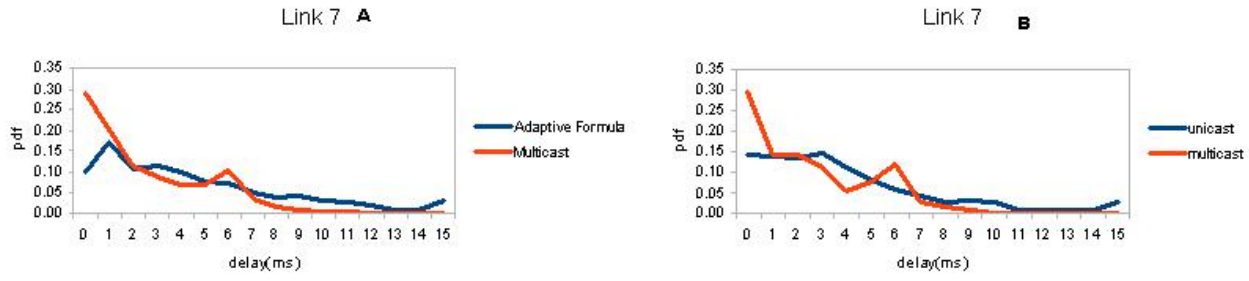

Figure 6.8: *converge 7*

Link 3	Variance	Absolute Difference
Adaptive/multicast	0.03/0.03	0
unicast/multicast	0.06/0.01	0.05
Link 4		
Adaptive/multicast	0.00/0.01	0.01
unicast/multicast	0.01/0.02	0.01
Link 5		
Adaptive/multicast	0.00/0.01	0.01
unicast/multicast	0.01/0.02	0.01
Link 6		
Adaptive/multicast	0.00/0.01	0.01
unicast/multicast	0.00/0.01	0.01
Link 7		
Adaptive/multicast	0.00/0.01	0.01
unicast/multicast	0.00/0.01	0.01

Table 6.3: *Absolute differences of variances various measurements for 400 probe experiments*

Appendix C

Link 3	
Queue/multicast	1.02
Adaptive/multicast	0.08
unicast/multicast	0.61
Link 4	
Queue/multicast	3.14
Adaptive/multicast	2.51
unicast/multicast	0.83
Link 5	
Queue/multicast	1.53
Adaptive/multicast	1.76
unicast/multicast	4.14
Link 6	
Queue/multicast	3.35
Adaptive/multicast	1.76
unicast/multicast	0.59
Link 7	
Queue/multicast	0.38
Adaptive/multicast	0.89
unicast/multicast	0.78

Table 6.4: *Norm table of various measurements computed over 50 probes*

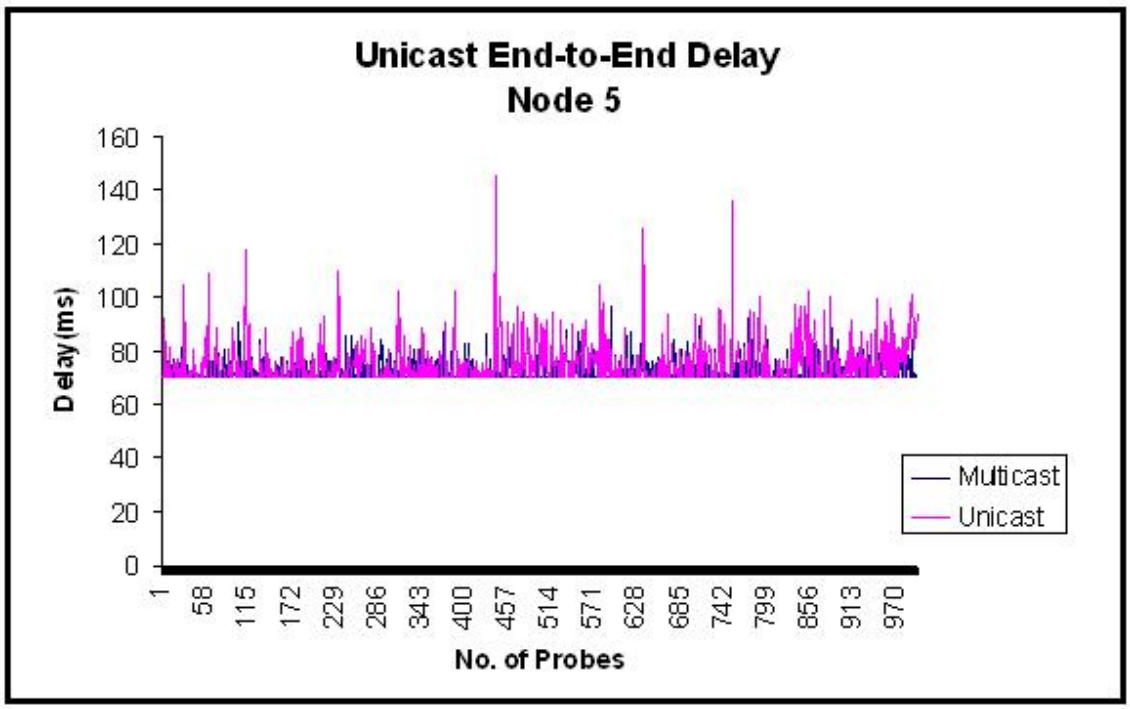

Figure 6.9: *End-to-End Delay Node 5 Unicast*

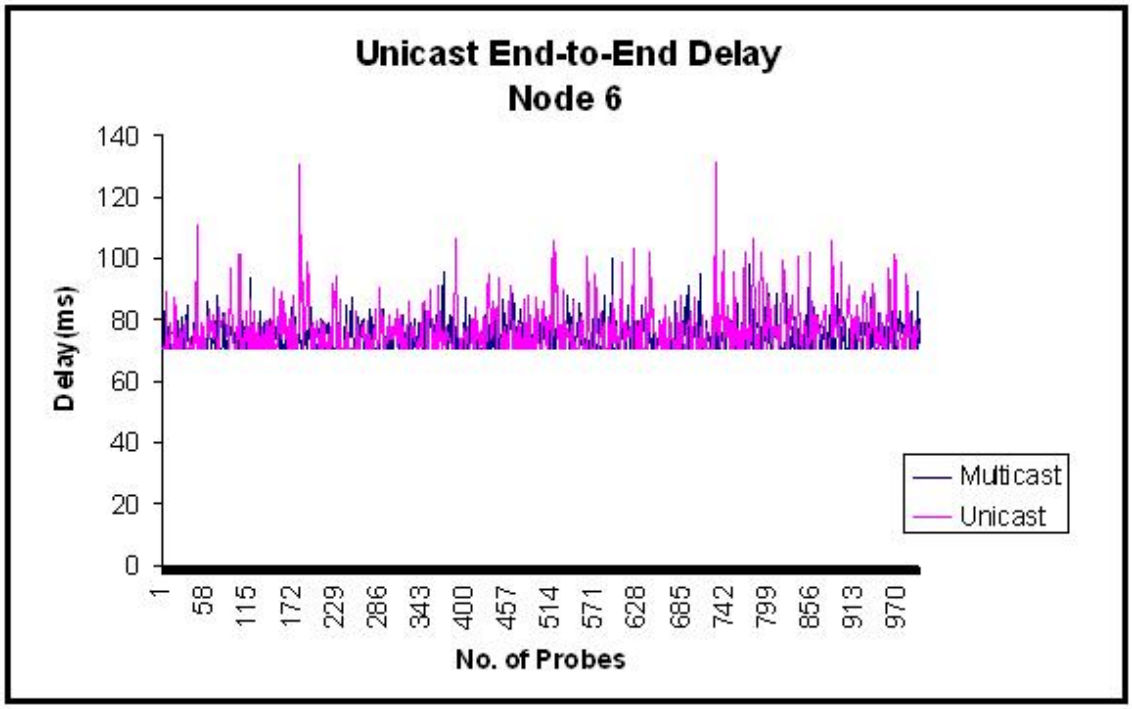

Figure 6.10: *End-to-End Delay Node 6 Unicast*

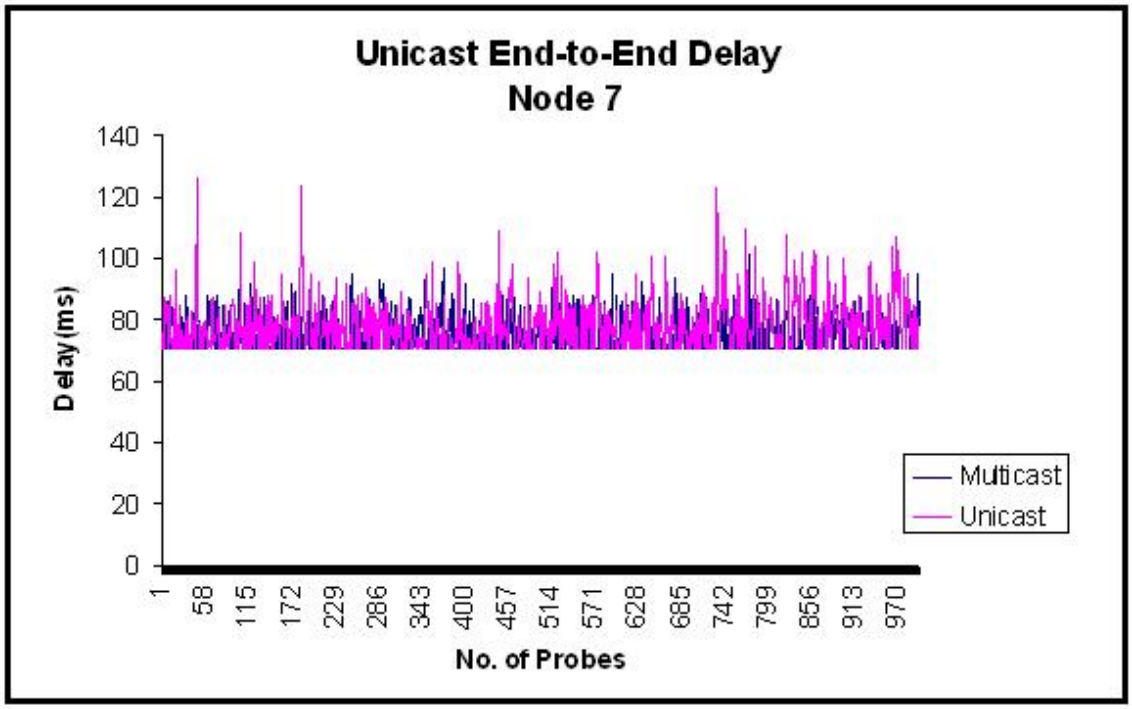

Figure 6.11: *End-to-End Delay Node 7 Unicast*

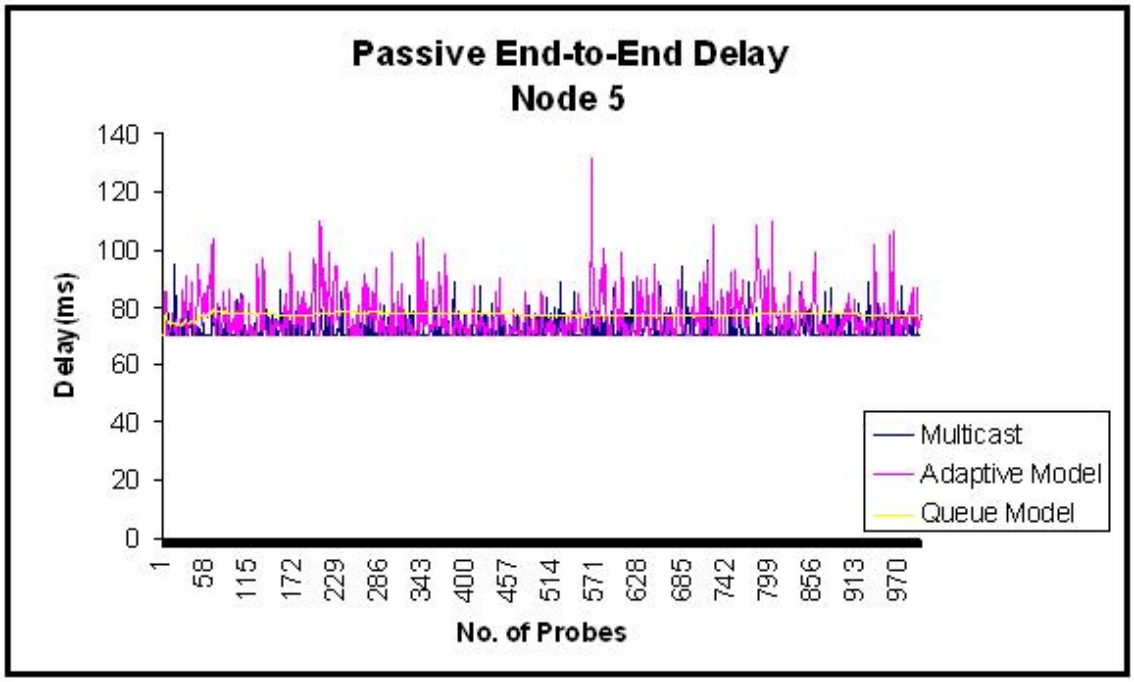

Figure 6.12: *End-to-End Delay Node 5 Passive*

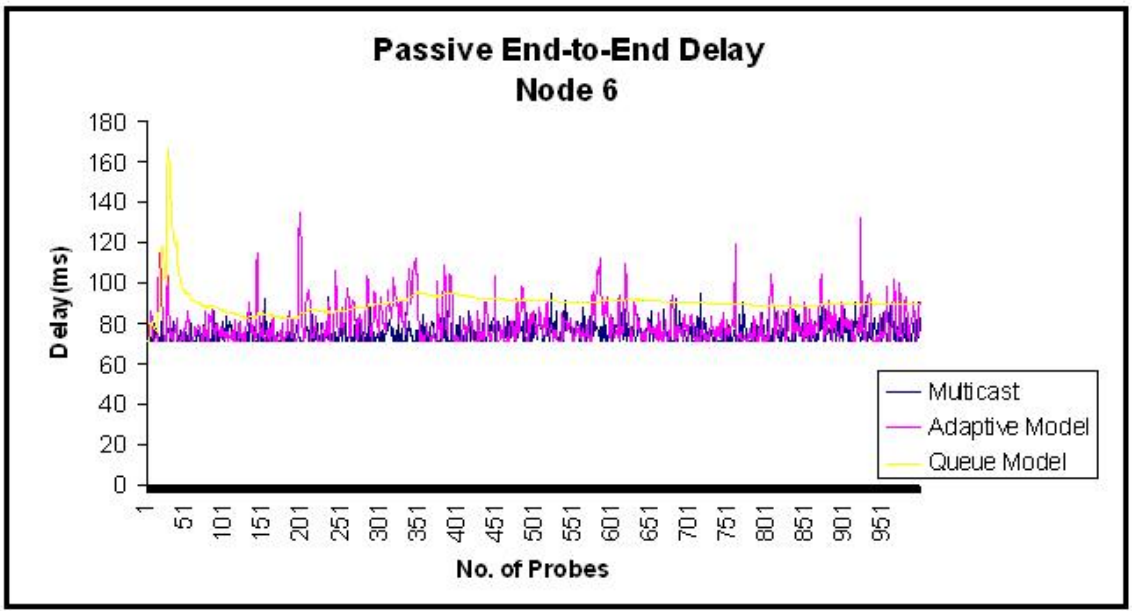

Figure 6.13: *End-to-End Delay Node 6 Passive*

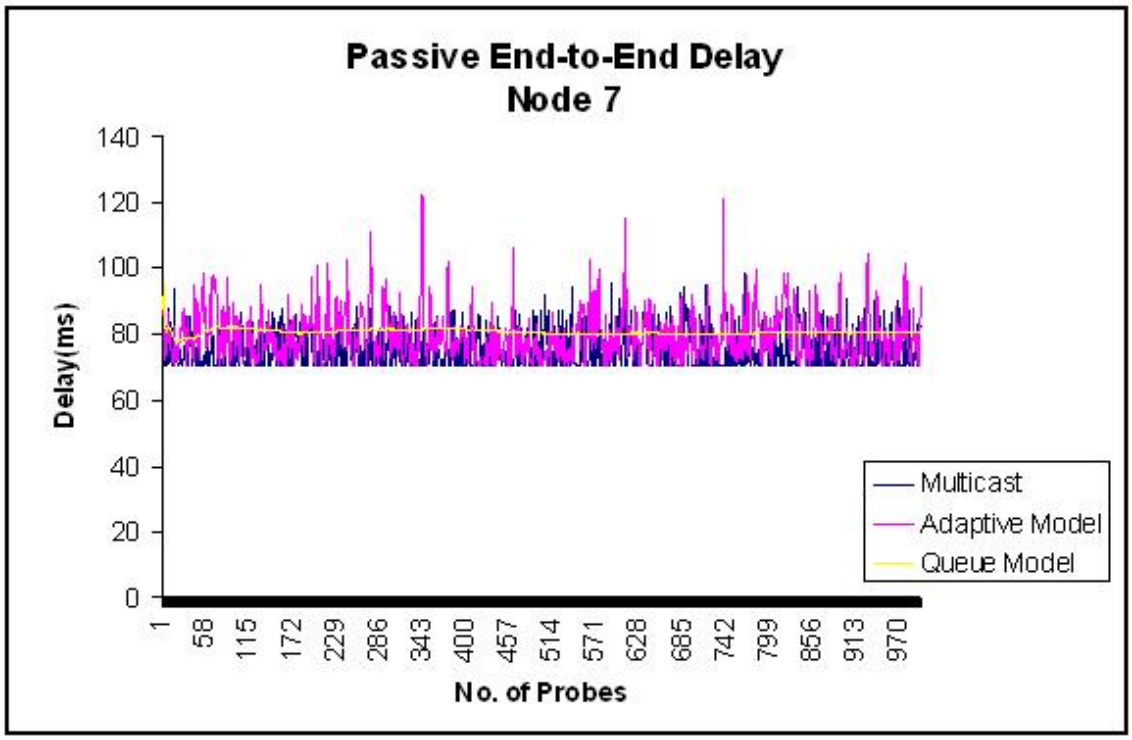

Figure 6.14: *End-to-End Delay Node 7 Passive*

Appendix D

6.1 ns-2 Scripts

```
set ns [new Simulator]
$ns multicast
set f [open mout.tr w]
$ns trace-all $f
$ns namtrace-all [open mout.nam w]
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]
set n5 [$ns node]
set n6 [$ns node]
set n7 [$ns node]
# Use automatic layout $ns duplex-link $n0 $n1 1.0Mb 10ms DropTail
$ns duplex-link $n1 $n2 5.0Mb 50ms DropTail
$ns duplex-link $n1 $n3 5.0Mb 50ms DropTail
$ns duplex-link $n2 $n4 1.0Mb 10ms DropTail
$ns duplex-link $n2 $n5 1.0Mb 10ms DropTail
$ns duplex-link $n3 $n6 1.0Mb 10ms DropTail
$ns duplex-link $n3 $n7 1.0Mb 10ms DropTail
#queue size
$ns queue-limit $n0 $n1 10
$ns queue-limit $n1 $n2 10
$ns queue-limit $n1 $n3 10
$ns queue-limit $n2 $n4 10
$ns queue-limit $n2 $n5 10
$ns queue-limit $n3 $n6 10
$ns queue-limit $n3 $n7 10
$ns duplex-link-op $n0 $n1 orient down
```

```
$ns duplex-link-op $n1 $n2 orient left-down
$ns duplex-link-op $n1 $n3 orient right-down
$ns duplex-link-op $n2 $n4 orient left-down
$ns duplex-link-op $n0 $n1 queuePos 0.5 $ns duplex-link-op $n2 $n5 orient right-down
$ns duplex-link-op $n3 $n6 orient down
$ns duplex-link-op $n3 $n7 orient right-down
set group [Node allocaddr]
set mrthandle [$ns mrtproto DM ]


    #source
set udpg [new Agent/mUDP]
#poisson multicast
set e [new Application/Traffic/Exponential]
$e set packetSize_ 40b
$e set burst_time_ 0
$e set idle_time_ 18ms
$e set rate_ 20Kb
$e attach-agent $udpg
$ns attach-agent $n0 $udpg
$ns color 40 white
$udpg set fid_ 40
$udpg set_filename udp0.out
$udpg set dst_addr_ $group
$udpg set dst_port_ 0
#background traffic agents
set udp04 [new Agent/UDP]
set udp05 [new Agent/UDP]
set udp65 [new Agent/UDP]
set tcp04 [new Agent/TCP]
$ns attach-agent $n0 $tcp04
set sink04 [new Agent/TCPSink]
$ns attach-agent $n4 $sink04
$ns connect $tcp04 $sink04
$ns color 41 violet
$tcp04 set fid_ 41
#ftp application 04
set ftp04 [new Application/FTP]
$ftp04 attach-agent $tcp04
set rng041 [new RNG]
$rng041 seed 0
set rng042 [new RNG]
```

```
$rng042 seed 0
set RV04 [new RandomVariable/Exponential]
$RV04 set avg_ 0.045
$RV04 use-rng $rng041
#random size
set RVSize04 [new RandomVariable/Pareto]
$RVSize04 set avg_ 1000
$RVSize04 set shape_ 1.5
$RVSize04 use-rng $rng042
#number of flows
set nFlows1 1000 #keep the random tines in array t
set t1 [$ns now]
for set j 1 $j¡=$nFlows1 incr j
set t1 [expr $t1 + [$RV04 value]
]
#put in array
set times(j) $t1
#random size in array
set s [expr [$RVSize04 value]
]
set Size(j) $s
$ns at $times(j) "$ftp04 send $Size(j)"
#0 to 7
$ns color 7 purple
set udp07 [new Agent/UDP]
set exp07 [new Application/Traffic/Exponential]
$exp07 set packetSize_ 1000 $exp07 set burst_time_ 0
$exp07 set idle_time_ 20ms
$exp07 set rate_ 0.5Mb
$exp07 attach-agent $udp07
$ns attach-agent $n0 $udp07
$udp07 set fid_ 7
set null07 [new Agent/Null]
$ns attach-agent $n7 $null07
$ns connect $udp07 $null07
#ftp application 56
set tcp56 [new Agent/TCP]
$ns attach-agent $n5 $tcp56
set sink56 [new Agent/TCPSink]
$ns attach-agent $n6 $sink56
$ns connect $tcp56 $sink56
$ns color 56 gray
$tcp56 set fid_ 56
```

```
set ftp56 [new Application/FTP]
$ftp56 set packetSize_ 1000
$ftp56 attach-agent $tcp56
set rng1 [new RNG]
$rng1 seed 0
set rng2 [new RNG]
$rng2 seed 0
set RV [new RandomVariable/Exponential]
$RV set avg_ 0.045
$RV use-rng $rng1
#random size
set RVSize [new RandomVariable/Pareto]
$RVSize set avg_ 1000
$RVSize set shape_ 1.5
$RVSize use-rng $rng2
#number of flows
set nFlows 1000 #keep the random tines in array t
set t [$ns now]
for set j 1 $j¡=$nFlows incr j
set t [expr $t + [$RV value]
#put in array
set times(j) $t
#random size in array
set s [expr [$RVSize value]
set Size(j) $s
$ns at $times(j) "$ftp56 send $Size(j)"

#source and receivers
$ns color 400 red
set src04 [new AgentmUDP]
#poisson
set app04 [new Application/Traffic/Exponential]
$app04 set packetSize_ 40b
$app04 set burst_time_ 0
$app04 set idle_time_ 8ms
$app04 set rate_ 20Kb
$app04 set codec_selector_ 1
$app04 attach-agent $src04
$ns attach-agent $n0 $src04
$src04 set fid_ 400
$src04 set_filename src04.out
#receiver
set rcvr4 [new Agent/mUdpSink]
```

```
$rcvr4 set_filename rcvr04.out
$ns attach-agent $n4 $rcvr4
$ns connect $src04 $rcvr4
#source
$ns color 50 yellow
set src05 [new Agent/mUDP]
#poisson application
set app05 [new Application/Traffic/Exponential]
$app05 set packetSize_ 40b
$app05 set burst_time_ 0
$app05 set idle_time_ 72ms
$app05 set rate_ 20Kb
$app05 set codec_selector_ 1
$app05 attach-agent $src05
$ns attach-agent $n0 $src05
$src05 set fid_ 50
$src05 set_filename src05.out
#receiver
set rcvr5 [new Agent/mUdpSink]
$rcvr5 set_filename rcvr05.out
$ns attach-agent $n5 $rcvr5
$ns connect $src05 $rcvr5
#source
$ns color 60 green
set src06 [new Agent/mUDP]
#poisson application
set app06 [new Application/Traffic/Exponential]
$app06 set packetSize_ 40b
$app06 set burst_time_ 0
$app06 set idle_time_ 8ms
$app06 set rate_ 30Kb
$app06 set random_ true
$app06 attach-agent $src06
$ns attach-agent $n0 $src06
$src06 set fid_ 60
$src06 set_filename src06.out
#receiver
set rcvr6 [new Agent/mUdpSink]
$rcvr6 set_filename rcvr06.out
$ns attach-agent $n6 $rcvr6
$ns connect $src06 $rcvr6
#source
$ns color 70 blue
```

```
set src07 [new Agent/mUDP]
#poisson application
set app07 [new Application/Traffic/Exponential]
$app07 set packetSize_ 40b
$app07 set burst_time_ 0
$app07 set idle_time_ 72ms
$app07 set rate_ 30Kb
$app07 set random_ true
$app07 attach-agent $src07
$ns attach-agent $n0 $src07
$src07 set fid_ 70
$src07 set_filename src07.out
#receiver
set rcvr7 [new Agent/mUdpSink]
$rcvr7 set_filename rcvr07.out
$ns attach-agent $n7 $rcvr7
$ns connect $src07 $rcvr7
#source/receivers
#multicast receiver
set rcvrm1 [new Agent/mUdpSink]
$rcvrm1 set_filename tomo1.out
$ns attach-agent $n1 $rcvrm1
set rcvrm2 [new Agent/mUdpSink]
$rcvrm2 set_filename tomo2.out
$ns attach-agent $n2 $rcvrm2
set rcvrm3 [new Agent/mUdpSink]
$rcvrm3 set_filename tomo3.out
$ns attach-agent $n3 $rcvrm3
set rcvrm4 [new Agent/mUdpSink]
$rcvrm4 set_filename tomo4.out
$ns attach-agent $n4 $rcvrm4
set rcvrm5 [new Agent/mUdpSink]
$rcvrm5 set_filename tomo5.out
$ns attach-agent $n5 $rcvrm5
set rcvrm6 [new Agent/mUdpSink]
$rcvrm6 set_filename tomo6.out
$ns attach-agent $n6 $rcvrm6
set rcvrm7 [new Agent/mUdpSink]
$rcvrm7 set_filename tomo7.out
$ns attach-agent $n7 $rcvrm7
#multicast traffic start
$ns at 0.10
"$n1 join-group $rcvrm1 $group"
```

```
$ns at 0.11 ”$n2 join-group $rcvrm2 $group”
$ns at 0.12 ”$n3 join-group $rcvrm3 $group”
$ns at 0.13 ”$n4 join-group $rcvrm4 $group”
$ns at 0.14 ”$n5 join-group $rcvrm5 $group”
$ns at 0.15 ”$n6 join-group $rcvrm6 $group”
$ns at 0.16 ”$n7 join-group $rcvrm7 $group”
#multicast source
$ns at 0.20 ”$e start”
#start background traffic
$ns at 0.19 ”$exp07 start”
#Data traffic start
$ns at 0.20 ”$app04 start”
$ns at 0.20 ”$app05 start”
$ns at 0.20 ”$app06 start”
$ns at 0.20 ”$app07 start”
$ns at 150.0 ”finish”
proc finish
global ns
$ns flush-trace
puts ”running nam...”
exec nam mout.nam &
exit 0

$ns run
```

The unicast code is similar to the above code except that the time difference between the pairs app04 app05 and app06, app07 is 0.1ms instead of 64ms.

Appendix E

fd = fopen('delay.dat','r'); #data source
$[xt, cnt] = fscanf(fd, "\%f", [4, 1000])$; read data in array
fclose(fd)
i=1;
format bank; #d.p
ffid = fopen('delay', 'a+');
for i=1:1000,
b=xt'(i, :);
Y=b';
$d = A \setminus Y$;
e=d-m;
X=abs(e);

fdisp(ffid, X');
endfor;
fclose(ffid);
Y is the the end-to-end delay
m=[10,50,50,10,10,10,10] it is the propagation delay

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

To use script you need the end-to-end delay. Keep them in the delay.dat file and run the script.

Bibliography

[1] C. Tschudin C. Jelger G.Bouabene G. Leduc L. Peluso M. Sifalakis M. Schoeller. Ana blueprint first version, Feb 2007.

[2] L. Parziale David T. Britt C. Davis J. Forrester. *TCP/IP Tutorial and Technical Overview* http://www.redbooks.ibm.com/redbooks/pdfs/gg243376.pdf. IBM, 8 edition, December 2006.

[3] J. Crowcroft S. Hand R. Mortier Warfield. An argument for network pluralism. *ACM*, 1581137486/03/0008S, August 2003.

[4] David D. Clark, John Wroclawski, Karen R. Sollins, and Robert Braden. Tussle in cyberspace: defining tomorrow's internet. *ACM SIGCOMM Computer Communication Review*, 32(4):347–356, October 2002.

[5] Stefan Savage, Neal Cardwell, David Wetherall, and Tom Anderson. Tcp congestion control with a misbehaving receiver. *SIGCOMM Comput. Commun. Rev.*, 29(5):71–78, 1999.

[6] Raghupathy Sivakumar Karthikeyan Sundaresan. Atp:a reliable transport protocol for ad-hoc networks. *IEEE TRANSACTIONS ON MOBILE COMPUTING*, 4(6), Nov/Dec 2005.

[7] Glenn Fink and Deb Frincke. Autonomic computing, freedom or threat, http://www.usenix.org/publications/login/2007-04/openpdfs/fink.pdf, April 2007.

[8] Ofir Arkin. Demystifying passive network discovery and monitoring systems *http://www.usenix.org/publications/login/2005-06/pdfs/arkin0506.pdf*, June 2005.

[9] R. Castro M. Coates G. Liang R. Nowak B. Yu. Network tomography recent developments. *Statistical Science 2004*, 19(3):499–517, 2004.

[10] Y. Tsang, M. Coates, and R. Nowak. Passive network tomography using em algorithms,proceedings of ieee international conference on acoustics, speech and signal processing. In *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing*, Salt Lake City, UT, May 2001.

[11] Colin English, Sotirios Terzis, and Paddy Nixon. Towards self-protecting ubiquitous systems: monitoring trust-based interactions. *Personal Ubiquitous Comput.*, 10(1):50–54, 2005.

[12] Mark Burgess. Promise theory, *http://research.iu.hio.no/promises.php*.

[13] Francesco Lo Presti, N. G. Duffield, Joe Horowitz, and IEEE/ACM Transactions on Network Don Towsley. Multicast-based inference of network-internal delay distributions,ieee/acm transactions on network. *IEEE/ACM Transactions on Network*, 10(6):761–775, 2002.

[14] G. Liang and B. Yu. Maximum pseudo likelihood estimation in network tomography. *IEEE Transaction on Signal Processing*, 51(8):2043–2053, August 2003.

[15] Christian Tschudin Christophe Jelger and Stefan Schmid. Basic abstractions for an autonomic network architecture, http://cn.cs.unibas.ch/pub/doc/2007-aoc.pdf.

[16] M. Sifalakis S. Schmid and D. Hutchison. *Towards Autonomic Networks*, volume 4195 of *LNCS*, chapter 1, pages 1–11. Springer Berlin/Heidelberg, Sept. 2006.

[17] Roy Sterritt. Towards autonomic computing: Effective event management. In *Proceedings of the 27th Annual NASA Goddard Software Engineering Workshop (SEW-27'02)*, page 40, Washington, DC, USA, 2002. IEEE Computer Society.

[18] IBM. Autonomic computing *http://www.ibm.com/autonomic/pdfs/Autonomic_Computing_Overview.pdf*.

[19] M. Burgess. Computer immunology. *Proceedings of the Twelth Systems Administration Conference (LISA XII) (USENIX Association: Berkeley, CA)*, page 283, 1998.

[20] M. Coates and R. Nowak. Network loss inference using unicast end-to-end measurement, proc. of itc seminar on ip traffic, measurement and modelling, Sept. 2000.

[21] N. Bieberstein S. Bose M. Fiammante K. Jones R. Shah. *Service Oriented Architecture (SOA) Compass*. IBM Press, 3 edition, April 2006.

[22] Mark Burgess and Siri Fagernes. Promise theory - a model of autonomous objects for pervasive computing and swarms. In *ICNS '06: Proceedings of the International conference on Networking and Services*, page 118, Washington, DC, USA, 2006. IEEE Computer Society.

[23] Y. Yardi. http://en.wikipedia.org/wiki/network_tomography, 1996.

[24] Earl Lawrence, George Michailidis, and Vijay N. Nair. Local area network analysis using end-to-end delay tomography. *SIGMETRICS Perform. Eval. Rev.*, 33(3):39–45, 2005.

[25] J Horowitz F. Lo Presti, N.G Duffied. Multicast-based inference of network-internal delay distribution, umass cmpsci technical report 99-55.

[26] Jae Myung. Tutorial on maximum likelihood estimation, journal of mathematical psychology 47 (2003) 90100, October 2002.

[27] M. Coates and R. Nowak. Network tomography for internal delay estimation,in proceedings of ieee international conference on acoustics, speech and signal processing, salt lake city, ut, 2001.

[28] Mark Burgess. Two dimesional time series for anomaly detection and regulation in adaptive system, http://research.iu.hio.no/papers/dsom2002.pdf.

[29] S. Vander Wiel J. Cao, D. Davis and B. Yu. Time-varying network tomography: router link data.,journal of american statistics association, 95(452):1063 1075, 2000., 2000.

[30] Alfred O. Hero III M. Coates. Internet tomography, ieee signal processing, magazine., 05 2002.

[31] Earl Lawrence. Network delay tomography using flexicast,j. r. statist. soc. b 68, part 5, pp. 785813, 2006.

[32] Mark Burgess. Analytical network and system administration, john wiley and sons, ltd,2004, 0-470-86100-2.

[33] Gert Botha Erasmus. *Stochastic Models of Steady State and Dynamic Operation of Systems of Congestion,"http://upetd.up.ac.za/thesis/available/etd-10182006-122618/unrestricted/02chapter2.pdf"*. PhD thesis, University of Pretoria, June 2005.

[34] Ucb/lbnl/vint network simulator ns (version 2) available:http://www.isi.edu/nsnam/ns/.

[35] http://140.116.72.80/ smallko/ns2/tool_en.htm. Ns-2 delay measurement script.