Congestion Inference and Traffic Engineering in Networks

Vijay Arya

Université de Nice - Sophia Antipolis – UFR Sciences
École Doctorale STIC

THÈSE

Présentée pour obtenir le titre de :

Docteur en Sciences de l'Université de Nice - Sophia Antipolis

Spécialité : INFORMATIQUE

par

Vijay ARYA

Équipe d'accueil : Planète – INRIA Sophia Antipolis

CONGESTION INFERENCE AND TRAFFIC ENGINEERING IN NETWORKS

Thèse dirigée par Thierry TURLETTI

Soutenue publiquement le 5 Juillet 2005, à 10h30 devant le jury composé de :

Président :	Pierre	BERNHARD	UNSA/ESSI, France
Directeur :	Thierry	TURLETTI	INRIA-Sophia, France
Rapporteurs :	Guy	LEDUC	Université de Liège, Belgium
	Torsten	BRAUN	Universität Bern, Switzerland
	Bernard	COUSIN	Université de L' IFSIC - IRISA, France
Examinateurs :	Timur	FRIEDMAN	U. P. & M. Curie, LiP6 - CNRS, France
	Philippe	OWEZARSKI	LAAS - CNRS, France

Thèse

Inférence de congestion
et
Ingénierie de Trafic dans les Réseaux

Congestion Inference
and
Traffic Engineering in Networks

Vijay Arya

July 2005

CONGESTION INFERENCE AND TRAFFIC ENGINEERING IN NETWORKS

by
Vijay Arya
Directeur de thèse: Thierry Turletti
Planète Projet, Inria Sophia Antipolis, France

ABSTRACT

This thesis focusses on problems related to (i) Inference of Congestion in the Internet and (ii) Multicast Traffic Engineering in Overlay Networks.

In the first part of the thesis, we propose methods which help to improve the quality of the congestion inference on both end-to-end paths and internal network links in the Internet.

Today, transport protocols which transfer data in the Internet classify all packet losses as congestion. However, the development of various wireless access networks had led to the growth of wireless links in the Internet. Therefore, the packet losses observed by a transport protocol may have resulted either due to congestion or due to the presence of wireless links on the end-to-end path. In order to make best use of available network bandwidth, a transport protocol must reduce its sending rate only in response to congestion losses and not in response to wireless losses. In the second chapter of this dissertation, we consider the problem of differentiating congestion and wireless losses for unreliable transport protocols which transfer multimedia flows. Previously proposed end-to-end loss differentiation schemes significantly misclassify wireless and congestion losses. We propose an explicit loss differentiation scheme which uses agents at the boundaries of wireless links and results in an accurate inference of congestion on the end-to-end path. We show how the agents can intelligently record the cause of packet losses within packets which are not lost, with low overhead.

Inferring the level of congestion on internal network links or paths is useful for the purposes of monitoring and management of networks. The level of congestion on a link can be inferred by measuring the loss rate on that link. Installing or upgrading network elements to monitor internal links is an expensive process and hence the characteristics of internal links are often inferred from end-to-end measurements.

The process of inferring characteristics of internal links from end-to-end measurements has come to be known as Network Tomography. One of the earliest proposed methods of performing network tomography is MINC (Multicast-based Inference of Network Characteristics), which infers loss rates (congestion) on internal network links using end-to-end multicast measurements. In MINC, per-link loss rates are inferred by analyzing binary feedbacks which are reported by multicast receivers in response to measurement probes sent from the source. However, due to the presence of buggy or malicious multicast receivers, feedbacks collected may be incorrect leading to a faulty inference of link loss rates. In the third chapter of this dissertation, we present a statistical verification procedure which can identify if the binary feedback data collected from receivers of a multicast tree contains incorrect feedbacks. The procedure does not require the knowledge of multicast tree topology and is able to identify incorrect data even in the presence of colluding receivers.

In order to infer link characteristics from end-to-end multicast measurements, dedicated infrastructures to perform these measurements need to be deployed. For large scale multicast measurements, the task of deployment is complex and expensive. To avoid this, it was proposed to couple the process of reporting binary feedbacks for MINC loss inference with RTCP (Real-Time Control Protocol). In this passive measurement architecture, existing multicast sessions which use RTP (Real-Time Transport Protocol) to transfer their data can use their data packets as probes and report binary feedbacks needed for loss inference by piggy-backing them on RTCP packets. For loss inference, MINC requires receivers to report per probe binary feedbacks and this poses constraints since RTCP feedback bandwidth must not exceed 5%

of data bandwidth. In the fourth chapter of this dissertation, we develop an extended MINC loss estimator which can perform loss inference using aggregate receiver feedbacks. Aggregate feedbacks require less feedback bandwidth and the estimator is able to perform loss inference using aggregate feedbacks without significant loss of accuracy. We also compare this estimator to the approach where MINC loss inference is performed using a reduced set of binary feedbacks.

In the second part of the thesis, we propose a method which helps to perform multicast traffic engineering in overlay networks.

In overlay networks, the network nodes which are generally servers can perform intelligent and adaptive networking functions which normally routers do not support. By studying the state of overlay paths, nodes can route traffic dynamically and perform load balancing or routing based on certain constraints. One of the tasks which overlay networks support is multicast routing. To make best use of the traffic conditions in the network, overlay nodes can route multicast traffic adaptively and achieve goals of traffic engineering. To be able to do so, they need mechanisms to choose specific or explicit multicast trees in the overlay. One way of routing traffic on explicit trees is to perform source routing, that is, to specify the tree within multicast data packets. In the fifth chapter of this dissertation, we show efficient ways of encoding multicast trees within data packets. These encodings are almost optimal in terms of space and can be read and processed efficiently. They can be used to represent multicast trees within data packets to route multicast traffic on explicit trees in a stateless manner. We show the correspondence of multicast trees to theoretical tree data structures and obtain lower bounds on the number of bits needed to represent multicast trees.

ACKNOWLEDGMENTS

The ideas as to how a doctoral dissertation should look like were formed in my mind by constantly seeing the hard-bound thick volume placed in the topmost row of one of the bookshelves at home. That was my father's dissertation, composed around the time that I arrived into this world. As this dissertation comes to its completion, I sometimes wonder if I should receive my doctoral degree for this slim booklet.

I would like to thank several people who helped me arrive at this stage. Many thanks to my master's thesis supervisor Naveen Garg for having provided a heavy but wonderful introduction to research. I first tasted research in the "k-median" group meetings with Naveen, Rohit Khandekar, and Vinayaka Pandit.

Sincere thanks to my supervisor Thierry Turletti for having always been there to listen to and comment upon whatever I had thought out or wanted to discuss about and patiently reviewing all my work. Thierry's enthusiastic nature helped me perform better.

I am deeply indebted to all researchers of Planète team for their invaluable help and advice. Thanks to our team head Walid Dabbous for providing the elder-brotherly advice on various matters. Thanks to Chadi Barakat for numerous technical discussions and for reviewing pieces of my work. Chadi's command and ease in stochastic modeling are somethings I aspire to achieve one day. Thanks to Arnaud Legout for rigourously reviewing some of my papers which now form a part of this dissertation. Many thanks to both Thierry and Walid for having arranged the funding for this thesis, which comes from French and European projects.

Some of the work in this thesis was done at Rensselaer Polytechnic Institute (RPI), NY, USA. I would like to thank Prof. Shivkumar Kalyanaraman for having hosted my stay at RPI. Discussions with Shiv helped to strengthen my knowledge of networks and traffic engineering. I hope to carry on my collaborations with Shiv. My stay at RPI became really enjoyable because of Anand Muthukrishnan and Mayank Jain. Thanks to them and thanks also to the former for having sportively witnessed a car crash from the eyes of a passenger.

Some of the work in this thesis is due to collaboration with Timur Friedman at LIP6, Paris and Nick Duffield at AT&T Research, USA. Many thanks to both of them for being the best mentors a student can possible have. This thesis would not have been possible without their collaboration and I hope to continue collaborations with them.

I would like to thank all Jury members of my thesis for patiently reviewing the dissertation, giving helpful comments, and asking me some great questions during the defense. Thanks to Pierre Bernhard, Guy Leduc, Torsten Braun, Bernard Cousin, Timur Friedman, and Philippe Owezarski. Thanks to Pierre Bernhard, the president of the jury, for having orchestrated the defense in an enjoyable manner. Special thanks to Tosten Braun with whom I worked on some interesting problems during his stay in Planète. Discussions with Torsten also helped me in my own work and I hope to collaborate with Torsten in the future. Thanks also to our team secretary Aurelie Richard for having arranged the defense with care and for help on various matters in the past few years.

Thanks to the old and new generations of Planète doctoral students and engineers for lunch and café discussions. The older generation included Fatma Louati (whose peugeot I drive today), Miguel A. Ruiz Sanchez, Rares Serban, Laurentiu Barza, Ni Qiang (whose wife cooks delicious Chinese food), Hitoshi

Asaeda, and Kim Hahnsang. The new generation includes my office mate Hossein Manshaei whose help was invaluable on various issues including car driving, ”les tunisians” Abdel Basset Trad and Mohamed Kaafar, and ”le libanais” Mohammad Malli (beware, whose religious philosophy can be enlightening). Ceilidh Hoffman's arrival to our team last year was like monsoon rains on parched lands. Thanks to Ceili for being a person who spoke only english, for the delicious dinners, discussions, chats, and walks. This dissertation is typeset using the tex format I borrowed from her.

Outside team, several people made stay at Sophia and Antibes easy and fun. Many thanks to the members of 2VB team - Ana, Luis, Sylvain, Adeline, Kuntal, Olivier - for all the hikes, dinners, and drinks. Thanks to Parijat, Bala, Arzad, Dinesh, and Ahmed from Maestro for help on various matters, chats, and some pizza dinners. Special thanks to Florence from Maestro and Sylvain who are solely responsible for the french sections of this dissertation.

This dissertation is dedicated to my family. I would like to thank my parents, my sisters, and my brother for having been supportive and encouraging all throughout my studies. I am now glad they have eventually stopped asking me the question ”How is your PhD coming up?” when I call them.

Vijay Arya
vijay.arya@sophia.inria.fr
Sophia Antipolis, France

À ma famille

To

Dadaa, Mummy,

Neetu, Saroj, and Pappu

CONTENTS

FIGURES

TABLES

1

INTRODUCTION

This thesis focusses on problems related to (i) Inference of Congestion in the Internet and (ii) Multicast Traffic Engineering in Overlay Networks. In the first part of the thesis, we propose methods which help to improve the quality of congestion inference on both end-to-end paths and individual links in the Internet. In the second part of the thesis, we propose a method which helps to perform multicast traffic engineering in overlay networks.

1.1 Congestion in the Internet

Internet is a store and forward packet-switched network which offers a single class best-effort service model. In this network, packets introduced by various hosts traverse communication links, briefly wait in router queues and reach their respective destinations. The routers which forward packets do not provide any priority to packets and generally forward them in the order that they are received. One of the key aspects of such best-effort networks is *Congestion*. In general, congestion occurs whenever the demand for a network resource is greater than its capacity [1]. If for instance during an interval of time the number of packets wanting to traverse a network link exceeds its capacity, congestion occurs on that link. As a consequence of congestion, packet queues at router buffers start to grow and eventually packets which arrive at full buffers are dropped by routers. Other than dropping packets which they cannot accommodate, routers do nothing to inform anyone that there is congestion[1]. When congestion in the network increases beyond a threshold, performance of the network degrades. In order to cooperatively use the network to its capacity to transfer data, it is the duty of hosts or rather

[1]We note here that ECN (Explicit Congestion Notification) [2] which is an exception to this, is not widely deployed

the transport protocols which run on hosts to infer that there is congestion and to keep it well under control.

1.1.1 Inference of Congestion

The simplest measurable symptom of congestion is a packet loss. If packets traversing a network path are being dropped frequently, it is likely that the network path is facing congestion. The other symptom of congestion on a network path is the increase of its latency (delay). If packets traversing a network path are facing increased delay, it is likely that they are spending more time in long router queues which have developed as a consequence of congestion. However, measuring delay and inferring congestion from it is a more complex process. In this thesis, we shall be concerned with the inference of congestion in terms of packet loss rate.

Transport protocols such as TCP which transfer data in the Internet infer the level of congestion on the end-to-end path and based on this inference, increase or decrease the rate at which they send packets into the network. If these protocols do not infer the level of congestion correctly, they may either harm the flow that they carry or other flows in the network. A protocol which falsely infers more congestion does not utilize the available bandwidth and a protocol which falsely infers less congestion may steal bandwidth from other network flows. Therefore, correct inference of congestion on the end-to-end path is crucial to the functioning of transport protocols in the Internet.

On the other hand, knowledge of the level of congestion on specific network links or paths is needed for the purposes of monitoring and management of networks. This information is useful to network operators who manage individual internetworks within the Internet. Knowing the level of congestion on specific paths of their network allows them to find out how well their network is performing. They can use this information to make decisions concerning the routing of traffic within their network and the upgrade of certain parts of their network. Similarly, service providers who offer services to clients located in different networks are interested in knowing whether the paths they utilize remain congested or not.

In the first part of the thesis, we propose solutions which help to improve the quality of congestion inference on both end-to-end paths and individual links in the Internet.

1.1.2 Problems and Contributions

Differentiation of Wireless and Congestion losses

Starting with Jacobson's implementation of TCP [3], today most transport protocols in the Internet mimic TCP behavior and interpret every packet loss as a symptom of congestion on the end-to-end path. With the growth of wireless technologies, Internet now has different types of wireless access networks such as wireless LANs and 3G Mobile networks as well as some

internal wireless links. Wireless links are error prone and normally the wireless data-link layer only provides partial reliability. In such a framework, end-to-end transport protocols observe packet losses either when there is congestion or when the end-to-end path crosses a wireless link. To use the available network bandwidth, a transport protocol must reduce its sending rate only in response to congestion losses and not in response to wireless losses. This motivates the problem of loss differentiation wherein the goal is to differentiate between congestion and wireless losses, so that transport protocols can make an accurate inference of congestion on end-to-end paths. The problem was first discussed with respect to TCP by authors of [4]. They proposed the installation of a snoop agent at base stations which prevents a TCP sender from falsely reducing its sending rate in response to wireless losses. In this thesis, we consider the problem of loss differentiation with respect to unreliable transport protocols which carry multimedia flows.

There are two approaches to loss differentiation - *end-to-end* and *explicit*. In end-to-end loss differentiation, the receiver at the transport level guesses the cause of a packet loss without any help from internal network elements. End-to-end loss differentiation schemes work by utilizing facts such as inter-arrival time of packets. However, all these schemes fail to perform correctly, and significantly misclassify wireless and congestion losses. We propose an Accurate and Explicit loss Differentiation scheme (AED) which uses agents at the boundaries of wireless links [5]. The agents intelligently record the cause of packet losses within packets which are not lost, with low overhead. AED allows transport protocols to accurately differentiate between wireless and congestion losses.

Trustworthy Loss Inference

The information about the level of congestion on specific links or paths of a network can be obtained using two approaches: (i) By installing special monitoring software at network elements (ii) From end-to-end measurements. When networks which need to be monitored span different administrative domains, it is virtually impossible to install special equipment inside network elements to monitor individual links or paths. Even in one administrative domain, this requires the upgrade of several network elements, a task which is expensive. For this reason, the characteristics of internal links or paths in a network are often inferred by performing end-to-end measurements.

The field of networking in which characteristics of internal network links or paths are inferred from end-to-end measurements has come to be known as Network Tomography (due to its similarity with medical tomography) [6]. One of the earliest proposed methods of performing network tomography is MINC (Multicast-based Inference of Network internal Characteristics [7]). Based on end-to-end multicast measurements, MINC can infer loss rates (i.e., level of congestion) on specific network links. To perform loss inference, the source injects probe

packets into the multicast tree and each receiver reports whether it receives the probe packet or not. The source and receivers are simple end-hosts in the Internet. Using the binary feedback traces collected from all receivers, MINC infers the loss rates of links in the multicast tree. However, due to the presence of buggy or malicious multicast receivers, the feedbacks collected from receivers may be incorrect and this can result in a faulty inference of link loss rates. A faulty inference may point high loss rates on good links and low loss rates on lossy links. In this thesis, we consider the problem of verifying binary multicast measurements in order to ensure sound and trustworthy inference of link loss rates.

We propose a statistical verification procedure called *ICheck* which can verify the integrity of binary feedback data collected from multicast receivers [8]. The procedure uses ideas from MINC loss estimation itself and exploits the inherent correlation that exists in the feedback traces collected from different receivers of a multicast tree. Broadly, it utilizes the principle that feedbacks of different subsets of receivers can be used to infer the loss rate of the same link in the multicast tree. Therefore, by comparing different estimates of link loss rates, inconsistencies in feedbacks can be detected. The procedure has two key properties. It does not need the knowledge of the multicast tree topology and is able to detect inconsistencies even in the presence of colluding receivers.

Low Feedback Loss Inference

In order to infer link loss rates through active end-to-end measurements, dedicated infrastructures to perform these measurements must be deployed. For large scale multicast measurements, this task is complex. It involves getting access to hosts located in different parts of Internet and installing measurement software in in them. Generally hosts run on different operating systems and this increases the difficulties of installation and configuration. To facilitate the task of performing end-to-end multicast measurements, authors of [9] proposed a passive impromptu architecture which couples the process of reporting measurements with RTCP (Real-Time Control Protocol) [10]. In this architecture, multicast sessions which use RTP (Real-Time Transport Protocol) [10] to transfer data can utilize their data packets as probes and piggy-back the feedbacks needed for MINC loss inference on RTCP packets. However, one of the constraints here is that receivers are unable to report per packet binary feedbacks needed for loss inference since doing so can cause the RTCP feedback bandwidth to exceed 5% of data bandwidth. In this thesis, we consider the problem of performing MINC loss inference with less feedback data.

In the architecture proposed by authors of [9], receiver feedbacks are thinned to maintain the RTCP feedback bandwidth low, i.e., receivers report feedbacks corresponding to a reduced (sampled) set of probe packets and these feedbacks are used for loss inference.

We design an Extended MINC Loss Estimator (EMLE) which can perform loss inference

using aggregate feedbacks [11]. Each aggregate feedback is reported for a group of w probe packets and can be represented using less than w bits. Thus EMLE allows the estimation of link loss rates using less feedback bits. We compare EMLE to the case where MINC loss estimator is used along with a thinned set of feedbacks.

1.2 Multicast Traffic Engineering

The second part of this thesis deals with multicast traffic engineering in overlay networks. The goal of traffic engineering is to ”enhance the performance of a network, at both traffic and resource levels” [12]. In multicast traffic engineering, the main goal is to steer multicast traffic on selected trees in the network in a manner such that certain performance parameters associated with multicast traffic are met and network resources are utilized optimally.

Overlay networks are self-organizing virtual networks in which participating hosts form a network at the application level. Examples of overlay networks include RON (Resilient Overlay Networks) [13], OMNI (Overlay Multicast Network Infrastructure) [14], Akamai, and several peer to peer networks. In these networks, nodes which are generally servers can perform intelligent and adaptive networking functions which routers in the Internet normally do not support. Although services provided by implementing networking functions at the application level may be less efficient, the salient feature of overlay networks is that they are deployable.

1.2.1 Problem and Contribution

By studying the state of overlay paths, overlay nodes can route traffic dynamically to balance the load on paths or route traffic according to certain constraints to achieve goals of traffic engineering. One of the services which can be supported by overlay networks is multicast routing. In order to make the best use of available traffic conditions in the overlay, nodes can construct trees to route multicast traffic intelligently. For this, they need mechanisms to choose specific or explicit trees in the overlay. Maintenance of several different trees in the overlay by introducing per tree state at nodes is not a scalable option. Moreover, overlay nodes may be constrained in terms of memory. The problem of stateless and explicit multicast routing in overlays remains largely unaddressed in the literature.

One approach to performing stateless and explicit multicast routing is multicast source routing, i.e., to encode the multicast tree within multicast data packets. In this thesis, we consider this approach and design efficient ways of encoding multicast trees within data packets [15]. We show the correspondence of multicast trees to theoretical tree data structures and based on this correspondence show simple lower bounds on the number of bits needed to represent multicast trees. Our encodings are almost optimal in terms of space and can be read and processed

efficiently. They can be used to represent multicast trees within data packets to route multicast traffic on explicit trees in a stateless manner.

1.3 Organization of dissertation

The balance of this dissertation is organized as follows. Our contributions related to congestion inference appear in chapters 2, 3, and 4 of the dissertation. Our contribution related to multicast traffic engineering in overlay networks appears in chapter 5 of the dissertation.

In chapter 2, we discuss the problem of loss differentiation. We describe AED, an explicit loss differentiation scheme which helps transport protocols to accurately differentiate between wireless and congestion losses.

In chapter 3, we discuss the problem of verifying the integrity of binary feedbacks which are collected from multicast receivers. We show how link loss rates inferred by MINC become erroneous if multicast receivers report incorrect binary feedbacks. We develop a statistical verification algorithm called *ICheck* which can verify the integrity of binary feedbacks collected from multicast receivers.

In chapter 4, we discuss the problem of inferring loss rates of links using less feedback data. We develop the Extended MINC Loss Estimator (EMLE), which can infer link loss rates using aggregate feedbacks. We compare EMLE to the case where MINC loss estimator is used along with a thinned set of feedbacks.

In chapter 5, we present efficient methods of encoding multicast trees within data packets with the goal of perform stateless and explicit multicast routing in overlay networks.

The conclusions of the thesis appear in chapter 6 of the dissertation.

2

LOSS DIFFERENTIATION

2.1 Summary

In recent years mobile computing has experienced an explosive growth, mainly due to the integration of wireless networks with the wired Internet. To deploy bandwidth-greedy multimedia applications on such heterogeneous environments, efficient congestion control algorithms are required. In such environments, a packet can cross one or more wireless links before eventually reaching its destination. When congestion control protocols such as TFRC are used to transmit multimedia flows, they need to differentiate between congestion losses and wireless losses to behave correctly and efficiently. Currently proposed end-to-end loss differentiation mechanisms can not predict the differences between congestion and wireless losses reliably. In this work, we explore explicit loss differentiation. We have designed a simple window framework to explicitly and accurately differentiate wireless and congestion losses. By deploying agents at the boundaries of wireless links, we show that we can efficiently implement our window-based scheme with low overhead. We also point out the current limitations of our scheme.

2.2 Introduction

The Internet today is no longer completely wired, with Wireless LANs (WLANs), wireless backbones, and mobile networks getting appended to it. Most business offices, university campuses and hospitals now have WLANs. Both GSM(GPRS) and UMTS have standards for packet access from mobile terminals. Simultaneously, there is a growing popularity of real-time multimedia applications on the Internet such as audio/video streaming, IP telephony, video conferencing, network games, etc. The 3G wireless service providers are providing packet switched real-time multimedia services to their users and soon a 3G terminal will be able to access and play streaming audio or video available on a server in the Internet. To support such real-time multimedia applications on a network with wired and wireless links, efficient congestion control mechanisms are required for real-time flows.

TCP, the most dominant protocol in the Internet, is not suited for transferring real-time flows. TCP guarantees reliability and packet ordering at the expense of increase in end-to-end delay. Reliability however, is not a stringent requirement for real-time data. Instead, real-time data is timely and if it does not arrive after some deadline it may not even be useful. Besides, TCP uses additive increase multiplicative decrease algorithm which shows high variation in sending rate resulting in large jitters, degrading the user perceived quality. For these reasons, real-time flows are carried either directly using UDP or using protocols such as RTP (over UDP) [10]. RTP does not specify any explicit congestion control mechanism like TCP. In order to be fair to protocols that use congestion control such as TCP, and to avoid any possible congestion collapse, several TCP-friendly congestion control mechanisms were proposed for unreliable unicast flows [16, 17, 18, 19]. But all these mechanisms, like TCP, rely on implicit feedback and interpret any loss as a congestion loss. The implicit policy works accurately in wired networks where losses occur only due to congestion. However, with the presence of wireless links, a large percentage of packets are lost on these links. Wireless links are inherently error prone due to effects such as fading and multi-path and the wireless data link layer provides only partial reliability. Thus when TCP-friendly protocols are deployed on networks such as the Internet, they reduce their sending rate not only in response to congestion, but also in response to wireless losses. To avoid this erroneous behavior, several end-to-end loss differentiation schemes were proposed [20, 21, 22]. These schemes however, misclassify wireless and congestion losses. If congestion losses are misclassified as wireless, the scheme may not be TCP-friendly and may cause more congestion in the network. If wireless losses are misclassified as congestion, the scheme may not use the available bandwidth. Thus, there is a need to accurately differentiate between wireless and congestion losses.

In this chapter, we design a mechanism to accurately differentiate between wireless and congestion losses by deploying agents at the boundaries of wireless links. Our scheme is called

Accurate and Explicit Differentiation (AED) scheme. We consider a real-time flow which uses a protocol such as RTP [10] or DCCP [23] in conjunction with TFRC [24]. AED aims to inform the TFRC receiver about wireless and congestion losses so that it can send an accurate feedback to the sender. The sender would use the congestion loss information in the feedback to adjust its sending rate based on the exact level of congestion in the network. It can also use the wireless loss information in the feedback to determine the amount of bits to be allocated for source and channel coding, so that more losses can be recovered without retransmissions. Balance of the chapter is organized as follows. Section 2.3 discusses the past work, Section 2.4 explains our scheme, Sections 2.5 and 2.6 give the implementation details of our scheme. Sections 2.7 compares our scheme with end-to-end loss differentiation schemes and Section 2.8 points the limitations of our scheme. We conclude with Section 2.9.

2.3 Related work

The aim of loss differentiation schemes is to differentiate between wireless and congestion losses. These mechanisms are either *End-to-end* or *Explicit*. Explicit schemes are those that make use of agents deployed on intermediate network nodes. End-to-end schemes try to differentiate losses at the receiver without making use of any intermediate nodes.

Explicit Loss Differentiation was made use of in the context of TCP flows by the use of *Snoop Agents* [4]. Snoop agents are suited for first hop/last hop wireless topology encountered in mobile networks. Consider a TCP flow originating from a Fixed Host (FH) and terminating at a Mobile Host (MH). The snoop agent at the base station, by monitoring the TCP packets forwarded to MH and acknowledgments returned from the MH, maintains a cache of TCP packets that have been forwarded but not acknowledged by the MH. It detects a loss of packet on the wireless link by seeing duplicate acks or by a timeout based on locally maintained timers. When it does, it retransmits the lost packet to the MH if it has been cached, since these are packets lost on the wireless link. Additionally, it suppresses duplicate acks corresponding to the wireless losses, thus ensuring that the false congestion invocations at the sender are avoided. Note firstly that snoop agent decouples wireless from congestion losses without making any changes to the IP packets itself. Secondly, snoop works for reliable protocols since it also retransmits the lost packets cached.

Snoop agents are not suited for real-time flows since these flows are carried by unreliable protocols. For snoop agents to detect packet loss, the receiver needs to acknowledge every packet received. Although rate based protocols are more suited for real-time flows, snoop agents may be a possible candidate for congestion control mechanisms like Binomial congestion control [18] (where the receiver acknowledges every packet like TCP). But suppressing the duplicate acknowledgments for packets lost on the wireless link (to decouple wireless and con-

gestion losses) may increase the end-to-end delay which is harmful for real-time applications. For real-time applications, local retransmissions also may not help as retransmission delay may sometimes be unbearable. For real-time flows, agents which send information either to sender or receiver about wireless and congestion losses, by either marking packets or generating new packets, are required.

ECN (Explicit Congestion Notification) [2] is also a possible candidate for real-time flows. ECN is used by the routers to signal congestion in the network by marking IP packets. When TFRC is used in conjunction with ECN, the losses detected when marked IP packets are received are considered congestion losses and those detected when unmarked IP packets are received, are considered to be wireless losses and are not used for loss rate calculations. Using ECN can be inaccurate. For example, for a continuous bunch of wireless and congestion losses, ECN would classify all of them to be congestion losses. Although it may be the right time to invoke congestion control by reducing the sending rate, the amount of reduction will be more than necessary.

End-to-end schemes work at the transport level in the receiver. They make use of facts such as the time taken by the packet to travel through the network and packet inter-arrival time. Vidya and Biaz [20] proposed a scheme based on packet inter-arrival time suitable for last hop wireless bottleneck link. They approximate the minimum packet inter-arrival time seen during a connection (T_{min}) to the time taken by a packet to traverse the last hop wireless link. If a packet is lost and the subsequent packet arrives approximately after T_{min}, the loss is classified as wireless. This scheme has two main problems. Firstly, with more than one flow, packets from different flows get interspersed and the packet after a wireless loss may not come immediately after T_{min}. Secondly, in practice, the wireless channel is generally varying and the time taken by a packet to traverse the wireless link keeps changing, causing misclassification. Tobe et al, [21] use Relative One-way Trip Time (ROTT) as a measure of the time taken by a packet to travel from source to destination. Their scheme is based on the fact that when ROTT is plotted against time at the receiver, one can observe spikes during congestion. Thus losses during spikes are classified as congestion losses, and otherwise wireless. The problem with this is scheme is that congestion can occur on one or more routers together, resulting in spikes of varying sizes. The difficult part is to catch spikes of different sizes online. The Zig-zag scheme [22] also makes use of ROTT. In this scheme, the receiver keeps an estimate of running $ROTT_{mean}$ and $ROTT_{variance}$. Based on the number of losses and the difference of the current ROTT and $ROTT_{mean}$, they propose certain conditions for classification. All the end-to-end schemes were compared in [22]. End-to-end schemes generally have topological constraints and need to be tunned for specific topologies. Although some of these do improve TFRC throughput, they all suffer from significant misclassification. Thus congestion losses in the network may increase and TCP-friendliness of the flow may decrease. Also, the sender is

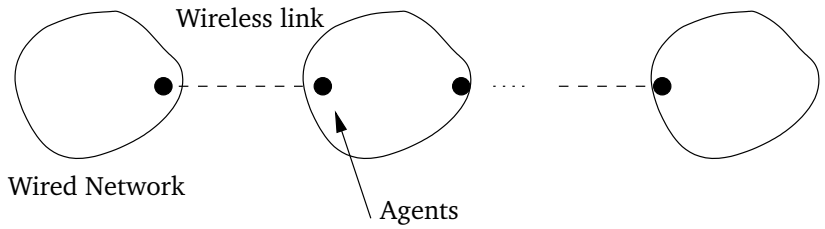


Figure 2.1: General topology of a hybrid network

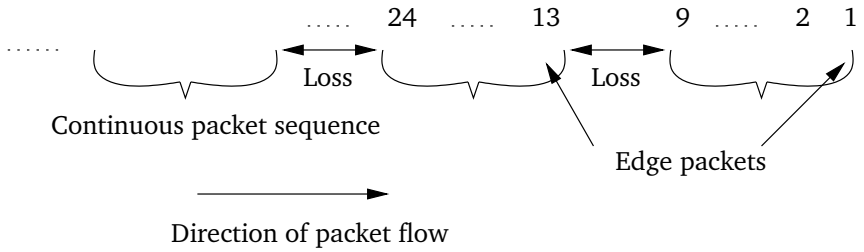


Figure 2.2: Packet sequence as seen by an agent

unable to use receiver's feedback for adjusting the bits allocated for source and channel coding since the feedback may be inaccurate.

We consider an explicit loss differentiation mechanism which is both accurate and is not limited by any topological constraints. We consider a general topology and show how by deploying agents at the boundaries of wireless links, one can accurately differentiate the losses. Using the AED scheme, TFRC is able to function like the optimal Omniscient-TFRC used for comparison in [22]. Omniscient-TFRC knows precisely which losses are congestion and which wireless and thus performs optimally on all topologies. In subsequent sections, we give the details of our scheme.

2.4 The AED Mechanism

We consider the following general topology of a hybrid network (figure 2.1) with a series of wired subnetworks interleaved by wireless links. Each subnetwork consists of at least one node. For example, a mobile node is considered as a wired network with one node.

Packets in the network are lost either due to congestion on the wired subnetworks or on the wireless links. To determine the right cause of a packet loss, we deploy agents before and after each wireless link. The agents snoop through each packet and detect a loss by seeing a packet with an out of order sequence number. For the packets flowing from left to right in Figure 2.1, agents at the end of wired networks treat a packet loss as congestion loss and agents after a wireless link treat a packet loss as wireless loss.

Figure 2.2 shows a general packet sequence with interleaved losses, as seen by an agent. The agent would want to mark the packets which are not lost, with information about the lost packets, that is, whether packets were lost due to congestion or a wireless link. However, the information must be recorded using a scheme which is most immune to future packet losses so that the information eventually reaches the receiver. For example, consider a scheme in which the agent marks the edge packets (packets just after a loss). Suppose that in Figure 2.2, packets 10..12 were lost on a wireless link and edge packet 13 holds this information. If packet 13 itself is lost by congestion in future, we would loose all the information as to how 10...12 were lost.

We consider a generalized scheme in which each packet holds information about a window of previous n packets. We call such a window as the *history* window. For a packet with sequence number j, its history window holds information about packets $j-1, ...j-n$, as to whether these packets were lost due to congestion, wireless link, or not lost at all. Figure 2.3 shows the concept of keeping a history window of size four. In this example, packet 5 is considered to be lost by congestion and 4 on a wireless link. The history windows maintained by packets 6 and 7 have been shown. 6's window records the fact that 5 is a congestion loss, 4 a wireless loss and that 3 and 2 are not lost. With this information, if packet 6 is lost in future due to congestion or a wireless link, we still will not loose information as to how packets 5 and 4 were lost since this information would also be available in packet 7. Only if all packets $6, 7, 8$ were lost by congestion, we would loose the information as to how 4 was lost.

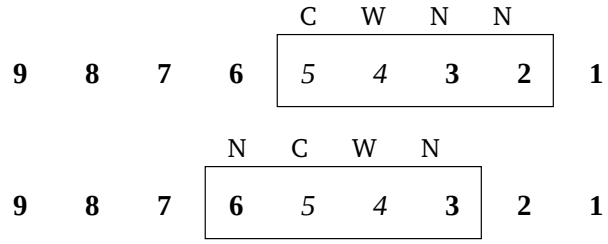


Figure 2.3: *History* window concept

Our next step is to determine how to record the contents of the history window efficiently in each packet. We propose a scheme which can represent a history window of size n, using just n bits. Each history window may consist of wireless losses, congestion losses and no losses. Now, suppose that we represent a wireless loss by 1 and congestion loss or no loss by 0, then we would face the following problem. Consider the scenario shown in Figure 2.4(a). In this example, 5 was lost on a wireless link somewhere before, 4 was lost due to congestion in the previous wired subnetwork. Hence 6's window is represented as 1000. Suppose that 3 gets lost on the wireless link. Now when the agent sees the packet sequence, it finds 5,4,3 missing. Since 5 is represented by a 1 in 6's window, it knows 5 was lost by a wireless loss. But it is unable to differentiate packets 4 and 3 in spite of the fact that they were lost because of different reasons.

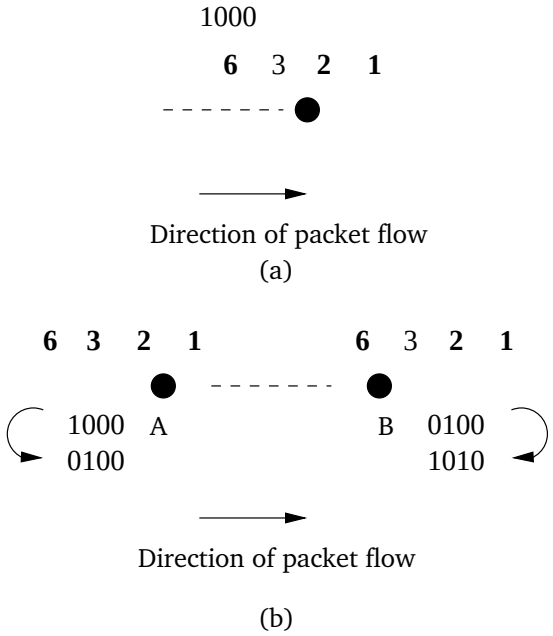


Figure 2.4: Flipping the bits before and after the wireless link

This problem can be tackled without increasing the number of bits to represent the history window, if each agent flips the bits appropriately. The point to observe is that the agent after a wireless link tries to detect *wireless* losses by seeing missing sequence numbers. So we must tell it explicitly if certain missing sequence numbers were due to congestion earlier. Hence this agent expects to see congestion losses represented by a 1. The reverse is true for the agent before the wireless links which aims to determine *congestion* losses by seeing missing sequence numbers. We must tell it explicitly if certain missing sequence numbers were due to a wireless link earlier.

Thus the agent before a wireless link interprets a wireless loss by 1 and congestion or no loss by 0. But before sending the packets on the wireless link, it flips the bits such that congestion loss is represented by 1 and wireless loss or no loss is represented by 0. The agent after the wireless link interprets congestion losses by 1 and wireless loss or no loss by a 0. But before sending the packets on the wired network, it flips the bits again such that wireless loss is represented by 1 and congestion or no loss is represented by 0. For the example in Figure 2.4(b), agent A doesn't see packets 5,4. It knows that 5 was lost by wireless loss since it is represented by 1 in 6's window. It concludes that 4 was lost by congestion, hence it flips the bits and represents 6's window by 0100. Now 3 gets lost on the wireless link. Agent B doesn't see 5, 4 and 3. Seeing 4 represented by 1, it concludes that 4 was a congestion loss and treats 5 and 3 as wireless losses. Hence it flips the bits again to 1010. In this manner, the receiver will obtain wireless losses represented by a 1 and congestion losses represented by a 0. Note that if

the receiver is a mobile node, it would have an agent inside it too.

Observe that packet reordering does not affect this scheme. Each packet will always hold the information about packets with preceding n sequence numbers, which are seen (or seen missing) ahead of it by the agent. The receiver will use the window information about a packet only if it doesn't receive that packet. For example, suppose that an agent sees packets 1..5 and then sees 7 followed by 6. 7's window will note 6 as lost and 6's window will note information starting from 5. To accommodate for packet reordering, the receiver can wait for certain number of subsequent packets before concluding a loss (as 3 packets in TFRC and TCP). It is important to observe two points here. Firstly, the agent simply inspects and labels a packet based on previously seen packets. It does not hold any packets either before or after the current packet. Secondly, the wireless and congestion losses will never be misclassified within the window. Although, it is possible that a packet which is not actually lost is marked lost within the window (due to reordering) or the information about a loss is not known (if the window is not large enough).

2.5 Implementation of AED

2.5.1 AED Agent Implementation

The AED agent can be deployed in between the wireless data link layer and the IP layer as shown in Figure 2.5. We show here, the implementation of the AED agent with DCCP [23]. Figure 2.5 also shows the packet structure which the agent sees. Each DCCP packet is shown encapsulated in a single IP packet (TFRC details have not been shown here). The agent uses the DCCP sequence number to detect packet loss. In the previous section, while explaining the mechanism, we considered packets flowing in one direction. In practice, both the agents will function in the same dual manner. When the IP packet is passed from the wireless data link layer to the IP layer, wireless losses will be detected and the agent flips the bits such that wireless loss is represented by a 1 and congestion or no loss by a 0 (agent after wireless link). When the packet is passed from IP layer to the wireless data link layer, congestion losses will be detected and the agent flips the bits such that congestion loss is represented by a 1 and wireless or no loss by a 0 (agent before wireless link).

The agent distinguishes between flows by looking at the tuple (source IP, dest IP, source port, dest port) in each IP packet. The TOS field in the IP packet can be used to detect if a flow is a DCCP flow. Since DCCP supports connection setup and tear-down, the agent knows when to allocate and free memory for a flow. An additional bit will be required to determine if a DCCP flow supports the use of AED or some other loss differentiation mechanism. The agent uses a space of w bits in the DCCP header to record the history window of size w. For each flow,

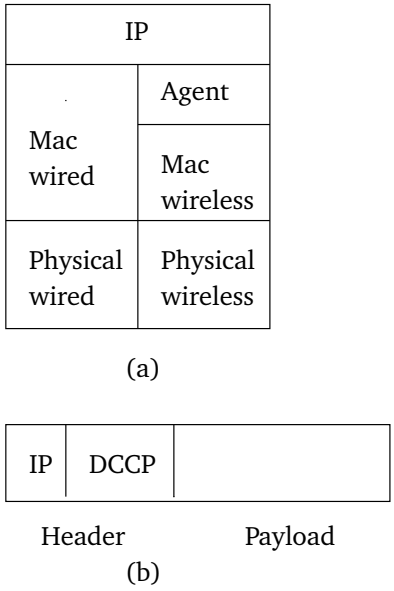


Figure 2.5: Packet structure and Agent implementation

each agent maintains a history of previous $w + k$ sequence numbers seen, where k serves as cushion for packet reordering. k could be set 3 as in TCP and TFRC. Thus per-flow memory overhead for the agent is only $O(w + k)$. Note here that, with this implementation, only the losses occurring within the IP layer will be classified as congestion losses. Any losses occurring below the wireless MAC layer will be classified as wireless (including possible losses at buffers inside the wireless Physical or MAC layer).

2.5.2 Receiver Implementation

The transport level implementation at the receiver is minimal. When the receiver receives an out of order DCCP packet, it will simply need to read the n window bits of this packet, to conclude the reasons of loss for the missing packets. With the convention used in section 2.4, if the window bit is set 1, it is a wireless loss and if it is set 0, it is a congestion loss. To deal with packet reordering the receiver will have to wait for k additional packets to find out if a packet is actually lost. In case of TFRC, the congestion losses will be used for calculation of loss rate.

2.6 Window size

To use the AED scheme, it is important to decide what window size to use. In this section, we investigate by simulations how the window size affects the throughput of TFRC when it uses AED for loss differentiation. Subsequently, we develop a closed form formula for the optimal

window size as a function of congestion and wireless loss rates. In AED, the window size used for loss differentiation depends on the length of end-to-end loss bursts experienced by the transport protocol, as a result of congestion and wireless links. With a window of certain size, the losses which fall within the window are correctly classified as wireless or congestion. The cause of losses which fall outside the window is not known and these losses are all classified as congestion. As a result, wireless losses which fall outside the window get misclassified as congestion. A window of size n consumes n bits within each data packet. As the window size grows, the transport protocol can determine the cause of losses more accurately. But a window of large size decreases the goodput since less data bits are transmitted per packet. We compare the throughput of TFRC which uses AED, denoted by *AED-TFRC*, with the throughput of *Omniscient-TFRC*(O-TFRC). O-TFRC is a hypothetical TFRC protocol which knows precisely which packets are lost due to congestion and which due to wireless errors. O-TFRC reduces its sending rate only in response to congestion. The throughput of a normal TFRC flow is proportional to

$$\frac{S}{R\sqrt{P}}$$

where S is the packet size, R is the round trip time and P is the total loss rate experienced by the flow due to wireless errors and congestion. The throughput of O-TFRC is proportional to

$$\frac{S}{R\sqrt{P_c}}$$

where P_c is the congestion loss rate. If AED-TFRC uses a window of size n, then its goodput will be proportional to

$$\frac{S-n}{R\sqrt{P_f(n))}}$$

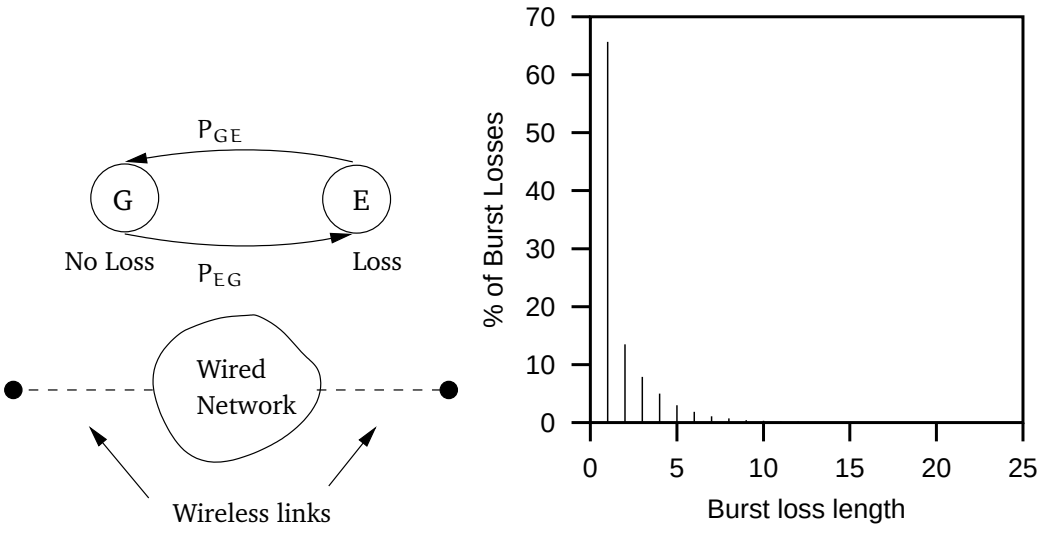
where $P_f(n)$ denotes the false congestion loss rate concluded by the protocol when a window of size n is used. $P_f(n)$ includes the congestion losses and the wireless losses which are misclassified as congestion. As the window size n increases, $P_f(n) \rightarrow P_c$ causing the goodput of AED-TFRC to increase. But as n increases, $(S-n)$ decreases causing the goodput to decrease. To determine the impact of window size of the throughput of AED-TFRC, we utilise the ratio

$$r(n) \quad = \quad \frac{goodput(\text{AED-TFRC})}{throughput(\text{O-TFRC})} = \frac{S-n}{S}\sqrt{\frac{P_c}{P_f(n)}} \tag{2.1}$$

$r(n)$ reaches maximum for the optimal value of window size n.

For simulation of wireless errors, we use the trace based error model by NGuyen, et al[25]. In [25], authors studied UDP packet traces on real wireless links to obtain an error model for wireless channels. They conducted their experiments using WaveLAN links in BARWAN wireless network. They used the UDP traces they collected to construct a simple two-state markov error

model (STMM) for wireless channels (Figure 2.6(a)). For this, they compute the mean error (L_E) and error-free (L_G) packet burst lengths of a trace and use the inverse of these lengths to compute transition probabilities of the STMM. $P_{GE} = 1/L_G$ and $P_{EG} = 1/L_E$. Authors of [25] collected several traces under different conditions. On average, $L_E = 2.618$ and $L_G = 166.394$, giving a wireless loss rate of 3%. They also observed that for 90% of loss bursts, the burst length was less than or equal to 4 packets. Thus a small window of size 4 in AED would be significantly effective. However, burst lengths may get augemented if there are several wireless links and due to congestion in the network.



(a) STMM and Topology used for simulation

(b) Distribution of burst loss lengths

Figure 2.6: Error Model, topology and distribution of loss burst lengths

Total packets	10^6
No.of loss bursts	20879
Maximum burst length	22 (1 burst)
Gross loss rate	3.96%
Congestion losses	25.14 % of losses
Wireless losses	74.86 % of losses

Figure 2.7: Loss statistics

We consider a simple topology with two wireless links separated by a wired subnetwork shown in Figure 2.6(a). For each wireless link, we simulate losses using STMM. Congestion

Window (n)	Bursts $\leq n$	Losses covered by n	Total Overhead(KB)
4	92.03 %	71.85 %	500
8	98.72 %	92.77 %	1000
16	99.98 %	99.75 %	2000

Figure 2.8: Loses covered by windows of different sizes

window (n)	Wireless losses misclassified (% of total losses)	M_w	M_c	False congestion rate $P_f(n)$	TFRC throughput ratio(%)
4	11.24 %	15.02 %	0	1.4452 %	83.26 %
8	1.83 %	2.44 %	0	1.0725 %	96.64 %
16	0.045 %	0.06 %	0	1.0018 %	99.91 %

Figure 2.9: Misclassification if unknown losses are considered congestion losses

losses in the wired subnetwork are simulated using a uniform packet loss probability of 1%. We do this for 10^6 packets and check the loss burst lengths seen by the receiver. Figure 2.6(b) shows the histogram of these loss burst lengths and figure 2.7 shows the corresponding loss statistics. A window of size n covers all loss bursts of length less than or equal to n and has an overhead of n bits per packet. Table 2.8 shows the losses covered by windows of various sizes versus the extra overhead for the connection. For bursts larger than the window size, the cause of certain losses is not known. If these unknown losses as classified as congestion, certain wireless losses will get misclassified as congestion. Figure 2.9 shows these misclassification rates. Based on the notation in [22], wireless misclassification rate M_w = (No. of wireless losses misclassified as congestion)/(total wireless losses) and congestion misclassification rate M_c = (No. of congestion losses misclassified as wireless)/(total congestion losses). For the simulation, packet size was 1000 bytes and the congestion loss rate P_c was 1%. Thus O-TFRC would have throughput proportional to $1000/R\sqrt{0.01}$ bytes per unit time. The fifth column in figure 2.9 shows the congestion loss rate $P_f(n)$ concluded by AED-TFRC for window size n. The last column shows the throughput ratio. Figure 2.10 shows the throughput ratio $r(n)$ as a function of window size n as n varies between 10 and 30. The optimal throughput of AED-TFRC (99.95%) was achieved for window size of 19.

We now give a closed form equation for the optimal window size as a function of the congestion and wireless loss rates experienced by TFRC protocol. For modeling, we assume a uniform loss model for both wired and wireless links. Assume a system in which a TFRC flow experiences a congestion loss rate of P_c and a wireless loss rate of P_w. Let P denote the total loss rate

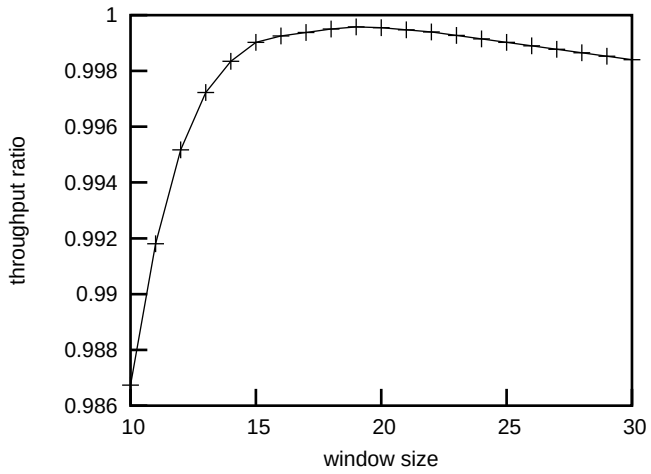


Figure 2.10: Throughput ratio $r(n)$ as a function of window size n

experienced by the flow. Then, P is given by

$$P = P_c + (1 - P_c)P_w$$

To obtain an expression for $P_f(n)$, observe that when a window of size n is used, a wireless loss following a group of n losses will be misclassified as congestion. Thus $P_f(n)$ is given by

$$P_f(n) = P_c + (P)^n(1 - P_c)P_w$$

Substituting $P_f(n)$ in (2.1), we get

$$r(n) = \frac{S - n}{S}\sqrt{\frac{P_c}{P_c + (1 - P_c)P_w(P)^n}}$$

$r(n)$ reaches maximum for the optimal value of n. This can be obtained as follows,

$$\frac{dr(n)}{dn} = 0 \quad \Leftrightarrow \quad (S - n)(P)^n = \frac{-2}{(\frac{1}{P_c} - 1)P_w \log(P)} \tag{2.2}$$

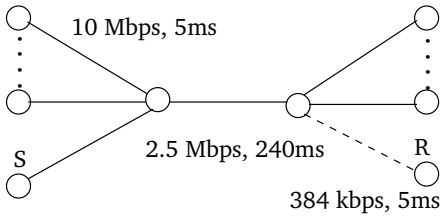
$$\Leftrightarrow \quad (S - n)(P)^n = c, \quad c > 0, S > 0, 0 < P \leq 1$$

$$\Rightarrow \quad n = S + \frac{Lambertw(\frac{-c\log(P)}{e^{S\log(P)}})}{\log(P)} \tag{2.3}$$

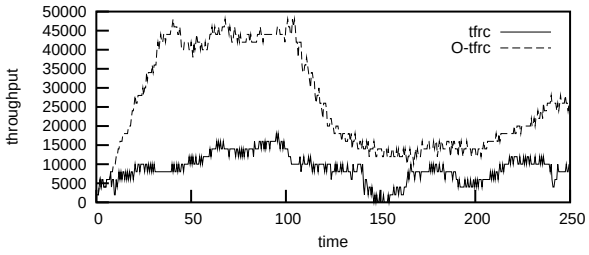
where c is a constant equal to the RHS of equation (2).

2.7 Comparison of AED with End-to-end loss differentiation schemes

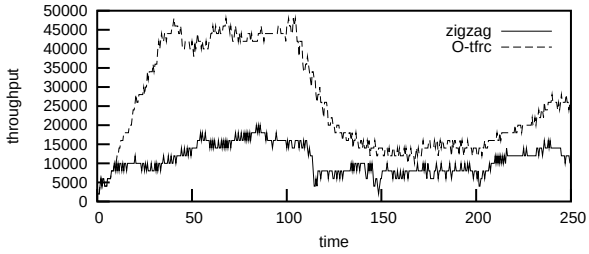
Authors of [22] studied the performance of various end-to-end loss differentiation schemes in conjunction with TFRC protocol using NS simulations. We compare the performance of



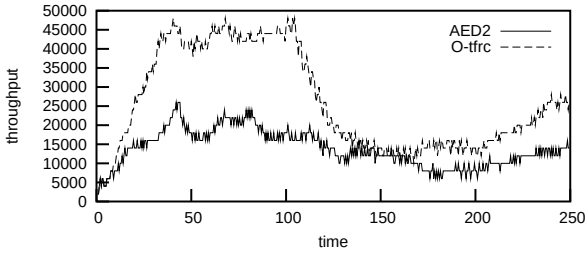
(a) Topology



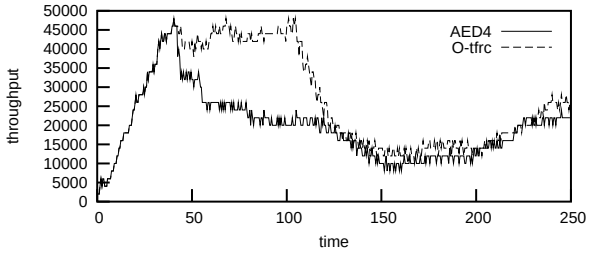
(b) Normal tfrc, $M_w = 100\%$



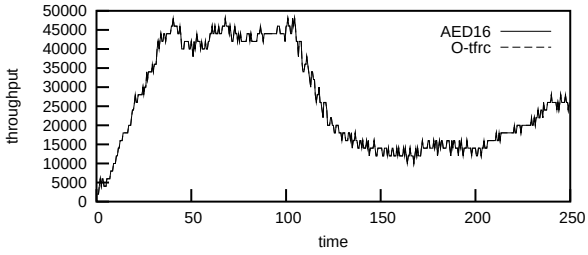
(c) Zigzag, $M_w = 67.3\%$



(d) AED (2 bits window), $M_w = 36.18\%$



(e) AED (4 bits window), $M_w = 11.76\%$



(f) AED (16 bits window), $M_w = 0.0$

Figure 2.11: Comparison of loss differentiation mechanisms using NS simulations

AED with end-to-end loss differentiation mechanisms. We use the dumb-bell shaped topology shown in figure 2.11(a) which represents the scenario for last-hop wireless networks. The wireless link has a bandwidth of 384 kbps and represents a last-hop link in mobile networks such as UMTS. Losses on this link are simulated using trace based STMM of NGuyen et al[25]. In [25], authors collected traces under different conditions and we observed that at low loss rates, there is not significant difference between TFRC and O-TFRC. To study the effectiveness of loss differentiation schemes, we use the parameters L_E and L_G from the trace with sufficient wireless losses. For the wireless link in our simulation, the transition probabilities were $P_{GE} = 0.965$ and $P_{EG} = 0.565$, producing a wireless loss rate of 8%.

A single TFRC flow exists between sender S and the mobile receiver R. All other flows are TCP, representing Internet traffic. For the entire simulation time of 250 seconds, 10 TCP flows coexist with the TFRC flow. At simulation time 100 seconds, 20 more tcp flows arrive and leave at 200 seconds, to produce some congestion. We repeated this simulation, each time using a different loss differentiation scheme for the TFRC flow. Figure 2.11 shows the throughput (in bytes) of normal TFRC, O-TFRC, Zigzag-TFRC and AED-TFRC. For our simulations, other end-to-end loss differentiations schemes yielded lower throughput than Zigzag. AED with a window size of 2 yields a higher throughput than Zigzag. With a window size of 16, the throughput of AED-TFRC is exactly same as O-TFRC. The wireless misclassification rate M_w for each TFRC is shown alongside each sub-figure.

2.8 Limitations

2.8.1 Header compression and IP security

In case of header compression at the IP layer, with IP and DCCP headers compressed, the AED agent will not be able to inspect the DCCP sequence number to detect loss. In order to do so, either it needs to be implemented within the IP layer or it remains outside and performs another independent compression and decompression. When protocols like IPSec are used, in case of authentication, the agent can only see but cannot change the DCCP header and when the IP packet is encrypted, it is not possible to either see or change the DCCP header present in the IP payload.

2.8.2 Increase of RTT

Although the procedure of inspecting and labeling the packets is extremely simple and can be implemented efficiently, if a packet passes through several agents it may result in slight increase of RTT.

2.8.3 Fragmentation

We assumed in the AED scheme that an entire DCCP packet is either lost or not lost. In case there is fragmentation of IP packets in the network, the loss of a DCCP packet will be detected and recorded only if IP packet containing the DCCP header(sequence number) would be lost. There is also the question of how to classify the losses after fragmentation. Suppose that an IP packet containing a DCCP packet is fragmented into two IP packets and subsequently one of them is lost due to congestion and the other on a wireless link. At present, AED will only record the case of the fragment which contains the DCCP sequence number. But fragmentation is not such a serious problem. In [26], an empirical study of Internet traffic was made. In the set of 60 traffic traces from the Internet that they collected, majority of them had fragmentation levels below 1%. Hence one would have very few or no packets per flow for which the precise cause of loss is not known.

2.8.4 Huge packet gaps

We believe that the topology shown in Figure 2.1 is the general topology of hybrid networks, but one cannot completely rule out cases in which a single flow splits and passes through two different wireless links. This is one of the cases when the agents will see huge packet gaps. But the packets in the gaps are not actually lost and can reach the receiver. It would be more useful to store information about packets ahead of the gap since these can get lost. When packet gaps exceed the window size, our scheme of defining and recording the window will not be able to record information about packets ahead of the gap. However, other schemes of defining and recording windows will add more overhead in terms of number of additional bits to be transmitted.

2.9 Conclusions

Accurate differentiation of wireless and congestion losses is necessary for proper functioning of several transport protocols on hybrid networks. In this chapter, we have explored the design and implementation requirements of AED, an accurate and explicit loss differentiation mechanism. We saw that by deploying AED agents which are aware of higher level protocols, one can accurately distinguish wireless from congestion losses. We investigated how the AED window size affects the TFRC throughput using simulations and analytical modeling. We showed that AED performs better than end-to-end loss differentiation schemes and yields higher TFRC throughput even with small window sizes. The success of several hybrid 3G networks depends on the performance of real-time protocols on these networks. We believe that explicit loss differentiation is a deployable solution in networks such as UMTS.

3

TRUSTWORTHY TOMOGRAPHY

3.1 Summary

Network tomography is a process by which internal characteristics of a network are inferred from "external" end-to-end measurements. To ensure that the inferred internal characteristics are sound and trustworthy, it is essential to verify the integrity of data collected from external measurements.

This work focuses on a particular method of performing network tomography called MINC (Multicast-based Inference of Network Characteristics), which infers characteristics of internal network links from end-to-end multicast measurements. MINC infers loss rates of network links by analyzing binary feedbacks which are reported by multicast receivers upon probe packets sent from the source. However, buggy or malicious multicast receivers may report incorrect feedbacks, leading to a faulty inference of link loss rates.

In this chapter, we show how the link loss rates inferred by MINC become erroneous if multicast receivers report incorrect binary feedbacks. Then, we develop a statistical verification algorithm called *ICheck* which can verify the integrity of binary feedbacks collected from multicast receivers. This algorithm performs statistical checks to determine if feedbacks of receivers are consistent with respect to one another. The algorithm does not require the knowledge of multicast tree topology and works even in the presence of colluding receivers. We present the performance of the algorithm on Model-based traces, NS traces, and MBone loss traces.

3.2 Introduction

In Medical Tomography, a doctor obtains the pictures of internal organs of a patient by using X-rays or ultrasound. This allows the doctor to analyze the patient's internal organs and possibly diagnose problems such as infections, blood clots and tumors without actually operating on the patient. Typically, the system which performs tomography consists of a block which emits X-rays or ultrasound and a recording block which captures the state of waves after they pass or reflect the patients body. Using the information supplied by the recording block, a picture of the patients body is constructed and used for diagnosis by the doctor.

Consider a situation which can occur in an old government hospital in any arbitrary country. Suppose that the recording block in the tomography system stops functioning correctly and starts reporting false information. In this case, a false image of the patients body will be obtained. A tumor present in the patient's stomach may have been displaced and shown in the kidney or a non-existent tumor could have been reported, leading to a false diagnosis and subsequent "treatment". In order to avoid such a situation, the information reported by the recording block needs to be verified for correctness. In this chapter, we consider an analogous problem, although less harmful to lives of human beings, with respect to network tomography.

Network Tomography refers to the process of inferring the internal characteristics of a network from end-to-end measurements. One of the goals of network tomography is to infer loss rates of links or paths in the network (wired). The loss rate of a link indicates the level of congestion on that link. Identification of lossy links or paths in the network is useful in the monitoring and management of networks. This information can be used by network operators and service providers to perform tasks of traffic engineering or to upgrade certain parts of their network.

Multicast-based Inference of Network internal Characteristics (MINC) is one of the earliest proposed methods of performing network tomography [7, 27, 28]. MINC can infer the internal characteristics of a network which lies under a multicast tree. Based on end-to-end multicast-based measurements, MINC can infer characteristics such as loss rates and delay distributions of network links [27, 28]. To infer loss rates, the source sends a stream of multicast probe packets into the multicast tree. Corresponding to each probe, each receiver reports whether it received the probe (1) or not (0). Based on the binary feedback traces collected from all receivers, per link loss rates in the multicast tree are inferred. The MINC loss inference is thus useful to monitor the level of congestion on internal network links.

Relying *solely* on "external" measurements to infer internal network characteristics is the essence of tomography. However, if one must use the inferred information in a trustworthy and reliable manner to make important decisions concerning the network, one must ensure the correctness of measured data. Due to configuration errors, software patches, and several

multi-platform implementations, networking software is often buggy and does not function as intended [29, 30]. For instance, in [31], bugs found during experimentation with NIMI were reported. Designers of NetLogger [32] point that 45% of problems in distributed applications arise due to the presence of bugs in networking software. Using incorrect feedbacks from buggy receivers will result in a faulty inference of link loss rates. A faulty inference may report high loss rates on good links and low loss rates on lossy links. Thus, to ensure a sound and trustworthy loss inference it is necessary to verify the integrity of binary feedbacks collected from multicast receivers.

This work presents an algorithm which can verify the integrity of binary feedbacks collected from multicast receivers in order to ensure a sound and trustworthy loss inference. Due to inherent correlations present in feedback traces of different multicast receivers, loss rates of paths in the multicast tree can be inferred in different ways. Our work exploits this idea to design a statistical verification procedure which detects loss rate inconsistencies that arise in erroneous feedback data. Furthermore, in conformance with the end-to-end nature of tomography, our procedure does not require any knowledge of the multicast tree topology.

3.2.1 Applications to Multicast Congestion Control

The problem of verifying feedbacks of multicast receivers is also of importance to multicast congestion control. In [33], authors showed that multicast receivers can report wrong feedbacks and mislead the multicast congestion control protocol to increase its sending rate, thereby harming other well behaved flows in the network. Therefore, for proper congestion control, feedbacks collected from multicast receivers must be checked for correctness. In practise, multicast receivers do not report per packet binary feedback but instead use RTCP (Real-Time Control Protocol) [10] to report summary loss information (fraction of packets lost within group of packets). But the problem of verifying binary feedbacks is still of interest to multicast congestion control for the following reasons.

First, studying the basic case of the problem (binary) allows us to understand the obstacles involved in tackling the general problem (summary loss information) better. Furthermore, our verification procedure is based on comparing loss rates of same links, inferred using feedback data collected from different sets of receivers. Authors of [34] have recently shown that loss rates of links can also be inferred using summary loss information present in RTCP packets. Therefore, we consider the problem of verifying the integrity of RTCP feedbacks of multicast receivers as future work.

Second, authors of [9] have implemented the RLE extension within RTCP XR packet type [35]. This allows multicast receivers to piggy-back per packet binary feedbacks which are needed to infer loss rates of links, on RTCP packets. If misbehaving receivers piggy-back correct binary feedbacks, these binary feedbacks would be inconsistent with the false RTCP

reports. The link loss rates inferred using binary feedbacks can be used to unearth congestion misbehavior. In order to hide this, a misbehaving receiver would need to report wrong binary feedbacks. Our verification procedure can be used in this context to check if receivers have misbehaved.

3.2.2 Contributions

This work presents two related contributions. Our first contribution is an analysis which explains how the loss rates inferred by MINC change when receivers report incorrect feedbacks. Our analysis proves that when receivers falsely report that they received the probe packet, the loss rates inferred by MINC on several links in the multicast tree can get altered.

For MINC loss inference, the binary feedbacks to N probes, reported by R receivers, are available in the form of a $N \times R$ binary feedback matrix. Given such a matrix that potentially contains incorrect feedbacks, the following questions can be posed:
(a) Is the given data erroneous, or equivalently, are the feedbacks of one or more receivers incorrect?
(b) Which are the receivers whose feedbacks are incorrect?
(c) In spite of errors, can we make the right MINC loss inference using the given feedback data?

Our second contribution is an algorithm called *ICheck* which answers (a). *ICheck* is a statistical procedure which searches for link loss rate inconsistencies that arise in erroneous feedback data. Broadly speaking, *ICheck* conducts statistical tests to determine the likelihood of obtaining the given feedback data from receivers of a multicast tree. *ICheck* uses the core principle of MINC loss inference itself. Like MINC, it is based on the premise that probe losses are independent across different links and independent between different probes (This is true in the presence of sufficient background Internet traffic [7]). *ICheck* takes as input only the $N \times R$ binary matrix and does not require the knowledge of the multicast tree topology.

From the discussion in this chapter, it will also become clear that questions (b) and (c) cannot be easily answered using an approach which relies solely on binary receiver feedbacks.

The rest of the chapter is organized as follows. Section 3.3 touches upon related work on network tomography and multicast congestion misbehavior. Section 3.4 explains the main principle used by MINC to infer loss rates of links in the multicast tree. Section 3.5 examines how the loss rates inferred by MINC change due to incorrect feedbacks. Section 3.6 presents the *ICheck* algorithm for feedback verification. Section 3.7 presents experimental results. Sections 3.8 and 3.9 present discussions and conclusions respectively.

3.3 Related work

In this section we describe aspects of network tomography and multicast congestion misbehavior relevant to this work.

3.3.1 Network Tomography

Network tomography involves estimating performance characteristics of networks (wired) based on a limited set of traffic measurements. The term ”network tomography” was coined by Y. Vardi due to similarities between medical tomography and network tomography. The literature concerning network tomography has come to comprise of two inverse problems: (i) Estimation of link level network characteristics based on measurements made on end-to-end paths. (ii) Estimation of path level network characteristics from measurements made on individual links.

Tomography of the first nature mainly comprises of MINC [7, 27, 28, 36, 37, 38] which infers link level characteristics from end-to-end multicast measurements. MINC can infer several link level characteristics such as loss rates, delay distributions, and delay variance. MINC infers the characteristics of links in the logical multicast tree. Each link in the logical multicast tree is a series of one or more physical links in the underlying network between two branch points. ”Losses on links” refer to the losses which occur due to buffer overflows at routers along the path in the underlying network. The delays of links comprise of the delays on physical links and router queues.

In order to infer the link level characteristics, MINC needs the topology of the logical multicast tree. This requirement goes against the end-to-end nature of tomography (of type (i) above). To overcome this constraint, algorithms which infer the logical multicast tree topology from probe measurements themselves, have been developed [39, 38]. These algorithms infer loss rates of shared paths between receiver pairs (using MINC) and use this information to construct the entire logical multicast tree topology (there is only one possible logical tree topology between a pair of receivers).

Origin-destination (OD) tomography falls under the second category [6, 40, 41]. In privately owned networks, it is possible to measure byte counts of traffic flowing in and out of link interfaces at routers. Given a set of such byte counts collected from various routers in the network, OD tomography involves the estimation of byte counts of traffic flowing between all origin-destination pairs in the network (known as the traffic matrix). Traffic matrices are important inputs required in the design and engineering of networks.

In this dissertation, we will be concerned with tomography of the first category and in particular with the estimation of link loss rates using MINC. The principle used by MINC to infer loss rates of links will be explained in section 3.4. The complete inference algorithm of

MINC loss estimator will be described in chapter 4 in conjunction with our work there.

3.3.2 Multicast Congestion Misbehavior

The problem of multicast congestion misbehavior was reported by Sergey, et al in [33]. In general, in congestion related misbehavior, flows that do not respond to congestion steal network bandwidth from those that do respond to congestion. By either not responding to congestion or by intentionally causing congestion, misbehaving flows can force the well behaved flows to back-off and claim the bandwidth that is left over. The problem is more serious in multicast congestion control since a multicast flow can affect a large number of flows in the network.

There are two paradigms to performing multicast congestion control[1]. In sender or Source-based Congestion Control (SCC), the sender calculates the rate at which packets must be injected into the network as a function of feedback reports received from all receivers in the group. In Receiver-based Congestion Control (RCC), the sender either splits or replicates the data into several different groups, each with a different sending rate. Each receiver subscribes to one or more groups based on its available network capacity and responds to congestion by un-subscribing to groups. Authors of [33] showed that both these approaches are vulnerable to receiver misbehavior. A misbehaving SCC receiver can send a wrong feedback and cause the sender to increase its sending rate, resulting in congestion. A misbehaving RCC receiver can inflate its group subscription level and cause congestion. Congestion forces well behaved flows in the network to reduce their sending rate, leaving more bandwidth for multicast flows of misbehaving receivers.

SCC protocols, which use a single rate approach are suitable for small sets of receivers and when the network is less heterogeneous in terms of link bandwidths. Protocols such as TFMCC [42], RMTP [43], PGMCC [44] and SAMM [45] use this approach. RCC protocols which use multi-rate approach are more suitable to heterogeneous networks. Protocols such as RLM [46], RLC [47], FLID-DL [48] use this approach. Several hybrid protocols such as MLDA [49], SARC [50] and DSG [51] also exist which use a combination of the two approaches. Sergey, et al [52] have proposed a misbehavior prevention mechanism for RCC protocols, which particularly suits the FLID-DL algorithm. Their prevention mechanism is based on in-band distribution of keys which guard access to multicast groups. These keys, which are split among multicast data packets, are lost when a receiver inflates its subscriptions to cause congestion. Thus, a receiver is automatically prevented from increasing its subscription level when congestion occurs.

Our verification algorithm which verifies binary feedbacks of multicast receivers can be

[1]This is with regard to multicast congestion control for unreliable flows which generally carry multimedia data

applied to detect multicast congestion misbehavior in case of SCC protocols. In the Internet, multicast flows which generally carry audio and video, use RTP/RTCP [10]. In this context, the per packet binary feedbacks reported via RTP XR packets can be verified using the algorithm presented in this chapter.

3.4 MINC

In this section, the principle used by MINC to infer the loss rates or the passage rates of links is described; passage rate = 1 − loss rate (The terms loss rate and loss probability are equivalent and so are passage rate and passage probability. If out of n packets sent on a link, m are lost, then the loss rate of the link is m/n and its passage rate is $(n-m)/n$).

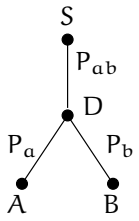


Figure 3.1: Multicast tree with two receivers

MINC infers the characteristics of a network underlying a multicast tree by exploiting the inherent correlation in multicast traffic. MINC infers loss rates in the logical multicast tree. Consider the logical multicast tree shown in figure 3.1 with source S, two receivers A, B and the branch node D. Suppose that the source sends a stream of probe packets and each receiver observes whether it receives the probe (1) or not (0). Consider the task of estimating the passage probability of the path DB. For this, consider those packets which were received by A. Since A received them, these multicast packets must have crossed the branch node D and also sent out on path DB. *Among them*, some may have crossed the path DB and some lost on this path. Thus, the ratio of the number of packets which both A and B received to the total number of packets which A received estimates the passage probability of path DB. The passage probability of path DA can be calculated similarly.

Formally, suppose that the sender injects N probe packets into the multicast tree. Let (i,j), $i,j \in \{0,1\}$ denote the *probe* corresponding to which A reported i and B reported j. Equivalently, (i,j) also denotes the *feedback* where A reported i and B reported j. Let n_{ij} denote the total number of probes of type (i,j). For example, n_{10} denotes the total number of probes for which A reported 1 and B reported 0. We extend this notation slightly by allowing $i,j \in \{0,1,*\}$, where "$*$" means a *dont care* (either a 0 or 1). For example, n_{1*} denotes the total number of probes for which A reported 1 and B reported either a 0 or 1; $n_{1*} = n_{10} + n_{11}$. Now, the passage

probability of the path DB, denoted by P_b and the passage probability of path DA, denoted by P_a are given by

$$P_b = \frac{n_{11}}{n_{1*}}, \quad P_a = \frac{n_{11}}{n_{*1}} \tag{3.1}$$

Similarly, the loss probabilities of path DB and DA are

$$\bar{P}_b = \frac{n_{10}}{n_{1*}}, \quad \bar{P}_a = \frac{n_{01}}{n_{*1}} \tag{3.2}$$

Having done this, the passage probability of path SD denoted by P_{ab} can be estimated as follows:

$$P_{ab} = \frac{n_{11}/n}{P_a \cdot P_b} = \frac{n_{*1}n_{1*}}{N \cdot n_{11}} \tag{3.3}$$

Since SD is the common path between A and B, P_{ab} is also called the *common passage probability*. The above principle is extended in MINC to calculate the passage probabilities of all the paths in the multicast tree. As mentioned before, although the topology of the multicast tree is needed to perform loss inference, the common passage probabilities calculated between different receiver pairs can be utilized to infer the topology of the multicast tree [39, 38]. Thus, loss inference can be performed entirely in an end-to-end manner using MINC. The principle used to infer the topology of a multicast tree from the common passage probabilities between different receiver pairs will be described in conjunction with our work in section 3.6.2.

3.4.1 Simple Observations I

An observation which aids the subsequent analysis is now made. Consider the (00) probe. This probe was either *(i)* lost on the path SD (denoted *common loss*) or *(ii)* it crossed SD and was lost simultaneously on paths DA and DB (denoted *independent loss*). We classify the (00) probes into these two respective categories. Let n_{00}^c denote the total number of probes lost on path SD. Let n_{00}^i denote the total number of probes which crossed SD and were lost simultaneously on DA and DB. Now, it is noted that $n_{00}^i + n_{01}$ are the total number of probes which crossed the path SD and lost on DA. *Among them*, n_{01} crossed DB and n_{00}^i were lost on DB. Thus, the passage probability P_b can also be written as

$$P_b = \frac{n_{01}}{n_{00}^i + n_{01}} \tag{3.4}$$

Thus, from (3.1) and (3.4) we have

$$P_b = \frac{n_{01}}{n_{00}^i + n_{01}} = \frac{n_{11}}{n_{1*}} \tag{3.5}$$

3.5 Misbehavior and its impact on passage probabilities

We use the term "misbehaving receiver" to denote the buggy or malicious receivers which can report incorrect feedbacks. A misbehaving receiver can misbehave either by altering a feedback from 0 to 1 (denoted by $0 \rightsquigarrow 1$) or by altering a feedback from 1 to 0 (denoted by $1 \rightsquigarrow 0$). When it misbehaves from $0 \rightsquigarrow 1$, it reports that it received the probe when it actually did not. When it misbehaves from $1 \rightsquigarrow 0$, it reports that it did not receive the probe when it actually did. If a receiver reports a wrong feedback, the passage probabilities inferred by MINC in the multicast tree change. Figure 3.2(a) shows the impact of misbehavior on the passage probabilities inferred by MINC when receiver A alters some of its feedbacks from $0 \rightsquigarrow 1$. The passage probability of the path from A to the source increases ($\Uparrow$) and the passage probabilities of all links connected to this path decrease ($\Downarrow$). Thus passage probabilities in a large region of the multicast tree are altered. Figure 3.2(b) shows the impact of misbehavior on the passage probabilities inferred by MINC when receiver A alters some of its feedbacks from $1 \rightsquigarrow 0$. In this case only the passage probability of the path from A to its parent decreases($\Downarrow$) to be congruous with data reported by A. The passage probabilities in the rest of the multicast tree remain unchanged.

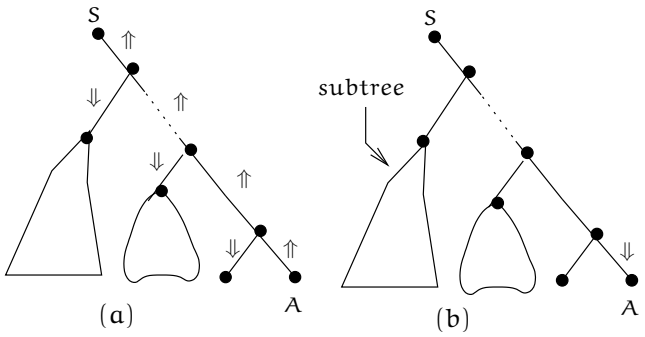


Figure 3.2: Effects of misbehavior on passage probabilities. (a) Receiver A reports more 1's, (b) Receiver A reports more 0's

To explain the above changes in probabilities, the misbehavior mechanism of a user is modeled as follows. It is assumed that a receiver j misbehaves with probability α_j ($0 \leq \alpha_j \leq 1$). If receiver j misbehaves from $0 \rightsquigarrow 1$, it changes its 0 feedback to 1 with probability α_j and vice versa.

Consider the two-receiver system shown in figure 3.1 with receivers A, B and sender S. Now, we calculate the *expected* passage or loss probabilities after misbehavior. Firstly, the misbehavior from $0 \rightsquigarrow 1$ is considered.

3.5.1 Receiver A misbehaves from $0 \rightsquigarrow 1$

After A misbehaves, let $\mathbb{P}_a$, $\mathbb{P}_b$ and $\mathbb{P}_{ab}$ denote the respective altered passage probabilities.

Lemma 3.1 *If A misbehaves from $0 \rightsquigarrow 1$ with probability α_a,* (i) $E[\mathbb{P}_b] \leq P_b$ (ii) $E[\mathbb{P}_a] \geq P_a$ (iii) $E[\mathbb{P}_{ab}] \geq P_{ab}$

Proof When A misbehaves from $0 \rightsquigarrow 1$, it causes two types of probe or feedback transformations:

$$x : (00) \Rightarrow (10) \quad y : (01) \Rightarrow (11)$$

The following table shows the *expected* feedback system after these transformations.

Original		After Misbehavior
n_{00}		$n_{00}(1-\alpha_a)$
n_{01}	$\Rightarrow$	$n_{01}(1-\alpha_a)$
n_{10}		$n_{10}+\alpha_a \cdot n_{00}$
n_{11}		$n_{11}+\alpha_a \cdot n_{01}$

The altered passage probability of path DB denoted by $\mathbb{P}_b$ will be,

$$\begin{aligned} E[\mathbb{P}_b] &= \frac{n_{11}+\alpha_a \cdot n_{01}}{n_{1*}+\alpha_a \cdot n_{00}+\alpha_a \cdot n_{01}} \\ &= \frac{n_{11}+\alpha_a \cdot n_{01}}{n_{1*}+\alpha_a(n^i_{00}+n_{01})+\alpha_a \cdot n^c_{00}} \end{aligned} \tag{3.6}$$

We know that $a/b = c/d \Rightarrow (a+c)/(b+d) = a/b = c/d$. Applying this to equation (3.5), we get

$$P_b = \frac{n_{11}+\alpha_a \cdot n_{01}}{n_{1*}+\alpha_a(n^i_{00}+n_{01})} \tag{3.7}$$

Subtracting (3.6) from (3.7) gives,

$$E[\mathbb{P}_b] = P_b\left\{1-\frac{n^c_{00}\cdot\alpha_a}{n_{1*}+n_{0*}\cdot\alpha_a}\right\} \qquad \Downarrow \tag{3.8}$$

Now,

$$E[\bar{\mathbb{P}}_a] = \frac{n_{01}(1-\alpha_a)}{n_{*1}} = \bar{P}_a(1-\alpha_a) \qquad \Downarrow \tag{3.9}$$

Proof for *(iii)* is similar to that of *(i)*. ∎

Intuitively, when A's feedbacks are altered from $0 \rightsquigarrow 1$, those feedbacks for which both A and B had reported 0, i.e. of the form (00), change to (10). Some of these (00) feedbacks correspond to probes which were lost on the path *SD*. These probes are now counted as having

crossed *SD* and lost on *DB*, causing the passage probability of *SD* to increase and the passage probability of *DB* to decrease.

In general, suppose that A misbehaves by causing n_x transformations of type x and n_y transformations of type y. The following corollary gives the conditions for observing the expected changes in probabilities after misbehavior.

Corollary 3.1 *If receiver A misbehaves from* $0 \rightsquigarrow 1$ *resulting in* n_x *and* n_y *transformations, then* $\mathbb{P}_b < P_b$ *and* $\mathbb{P}_{ab} > P_{ab}$ *if* $n_x/n_y > n_{10}/n_{11}$.

Proof After misbehavior we have,

$$\mathbb{P}_b = \frac{n_{11} + n_y}{n_{1*} + n_x + n_y}$$

$$\mathbb{P}_b < P_b \text{ iff } \frac{n_{11} + n_y}{n_{1*} + n_x + n_y} < \frac{n_{11}}{n_{1*}}$$

$$\text{i.e., iff } (n_{1*} - n_{11})n_y < n_{11}n_x$$

$$\text{i.e., iff } \frac{n_x}{n_y} > \frac{n_{10}}{n_{11}} = \frac{\bar{P}_a}{P_a}$$

Proof for P_{ab} is similar. ∎

Corollary 3.2 *If both receivers A and B misbehave from* $0 \rightsquigarrow 1$ *with the same probability, the receiver which suffers more losses before misbehavior causes a greater decrease in the passage probability of the other receiver.*

Proof After A and B misbehave from $0 \rightsquigarrow 1$ with probabilities α_a and α_b, the expected probe system is shown below.

Original		After Misbehavior
n_{00}		$n_{00}(1-\alpha_a)(1-\alpha_b)$
n_{01}	$\Rightarrow$	$n_{01}(1-\alpha_a) + \alpha_b(1-\alpha_a)n_{00}$
n_{10}		$(n_{10} + \alpha_a \cdot n_{00})(1-\alpha_b)$
n_{11}		$n_{11} + \alpha_a \cdot n_{01} + \alpha_b(n_{10} + \alpha_a \cdot n_{00})$

Working out $E[\bar{\mathbb{P}}_a]$ and $E[\bar{\mathbb{P}}_b]$ as before, we have

$$E[\bar{\mathbb{P}}_a] = (1-\alpha_a)\left\{\frac{n_{01} + \alpha_b \cdot n_{00}}{n_{*1} + \alpha_b \cdot n_{*0}}\right\} \tag{3.10}$$

$$E[\bar{\mathbb{P}}_b] = (1-\alpha_b)\left\{\frac{n_{10} + \alpha_a \cdot n_{00}}{n_{1*} + \alpha_a \cdot n_{0*}}\right\} \tag{3.11}$$

If $\alpha_a = \alpha_b$, the increase in each of the above depends on the ratios n_{00}/n_{*0} and n_{00}/n_{0*} respectively. ∎

3.5.2 Receiver A misbehaves from $1 \rightsquigarrow 0$

After A misbehaves, let $\mathbb{P}_a$, $\mathbb{P}_b$, and $\mathbb{P}_{ab}$ denote the respective altered passage probabilities.

Lemma 3.2 *If A misbehaves from $1 \rightsquigarrow 0$ with probability α_a,* (i) $E[\mathbb{P}_b] = P_b$ (ii) $E[\mathbb{P}_a] \leq P_a$ (iii) $E[\mathbb{P}_{ab}] = P_{ab}$

Proof If a receiver misbehaves from $1 \rightsquigarrow 0$, it causes two types of probe transformations: $\bar{x} : (10) \Rightarrow (00)$ and $\bar{y} : (11) \Rightarrow (01)$. Writing down the expected probe system after transformations as before, we will have

Original		After Misbehavior
n_{00}		$n_{00} + \alpha_a \cdot n_{10}$
n_{01}		$n_{01} + \alpha_a \cdot n_{11}$
n_{10}	$\Rightarrow$	$n_{10}(1-\alpha_a)$
n_{11}		$n_{11}(1-\alpha_a)$

$$
\begin{aligned}
E[\mathbb{P}_b] &= \frac{n_{11}(1-\alpha_a)}{n_{1*}(1-\alpha_a)} = P_b \\
E[\mathbb{P}_a] &= \frac{n_{11}(1-\alpha_a)}{n_{*1}} = P_a(1-\alpha_a) \qquad \Downarrow
\end{aligned}
$$

Proof for *(iii)* is same as that for *(i)* ∎

In general, suppose that A misbehaves by causing $n_{\bar{x}}$ and $n_{\bar{y}}$ transformations. The following corollary gives the general conditions for observing the expected changes in probabilities after misbehavior.

Corollary 3.3 *If A misbehaves from $1 \rightsquigarrow 0$ resulting in $n_{\bar{x}}$ and $n_{\bar{y}}$ transformations, then $\mathbb{P}_b = P_b$, $\mathbb{P}_{ab} = P_{ab}$ if $n_{\bar{x}}/n_{\bar{y}} = n_{10}/n_{11}$.*

Proof After misbehavior we have,

$$\mathbb{P}_b = \frac{n_{11} - n_{\bar{y}}}{n_{1*} - n_{\bar{x}} - n_{\bar{y}}}$$

$$\mathbb{P}_b = P_b \text{ iff } \frac{n_{11} - n_{\bar{y}}}{n_{1*} - n_{\bar{x}} - n_{\bar{y}}} = \frac{n_{11}}{n_{1*}}$$

$$\text{ie., iff } (n_{1*} - n_{11})n_{\bar{y}} = n_{11}n_{\bar{x}}$$

$$\text{ie., iff } \frac{n_{\bar{x}}}{n_{\bar{y}}} = \frac{n_{10}}{n_{11}} = \frac{\bar{P}_a}{P_a}$$

Similarly for $\mathbb{P}_{ab}$. ∎

The rest of the chapter concentrates on $0 \rightsquigarrow 1$ misbehavior, since $1 \rightsquigarrow 0$ misbehavior has almost no impact.

3.6 Algorithm for feedback verification

Broadly, given the $N \times R$ binary feedback matrix, the *ICheck* algorithm considers three receivers at a time and verifies the feedbacks of two particular receivers using the feedbacks of the third receiver. This verification is done by performing a test on the feedbacks of the three receivers. In this section, we describe this test in Lemma 3.3. Subsequently, we describe the *ICheck* algorithm.

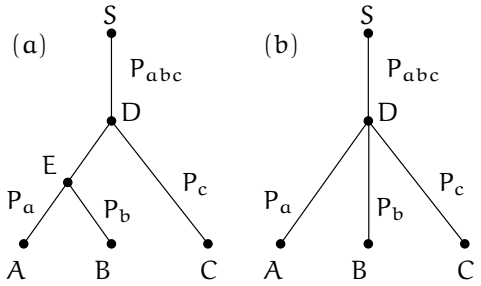


Figure 3.3: Multicast trees with three receivers

3.6.1 Simple Observations II

Figure 3.3 shows the two possible topologies which connect an arbitrary set of three receivers within a multicast tree. Consider the topology of Fig 3.3(a) with receivers A, B, C, and sender S. With usual notation, n_{ijk} denotes the number of probes for which A reports i, B reports j, and C reports k. Let P_c, P_a, P_b, and P_{abc} denote the passage probabilities of paths DC, EA, EB, and SD respectively. Consider the problem of estimating the passage probability

of path DC. Observe that, if either A or B received a probe, this probe must have reached the point D and must have been sent out on the path DC. Now, n_{10*} is a sample of probes received by A. Thus these probes crossed SD and were sent out on DC. Among them, n_{100} were lost on DC and n_{101} crossed DC. Thus P_c can be estimated as

$$P_c = \frac{n_{101}}{n_{10*}} \tag{3.12}$$

Using the same reasoning for the sample of probes n_{01*} which were received by B, P_c can also be estimated as

$$P_c = \frac{n_{011}}{n_{01*}} \tag{3.13}$$

From (3.12) and (3.13), we have

$$\frac{n_{100}}{n_{101}} = \frac{n_{010}}{n_{011}} = \frac{\bar{P}_c}{P_c} \tag{3.14}$$

Lemma 3.3 will show that the two ratios in (3.14) differ when feedbacks of either or both A and B are altered. An observation similar to the one in section 3.4.1 is now made. The n_{000} probes are split into two groups - n^c_{000} and n^i_{000}. Now, n^c_{000} are the number of probes lost on the path SD. n^i_{000} are those which crossed SD and were lost simultaneously in the *left subtree* rooted at D and on the path DC. Now it is observed that $(n^i_{000} + n_{001})$ crossed SD and were lost in the left subtree. Among them, n^i_{000} were lost on DC and n_{001} crossed DC. Thus we have,

$$\frac{n^i_{000}}{n_{001}} = \frac{\bar{P}_c}{P_c} \tag{3.15}$$

Lemma 3.3 *After* A *and* B *misbehave from* $0 \rightsquigarrow 1$ *with probabilities* α_a *and* α_b*, in the expected new system*

$$\frac{m_{100}}{m_{101}} \neq \frac{m_{010}}{m_{011}}$$

except when either of these conditions hold

(i) $P_{abc} = 1$

(ii) $\alpha_a/\alpha_b = n_{101}/n_{011}$

Proof After A and B misbehave, the relevant part of the expected new system is shown below. m_{ijk} denotes n_{ijk} in the expected system.

$$\begin{aligned}
m_{010} &= (n_{010} + \alpha_b \cdot n_{000})(1 - \alpha_a) \\
m_{011} &= (n_{011} + \alpha_b \cdot n_{001})(1 - \alpha_a) \\
m_{100} &= (n_{100} + \alpha_a \cdot n_{000})(1 - \alpha_b) \\
m_{101} &= (n_{101} + \alpha_a \cdot n_{001})(1 - \alpha_b)
\end{aligned}$$

Thus,

$$\begin{aligned}\frac{\mathbb{m}_{100}}{\mathbb{m}_{101}} &= \frac{n_{100}+\alpha_a\cdot n_{000}}{n_{101}+\alpha_a\cdot n_{001}} \\ &= \frac{n_{100}+\alpha_a\cdot n^i_{000}}{n_{101}+\alpha_a\cdot n_{001}} + \frac{\alpha_a\cdot n^c_{000}}{n_{101}+\alpha_a\cdot n_{001}} \end{aligned} \tag{3.16}$$

$$\begin{aligned}\frac{\mathbb{m}_{010}}{\mathbb{m}_{011}} &= \frac{n_{010}+\alpha_b\cdot n_{000}}{n_{011}+\alpha_b\cdot n_{001}} \\ &= \frac{n_{010}+\alpha_b\cdot n^i_{000}}{n_{011}+\alpha_b\cdot n_{001}} + \frac{\alpha_b\cdot n^c_{000}}{n_{011}+\alpha_b\cdot n_{001}} \end{aligned} \tag{3.17}$$

From equations (3.15) and (3.14) we have,

$$\frac{n_{100}+\alpha_a\cdot n^i_{000}}{n_{101}+\alpha_a\cdot n_{001}} = \frac{n_{010}+\alpha_b\cdot n^i_{000}}{n_{011}+\alpha_b\cdot n_{001}}$$

Thus, $\mathbb{m}_{100}/\mathbb{m}_{101} = \mathbb{m}_{010}/\mathbb{m}_{011}$ if

$$\frac{\alpha_a\cdot n^c_{000}}{n_{101}+\alpha_a\cdot n_{001}} = \frac{\alpha_b\cdot n^c_{000}}{n_{011}+\alpha_b\cdot n_{001}} \tag{3.18}$$

i.e., iff

(i) $n^c_{000} = 0 \Rightarrow P_{abc} = 1$ or

(ii) $\alpha_a/\alpha_b = n_{101}/n_{011}$

∎

Condition (ii) above essentially implies that

$$\frac{\alpha_a}{\alpha_b} = \frac{P_a(1-P_b)}{P_b(1-P_a)} \tag{3.19}$$

If A and B misbehave with the same probability then condition (3.19) *does not* hold *unless* P_a is also equal to P_b. Thus, even if A and B misbehave with the same probability the two ratios n_{100}/n_{101} and n_{010}/n_{011} would differ in most scenarios. Also, P_{abc} determines the amount of n^c_{000} probes available to distort the two ratios, making them different after misbehavior. If P_{abc} is low, there is a higher chance that the two ratios would differ more after misbehavior.

Lemma 3.4 *If* C *misbehaves from* $0 \rightsquigarrow 1$*, in the expected new system, both estimates of* P_c *(3.12) and (3.13) remain equal.*

Proof Assuming that C misbehaves with probability α_c, after misbehavior we have,

$$E[\bar{\mathbb{P}}_c] = \frac{n_{010}(1-\alpha_c)}{n_{01*}} = \frac{n_{100}(1-\alpha_c)}{n_{10*}}$$

∎

As a result of lemma 3.4, we have that lemma 3.3 holds irrespective of whether C misbehaves or not. (It is now noted that the above results of lemma 3.3 and 3.4 hold for the other 3-receiver topology of Fig 3.3(b) as well).

3.6.2 ICheck

Figure 3.4 presents the algorithm for feedback verification. *ICheck* examines the entire feedback data by considering feedbacks of three random receivers each time and applying the test of lemma 3.3 to detect inconsistencies. The test of Lemma 3.3 is applied in *HTest*. Lemma 3.3 tests the feedbacks of receivers A and B using the feedbacks of C. To apply this test on an arbitrary set of 3 receivers, *ICheck* needs to identify which of these receivers can function as C. For this, it uses the *LabelTree* procedure (Fig 3.4). This procedure uses the principle proposed in [39, 38] and labels the pair of receivers with minimum common passage probability as (A, B) and the other as C (Fig 3.5). Thus probes received by A or B would have also been sent out on the link DC. However, due to excessive feedback alterations, *LabelTree* may swap C with A or B. If A is swapped with C then the pair $(n_{010}/n_{110}, n_{001}/n_{101})$ is compared and if B is swapped with C then the pair $(n_{100}/n_{110}, n_{001}/n_{011})$ is compared. These unrelated ratios continue to remain different after A and B misbehave.

To perform the check of lemma 3.3, *HTest* is used. *HTest* is a standard statistical hypothesis test which is used to test the difference between two proportions. Given the feedbacks of three receivers A, B, and C, the following contingency table is constructed.

	Lost	Crossed	Total
Sample A	n_{100}	n_{101}	n_{10*}
Sample B	n_{010}	n_{011}	n_{01*}
Total	$n_{100} + n_{010}$	$n_{101} + n_{011}$	$n_{10*} + n_{01*}$

A two-tailed test is performed with *null hypothesis* H_0 and *alternative hypothesis* H_1 defined as,

$$H_0 : \frac{n_{100}}{n_{101}} = \frac{n_{010}}{n_{011}} \qquad H_1 : \frac{n_{100}}{n_{101}} \neq \frac{n_{010}}{n_{011}}$$

HTest tests whether the difference between the two ratios n_{100}/n_{101} and n_{010}/n_{011} is statistically acceptable with respect to the given sample sizes of sample $A(n_{10*})$ and sample $B(n_{01*})$. In this work, we use *Fisher's Exact Test* [53] as the representative statistical test. Fisher's exact test is a permutation test and outputs a *p-value* between 0 and 1. If the p-value is less than 0.05, null hypothesis H_0 can be rejected with 95% confidence, i.e., with 95% confidence one can say that feedbacks of either or both A and B are incorrect.

ICheck exploits the diversity of the multicast tree and the feedback data to overcome the weaknesses of lemma 3.3. Firstly, in each three receiver test, lemma 3.3 cannot check $0 \rightsquigarrow 1$

Procedure $\mathtt{ICheck}(F, k, \delta)$
$F[N \times R]$: Binary feedback matrix
$k \leq \binom{R}{3}$: Times to repeat
δ : confidence level

1: $\mathtt{inconsistent} \leftarrow 0$
2: **while** $k > 0$ **do**
3: $(x, y, z) = \mathtt{Random}(N)$ //same set not repeated
4: $(A, B, C) = \mathtt{LabelTree}(F, x, y, z)$
5: $\mathtt{failed} \leftarrow \mathtt{HTest}(F[A], F[B], F[C], \delta)$
6: **if** $\mathtt{failed}$ **then**
7: $\mathtt{inconsistent} \leftarrow \mathtt{inconsistent} + 1$
8: **end if**
9: $k \leftarrow k - 1$
10: **end while**
11: **print** $\mathtt{inconsistent}$

Procedure $\mathtt{LabelTree}(F, x, y, z)$

1: Compute P_{xy}, P_{yz}, P_{zx}
2: $\mathtt{temp} \leftarrow \min(P_{xy}, P_{yz}, P_{zx})$
3: **if** $\mathtt{temp} = P_{xy}$ **then**
4: **return**(x, y, z)
5: **else if** $\mathtt{temp} = P_{yz}$ **then**
6: **return**(y, z, x)
7: **else**
8: **return**(z, x, y)
9: **end if**

Figure 3.4: *ICheck* Algorithm

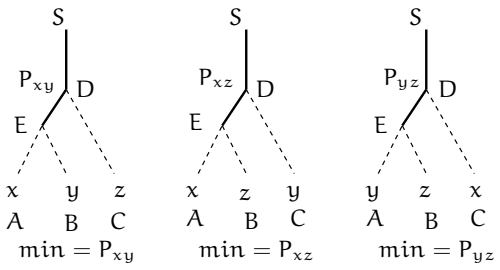


Figure 3.5: *LabelTree* Procedure : Minimum pair-wise common passage probabilities are shown in bold

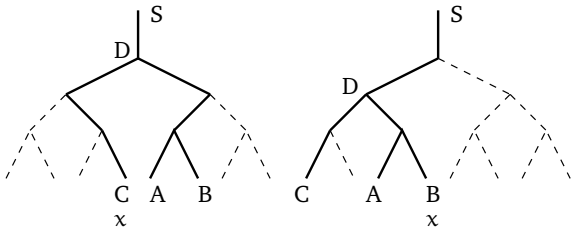


Figure 3.6: Functioning of ICheck

errors in C's feedbacks. However, since the algorithm tests several different three receivers sets, when a receiver x appears as C in one set, its feedbacks are not checked; but when it appears as A or B in another set, its feedbacks get checked (Figure 3.6). Secondly, the test of lemma 3.3 is weaker when passage probability of path SD, i.e. P_{abc} is very high. When a receiver x appears in one set of three receivers, the least common parent (node D) may be close to the source S resulting in $P_{abc} \approx 1$. But when it appears in another set of three receivers, node D may be far from the source, resulting in $P_{abc} < 1$ (Figure 3.6).

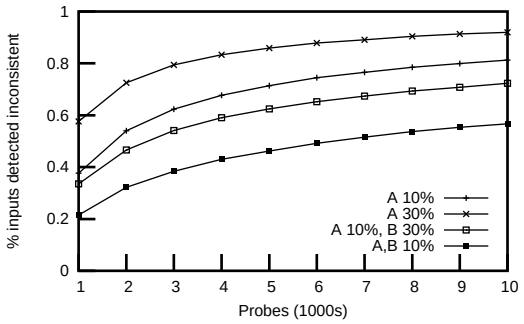
The complexity of *ICheck* varies with the number of three receiver sets checked. If k three receiver sets are tested, the complexity of *ICheck* is $O(Nk)$. As k increases, the integrity check becomes stronger. For the strongest check, when all $\binom{R}{3}$ three receiver sets are tested, k is $O(R^3)$.

3.7 Experiments

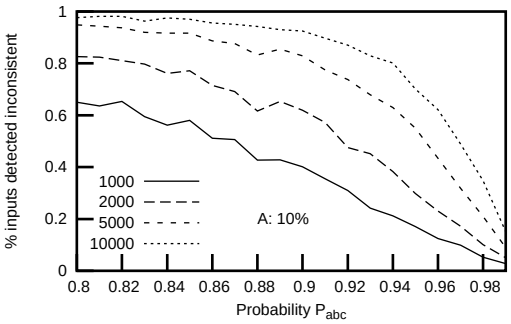
ICheck is a C program which we have implemented using the ideas discussed in the previous section. For Fisher's exact test, we use Algorithm 643 [54, 55] written in Fortran. To test the performance of *ICheck*, we have conducted experiments using model-based traces, NS traces and MBone traces. For Model-based traces, losses on each link are created using a time-invariant Bernoulli loss process. For NS simulations, losses on links occur due to buffer overflows at network nodes as the multicast probe competes with background TCP and UDP traffic. For MBone traces, we use the traces of a multicast audio session over the MBone.

3.7.1 Model Simulation

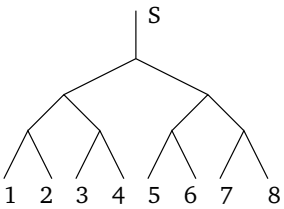
Model based simulations are used to study the performance of *ICheck* on a wide variety of inputs. The effectiveness of *ICheck* rests on the statistical test performed on three receiver sets. The factors from input data which influence these tests are *(i)* Number of probes *(ii)* Actual link loss rates in the multicast tree. These two factors work together to determine the sizes of two samples n_{10*} and n_{01*} which are compared. If these two samples are of small sizes and



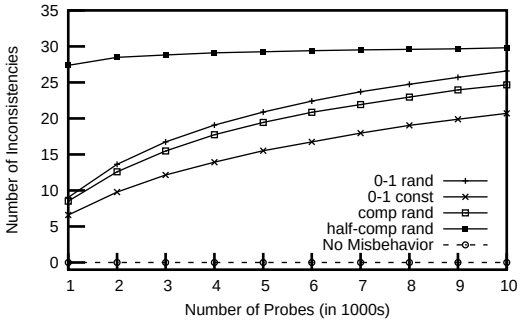
(a) Performance of the statistical test on different types of 3-receiver inputs



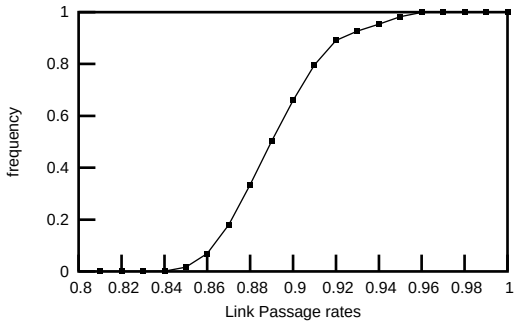
(b) Influence of P_{abc} in different types of 3-receiver inputs



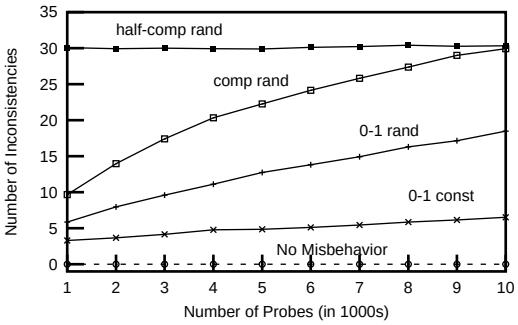
(c) 8-receiver tree



(d) Performance of *ICheck* on Model Based Traces



(e) CDF of link passage rates (NS simulation)



(f) Performance of *ICheck* on NS loss traces

Figure 3.7: Performance of *ICheck* Algorithm

their size difference is significant, only large alterations of feedbacks can be detected by the statistical test. As the sizes of these samples grow, their size difference matters less and even small alterations of feedbacks are detected. Whenever the samples are of comparable sizes, the detection is stronger. As the number of probes grow, sample sizes become large and eventually all alterations of feedbacks are detected.

Figure 3.7(a) shows the effectiveness of the statistical test performed by *ICheck* on different types of three receiver inputs. In this experiment, the three receiver topology of figure 3.3(a) was considered and the passage probabilities of all links were varied from 0.80 to 0.99 to yield an almost complete range of possible three receiver inputs which could be tested by *ICheck*. For each input the number of probes were varied from 1000 to $10,000$ probes. The *y-axis* in Fig 3.7(a) plots the percentage of inputs where the statistical test successfully detects the inconsistency with 95% confidence for different misbehavior mechanisms :(i)10% of receiver A's feedbacks are altered from 0 to 1 (ii)30% of receiver A's feedbacks are altered from 0 to 1 (iii)10% of receiver A's feedbacks and 30% of receiver B's feedbacks are altered from 0 to 1 (iv)10% of both A and B's feedbacks are altered from 0 to 1. As the number of probes increase, the detection is stronger. When receivers misbehave with the same probability, the detection is slightly weaker since the ratios n_{100}/n_{101} and n_{010}/n_{011} are less far-apart as compared to when receivers misbehave with different probabilities. Fig 3.7(b) plots the same results as a function of the passage probability P_{abc} for the case where 10% of receiver A's feedbacks are altered from 0 to 1. P_{abc} is crucial compared to other link probabilities since it determines the amount by which the two ratios can get distorted, when feedbacks are altered. As P_{abc} grows larger, the two ratios change less and more probes are needed for detection.

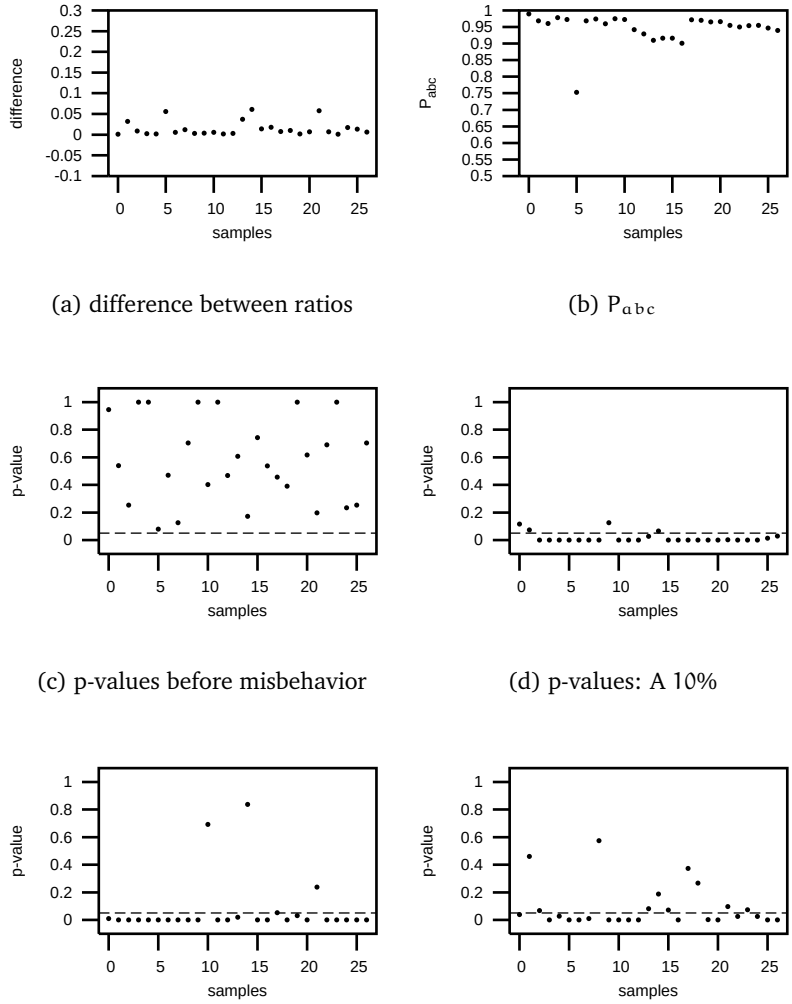
Next, the performance of *ICheck* was tested on general trees. Figure 3.7(d) shows the performance of *ICheck* on the 8-receiver complete binary tree shown in Fig 3.7(c). In this experiment, 1000 trees of the type in Fig 3.7(c) were generated with link passage probabilities varying uniformly from 0.80 to 0.99. For each tree, probes were simulated and loss traces obtained. In each trace, the following misbehavior mechanisms were introduced (i) Each receiver misbehaves from 0 to 1 with either 0%, 5%, 10% or 15% probability uniformly (0-1 rand). (ii) All receivers misbehave from 0 to 1 with with 10% probability (0-1 const) (iii) Each receiver complements 0%, 5%, 10%, or 15% of its feedbacks (comp rand) (iv) A random set of half of the receivers complement all their feedbacks (half-comp rand). In each trace, *ICheck* tested all the $\binom{8}{3} = 56$ three receiver sets. Figure 3.7(d) plots the average number of inconsistencies detected by *ICheck* for each misbehavior mechanism. When receivers complement their probes, the feedback matrix becomes quite inconsistent and several inconsistencies get detected. As the number of probes increase, the detection becomes stronger. The detection is weakest when all receivers misbehave with the same probability.

3.7.2 NS Simulation

For NS loss traces, the simulation parameters were setup as in the original work of MINC [27]. The 8-receiver complete binary tree shown in Fig 3.7(c) was considered. The bandwidth and propagation delay of each link were set 1.5Mbps and 10ms respectively. Each link was modeled as a FIFO queue with four-packet capacity. Node 0 sent 200 byte multicast probe packets with interpacket times chosen uniformly at random from 2.5 to 7.5 ms. We conducted 100 simulations and during each simulation, a variable amount of background traffic was introduced on each link in the tree using a random number of TCP and exponential on-off UDP flows. The probe losses observed by each receiver were used to generate the loss traces. Link passage rates in these simulations varied from 85% to 95%. Figure 3.7(e) shows the combined cummulative distribution function (CDF) of link passage rates for all links in the 100 simulations. For each trace four types of misbehavior mechanisms were considered as before. For each trace, *ICheck* tested all the $\binom{8}{3}$ three-receiver sets. Figure 3.7(f) plots the average number of inconsistencies detected by *ICheck* for each type of misbehavior mechanism. As observed before, when the number of probes increase, the detection becomes stronger.

3.7.3 MBone traces

For MBone traces, we analyzed the WRN traces collected by [56] and publicly available at the web site [57]. These traces correspond to multicast audio sessions of World Radio Network(WRN). Each dataset is about an hour of trace in which receivers in the multicast group recorded the sequence number of audio packets they received at 80ms intervals. The following traces were analyzed: WRNSep19, WRNNov1, WRNNov13, WRNNov14, WRNNov28, WRNDec1 and WRNDec11 (topologies for all these traces are shown at [57]). From each dataset, 3 receivers which experienced sufficient losses were chosen. Their traces were made binary based on whether an audio packet was received or not and divided into batches of size $10,000$ each. This resulted in a total of 27 3-receiver loss traces of size $10,000$ each. Figure 3.8(a),(b) and (c) show the properties of these samples. Figure 3.8(a) shows the difference between the ratios n_{100}/n_{101} and n_{010}/n_{011}, figure 3.8(b) shows the common passage probability P_{abc} for each sample, and figure 3.8(c) shows the p-value obtained when the two ratios were given to HTest. Since there is no misbehavior, the p-values for samples lie above 0.05. Figures 3.8(d),(e) and (f) show the p-values calculated after three types of misbehavior mechanisms : 3.8(d) 10% of receiver A's feedbacks were altered from 0 to 1 3.8(e) 10% of receiver A's feedbacks and 30% of receiver B's feedbacks were altered from 0 to 1 and 3.8(f) 15% of both receiver A and B's feedbacks were altered from 0 to 1. The p-values for most samples now lie below 0.05 indicating that one can conclude with 95% confidence that there is something wrong with the receiver feedbacks.



(a) difference between ratios

(b) P_{abc}

(c) p-values before misbehavior

(d) p-values: A 10%

(e) p-values: A 10%, B 30%

(f) p-values: A, B 15%

Figure 3.8: MBone Experiments

3.8 Discussion

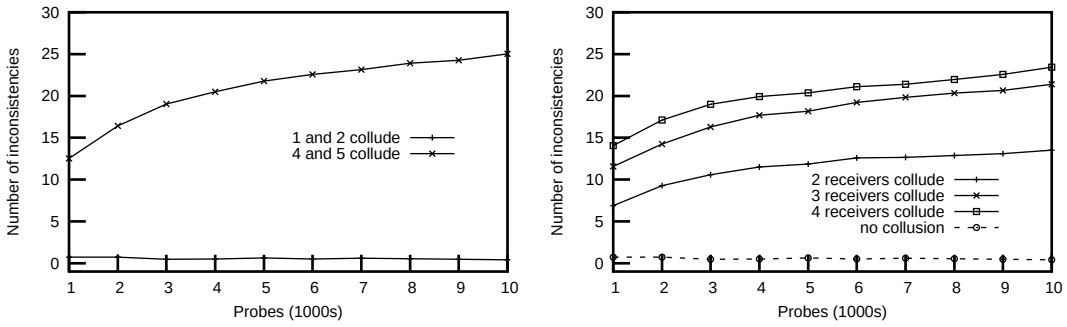
3.8.1 Collusion

ICheck is a consistency check. It checks if the feedbacks of all receivers are consistent with respect to one another. Due to this reason, it exhibits some resistance to collusion. Malicious receivers in a multicast tree can collude in small groups by reporting a 1 only when at least one member in the group received the probe packet. If two receivers have colluded together, their feedbacks may be consistent with respect to each other but inconsistent with respect to feedbacks reported by other receivers. Since *ICheck* tests different three-receiver groups, these inconsistencies could be detected.

Collusion introduces inconsistencies in two ways. First, collusion introduces spatial dependence between links. Spatial dependence (section 3.8.2) between non sibling links is detected as inconsistency by lemma 3.3. If two receivers which have colluded together appear as A and B in a set of three receivers, no inconsistencies would be detected (figure 3.3(a)). However, if two receivers which which have colluded appear as B (A) and C in a group of three receivers, probes flowing on link EB (EA) and DC would be dependent and this may result in an inconsistency. Second, when a receiver which has colluded appears as A or B in a group of three receivers, inconsistency may arise if losses on the common path to the three receivers have changed from 0 to 1. To illustrate the impact of collusion, we conducted the following experiment. The 8-receiver tree of Fig 3.7(c) was considered and passage probabilities of all links were varied from 0.80 to 0.99 to generate 100 random trees as before. In each tree, receivers 4 and 5 colluded with probability 50%, i.e., for 50% of the probes where at least one of the receivers received the probe, both of them reported 1. Fig 3.9(a) plots the average number of inconsistencies detected by *ICheck* after collusion. When 4 and 5 appear as B and C in (3, 4, 5), inconsistency could be detected due to the first reason above. When 4 appears as B in (3, 4, 2), inconsistency could be detected due to the second reason above. Fig 3.9(a) also shows that when 1 and 2 collude, no inconsistencies are detected. Fig 3.9(b) plots the average number of inconsistencies detected by *ICheck* when a random set of receivers collude in each tree, i.e., for 50% of the probes where at least one receiver in the set received the probe, all of them reported 1. In general, when receivers with least common parent high in the multicast tree collude, they result in more inconsistencies. If all receivers in the multicast tree collude, no inconsistencies would be detected.

3.8.2 Impact of Temporal and Spatial Dependence

In the MINC model, it is assumed that probe losses are independent across different links and independent between different probes. In reality, probes flowing on links in the Internet



(a) Two fixed receivers collude (in tree Fig 3.7(c))

(b) A random group of receivers collude (in tree Fig 3.7(c))

Figure 3.9: Collusion

may exhibit low levels of temporal or spatial dependence. Temporal dependence refers to the dependence between successive probes. For example, there may be back-to-back losses on a link. Spatial dependence refers to the dependence of probe behavior on different links. For example, in multicast overlay networks such as MBone, losses on sibling links may be dependent if these sibling links cross the same underlying physical path.

Studies based on real Internet traffic traces have concluded very low levels of temporal correlation beyond back-to-back packet losses [58, 56]. The test of lemma 3.3 remains valid in the presence of temporal correlations. To see why, consider the figure 3.3(a) (or (b)). Suppose that on all links SD, DC, DE, EA, and EB, the losses on each link occur in the form of a single loss or a back-to-back loss pair. Note that even in this case ratios compared by lemma 3.3 must be theoretically equal. However, in practise, we will need to compare larger probe samples corresponding to the two ratios as the ratios may not have converged to their average values as fast as in the case of single losses.

Spatial dependence between arbitrary links can render the test of lemma 3.3 invalid. Consider 3.3(a) again. Consider the case where losses on links DE and DC are dependant. Say for example, for about 10% of probes, a loss on DC implies a loss on DE. In this case, the ratios tested in lemma 3.3 must still be equal. However, consider the case where losses on DC and EB depend on each other but losses on DC and EA do not. Say for example, for about 10% of probes, a loss on DC implies a loss on EB. In this case, the ratios tested by lemma 3.3 will be unequal and the ratio on the left hand side will be larger than ratio on the right hand side. Authors of [56] studied spatial correlations in MBone. They checked the number of times that the same packet is not received by several receivers connected to MBone. They found this quantity to be low. Authors of [27] believe that large and long-lasting spatial loss dependence

is unlikely in real networks like the Internet due to traffic and link diversity.

3.8.3 Comparison to Nonce-based Scheme

To avoid receiver misbehavior, a *nonce* bit can be sent in every probe packet which receivers need to return in case they report that they received the probe packet. The nonce based scheme has some disadvantages. Firstly, since the probe packet is a multicast packet, all receivers receive the same nonce bit. Thus collusion becomes very easy and a receiver can collude with any arbitrary receiver to report more 1's. Secondly, it results in receivers reporting more bits. This can pose a constraint when MINC is used with RTCP and receiver feedbacks must occupy only 5% of data bandwidth [9]. Thirdly, it requires the change of existing protocols used for measurement.

On the other hand, *ICheck* checks the integrity of receiver feedback data. Whenever receivers misbehave in a manner which destroys the consistency of feedback data, *ICheck* succeeds. However, due to the nature of statistical tests conducted, *ICheck* is not a sufficient test, i.e, if *ICheck* detects inconsistency, it means that there is something wrong with receiver feedbacks; but if it does not detect inconsistencies, it means that most likely receivers feedbacks are correct. Depending on the loss rates in the multicast tree and the amount of misbehavior, it is feasible, although less likely, that the resultant feedback matrix after misbehavior remains entirely consistent.

3.9 Conclusions

In order to use end-to-end network inference in a trustworthy manner, it is essential to verify the integrity of receiver feedbacks. In this work, we showed how the MINC loss inference is affected by incorrect receiver feedbacks. We showed how the loss rates inferred by MINC change when incorrect feedbacks are received. Subsequently, we presented the *ICheck* algorithm which searches for loss rate inconsistencies that arise in erroneous feedback data. We presented the performance of *ICheck* on Model Based traces, NS traces and MBONE traces. Our experiments showed that *ICheck* successfully detects inconsistencies in the presence of different types of misbehavior mechanisms and even in the presence of collusion. *ICheck* does not require the multicast tree topology and thus does the affect the end-to-end nature of MINC tomography. The *ICheck* algorithm can be used in the phase before the MINC loss Inference to determine whether the inference based on the given feedbacks would most likely be trustworthy or not.

4

LOW FEEDBACK LOSS INFERENCE

4.1 Summary

To infer link loss rates, the MINC loss estimator requires each multicast receiver to report one bit of feedback per probe. This poses constraints when receivers report feedbacks using RTCP and the feedback bandwidth must not exceed 5% of data bandwidth.

One approach to reducing feedback bandwidth is to report less feedbacks, i.e., receivers can report feedbacks corresponding to a sampled set of probe packets, a process known as thinning [9]. The thinned set of feedbacks can then be used by the MINC loss estimator to infer loss rates of links.

In this chapter, we consider an alternate approach. We develop an extended MINC loss estimator (EMLE) which can infer link loss rates using *aggregate* feedbacks. Aggregate feedbacks consume less than 1 bit per probe and are reported upon groups of w consecutive probes. Aggregate feedbacks require between 1 and $\lceil \log_2(w+1) \rceil$ bits per w probes and thus help in the reduction of feedback bandwidth. EMLE extends the analysis of MINC loss estimator and performs loss inference from aggregate feedbacks without substantial loss of accuracy. We evaluate the performance of EMLE using model-based and NS simulations. We compare EMLE to the case where MINC loss estimator is used along with a thinned set of feedbacks.

4.2 Introduction

Multicast-based Inference of Network internal Characteristics (MINC) [7] is a method of performing network tomography in which characteristics of internal network links are inferred from end-to-end multicast measurements. MINC can infer internal characteristics of a network that lies under a multicast tree. MINC can infer characteristics such as loss rates and delay distributions of internal network links [28, 27]. To infer loss rates, the source injects a stream of probe packets into the multicast tree. Corresponding to each probe, each receiver reports whether it received the probe packet (1) or not (0). Using the binary feedbacks collected from all receivers, per link loss rates in the multicast tree are inferred. In this manner, the MINC estimator requires one bit of feedback per probe to perform loss inference.

In order to infer loss rates of links using active end-to-end multicast measurements, dedicated infrastructures to perform these measurements must be deployed. For large scale multicast measurements, the task of deployment is generally complex. To facilitate the measurement process, authors of [9] proposed a passive impromptu measurement architecture which couples the process of performing end-to-end multicast measurements with RTP/RTCP (Real-Time Transport and its Control Protocol) [10]. In this architecture, receivers within an existing multicast session which uses RTP to transfer data, can utilize their data packets as probes and report binary feedbacks upon the receipt or loss of RTP data packets by piggy-backing them on RTCP packets. These binary feedback traces can then be used to infer loss rates of links in the multicast tree. RTCP packets are multicast back to the group. Thus, any host which listens to these packets is able to perform loss inference. In this manner, the data being transferred by existing multicast sessions through the network can be utilized to infer link characteristics.

Including a binary feedback corresponding to the receipt or loss of every probe packet[1] within RTCP packets poses constraints in large multicast groups. This can cause the RTCP feedback bandwidth to exceed 5% of data bandwidth, a limit set aside by RTP for reporting purposes. In the architecture proposed by authors of [9], receivers report binary feedbacks upon a reduced set of probe packets to keep the feedback bandwidth low. The process of reporting less feedbacks is also called as *thinning* of feedbacks or equivalently thinning of probes. An example of thinning is say, for all receivers to report feedback upon the receipt or loss of every 5th probe packet. The thinned set of feedbacks is then used by the MINC loss estimator to perform loss inference. Eventually, whether the feedbacks are reported upon original or thinned set of probes, 1 bit of feedback is spent per probe.

In this work, we consider an alternate approach wherein receivers report less than N bits of feedback corresponding to all N probes. We develop an Extended MINC Loss Estimator (EMLE) which is able to perform loss inference using *aggregate* feedbacks. An aggregate feedback con-

[1] In this context, a probe packet is an RTP data packet

tains information about the *number of losses* observed within groups of w consecutive probes. Depending on its type, an aggregate feedback consumes between 1 and $\lceil \log_2(w+1) \rceil$ bits per w probes. Reporting aggregate feedbacks instead of per packet binary feedbacks results in the reduction of feedback bandwidth. EMLE can perform loss inference using aggregate feedbacks without significant loss of accuracy. We evaluate the accuracy and convergence rate of EMLE using model-based and NS simulations. We also perform simulations to compare EMLE to the case where MINC loss estimator is used along with a thinned set of binary feedbacks.

The rest of this chapter is organized as follows. In section 4.3, we describe how MINC measurements are piggy-backed in RTCP packets and explain certain aspects of MINC inference. In sections 4.4 and 4.5, we introduce the principles used by EMLE for loss inference. In section 4.6, we present a basic version of EMLE. The basic EMLE performs loss inference using aggregate feedbacks that require 1 bit per w probes. The analysis of basic EMLE is similar to the analysis of MINC loss estimator. In section 4.7, we present the generalized version of EMLE which can perform loss inference using all types of aggregate feedbacks that require between 1 and $\lceil \log_2(w+1) \rceil$ bits per w probes. In Section 4.8, we present experimental results. Sections 4.9 and 4.10 present discussions and conclusions respectively.

4.3 Related Work

4.3.1 RTCP for reporting MINC measurements

The proposals to utilize RTCP to report measurements for purposes of loss inference and network tomography appear in [59, 7, 9].

The specifications of Real-Time Transport Protocol (RTP) and its Control Protocol (RTCP) are described in RFC 3550 [10]. RTP is used to carry multicast audio and video data in the Internet and RTCP is used to provide feedback on the quality of the data distribution. RTCP packets are periodically multicast by each participant of a multicast session. RTP specification recommends that the collective RTCP feedback bandwidth must not exceed 5% of data bandwidth. Of this, 25% is allocated to the data senders and the remaining 75% (i.e., 3.75% of data bandwidth) is allocated to the receivers.

There are different types of RTCP packets such as SR (Sender Report), RR (Receiver Report), SDES (Source Description), etc. Each RTCP packet type contains a fixed size header followed by structural elements which may be of variable length. Different RTCP packet types can be stacked together to form a compound RTCP packet.

Each receiver in a multicast session periodically reports one compound RTCP packet, normally within a UDP packet. This compound RTCP packet typically contains one RR packet and one SDES packet, followed by zero or more RTCP packets. In order to report binary feed-

backs for MINC measurements, authors of [9] designed the RLE extension to RTCP XR packet type(eXtended Report), which is published in RFC 3611 [35]. By stacking an RTCP XR packet in a compound RTCP packet, receivers of a multicast session can report feedbacks needed for MINC loss inference.

RTCP has a built-in scaling mechanism to control the reporting bandwidth. Receivers of a multicast session collectively constrain the RTCP feedback bandwidth below 3.75% of data bandwidth. For this, each receiver independently sets its reporting timer and reports one compound RTCP packet at the end of each timer interval. The timer setting is a function of the recently observed RTCP traffic from other participants. By tracking the frequency and size of compound RTCP packets received, the participants collectively constrain the RTCP feedback bandwidth within specified limits.

Since the RTCP feedback bandwidth must be maintained below 3.75% of data bandwidth, receivers of a multicast session cannot report binary feedbacks corresponding to every probe packet. Thus, the feedbacks must be thinned. Since the space available to stack an RTCP XR packet is different at each receiver, each receiver must thin its feedbacks by a different thinning factor. However, for MINC loss inference, all receivers must report feedbacks corresponding to a common set of probe packets. To satisfy these constraints, authors of [9] designed a thinning mechanism in which each receiver thins its feedbacks by a factor of $1/2^{\mathsf{T}}$, i.e., each receiver reports feedbacks upon every 2^{T}th probe. The thinning exponent T is set independently by each receiver based on its perceived share of RTCP feedback bandwidth. In this manner, the probes for which different receivers report feedbacks, will overlap. Feedbacks reported upon probes corresponding to the largest thinning exponent will be available from all receivers.

An RTCP XR packet, like a regular RTCP packet, contains a fixed size header and this is followed by report blocks of variable length. The report blocks of type RLE are used to report binary feedbacks for MINC loss inference. Each RLE block contains three main parts (i) The start and end sequence numbers of probes being reported upon, (ii) the thinning exponent T used, and (iii) the binary feedback trace. For example, if the start and end sequence numbers of a trace are 13,821 and 13,865, and $\mathsf{T} = 2$ (i.e., the thinning factor is $1/2^2$), the binary feedback trace corresponds to probe sequence numbers : 13,824, 13,828 ...13,864 [35].

4.3.2 MINC loss Estimator

A good starting point for literature on MINC is [7]. The MINC loss estimator [27] estimates loss rates of links in the logical multicast tree. Each link in the logical tree corresponds to a series of one or more physical links between two branch points in the underlying real multicast tree. The losses on links in the logical tree essentially refer to the losses which occur due to buffer overflows at routers in the underlying path of the real multicast tree. Henceforth in this chapter, when we refer to multicast trees, we will be referring to the logical multicast trees.

MINC Loss estimator is a maximum likelihood estimator of loss rates of links in the multicast tree. For loss inference, the source sends a stream of probe packets and corresponding to each probe each receiver reports if it receives the probe or not. Based on binary feedback traces of all receivers, the MINC loss estimator infers the loss rates of links in the multicast tree. The loss rate of a link is defined as follows. If out of n packets sent on a link m are lost, the loss rate of the link is m/n. The passage rate of a link is $1 -$ loss rate. The MINC loss estimator is developed with the assumption that probe packet losses are independent across different links, and independent between different probes. This assumption has been shown to hold well in the Internet in the presence of large amounts of background traffic [7].

As mentioned in the last chapter, for loss inference, the MINC loss estimator requires the topology of the logical multicast tree. However, ideas of loss estimation from MINC can be used to compute the loss rates of shared paths between receiver pairs and this can be used to infer the topology of the logical multicast tree [39, 38]. The ideas of loss estimation in MINC have also been extended to estimate delay distributions and delay variance of network links [28, 36].

When binary feedbacks for MINC measurements are reported using RTCP, feedbacks may be lost since RTCP packets are sent using UDP . We shall comment on how the MINC loss inference is performed in presence of feedback losses in section 4.9.1. While developing the Extended MINC Loss Estimator, we will assume that feedbacks are not lost.

4.4 EMLE: Aggregate Feedbacks

For loss inference, as in MINC we assume that the source injects N probe packets into the multicast tree. But instead of reporting binary feedbacks corresponding to each probe packet, receivers report aggregate feedbacks corresponding to windows of w consecutive probes. The window size w is constant and common to all receivers. Thus, each receiver reports a total of N/w aggregate feedbacks. Aggregate feedbacks contain information about the number of probes lost within windows of w consecutive probes. However, they can be reported with a varying degree of precision level w', where $0 \leq w' \leq w - 1$. When the precision level is set to w', each receiver reports the actual number of probes lost if the number of losses is at most w'. If the number of losses is greater than than w', receivers simply report the value $w' + 1$. For example, when the precision level is set to 0, all receivers report feedbacks in the following manner:

$$\text{feedback} = \begin{cases} \mathbf{0} & \text{if } 0 \text{ losses among } w \text{ probes} \\ \mathbf{1} & \text{if } 1 \text{ or more losses among } w \text{ probes} \end{cases} \tag{4.1}$$

At the lowest precision level 0, receivers report only two values and thus each aggregate feedback requires only 1 bit per w probes. This results in a total per receiver feedback bandwidth

of N/w bits for N probes. At the highest precision level $w-1$, receivers report feedback values from the set $\{0,...w\}$ and thus each aggregate feedback requires $\lceil \log_2(w+1) \rceil$ bits per w probes. This results in a total per receiver feedback bandwidth of $(N/w)\lceil \log_2(w+1) \rceil$ bits for N probes. We note here that both the window size and the precision level remain constant for all aggregate feedbacks and are common to all receivers.

4.5 EMLE: Loss Distribution and Passage Probability

For a particular window size, as the precision level increases the feedback bandwidth increases. EMLE relates the feedback precision level to the accuracy with which it estimates the passage rates of links. As the feedback precision level increases, EMLE tries to make a more accurate inference of link passage rates.

The MINC loss estimator estimates the passage probability of all links in the multicast tree. For this, it first estimates the passage probability of paths from the source to each node in the multicast tree. The essential difference between the MINC loss estimator and EMLE lies in the method of estimation of passage probabilities of such paths.

Consider a path from source S to an arbitrary node k in the multicast tree, denoted by Sk. Now, the passage probability of the path Sk can also be viewed as "the probability of observing 0 losses on the path Sk given that 1 packet was sent from S". Let $A_k(i|w)$ denote the probability of observing i losses on the path Sk given that w packets were sent from S, where $0 \leq i \leq w-1$. Hence, the MINC loss estimator essentially estimates the passage probability of the path Sk as $A_k(0|1)$. We refer to the set of all elements $A_k(0|w), ...A_k(w-1|w)$ as the *loss distribution* of the path Sk. EMLE estimates some elements from the loss distribution of path Sk and subsequently computes its passage probability from the estimated elements.

When receivers report aggregate feedbacks for windows of w probes using a precision level of w', EMLE estimates the loss distribution elements $A_k(0|w), ...A_k(w'|w)$ corresponding to the path Sk. In other words, as the feedback precision level increases, EMLE estimates more loss distribution elements.

Let p_k denote the passage probability of path Sk. At the highest precision level $w-1$, when all elements of loss distribution are available, EMLE computes the passage probability of path Sk as

$$p_k = \sum_{j=0}^{j=w-1} \frac{(w-j)A_k(j|w)}{w} \tag{4.2}$$

At an intermediate precision level $0 \leq w' < w-1$, when a partial list of loss distribution elements is available, EMLE computes the passage probability of the path Sk by finding the root

of the following polynomial

$$\sum_{j=0}^{j=w'} \binom{w}{j} p_k^{w-j}(1-p_k)^j = \sum_{j=0}^{j=w'} A_k(j|w) \tag{4.3}$$

For example, at precision level 0, EMLE estimates only the first element of loss distribution of the path S_k i.e. $A_k(0|w)$. Subsequently, it computes the passage probability of path S_k as $\{A_k(0|w)\}^{1/w}$. When the window size $w = 1$, EMLE reduces to MINC loss estimator and both Eq. (4.2) and Eq. (4.3) estimate the passage probability of a path S_k as $A_k(0|1)$.

4.6 Basic EMLE

We begin by showing how EMLE estimates the passage rates of links in the logical multicast tree by using aggregate feedbacks which are reported at precision level 0. In this case, corresponding to each path from source to a node in the multicast tree, EMLE estimates only its first loss distribution element. We refer to this version of EMLE as the basic EMLE. The basic EMLE directly extends the inference algorithm of MINC loss estimator. Thus, when window size $w = 1$, the inference algorithms of basic EMLE and MINC are equivalent.

As usual, we assume that the source injects N probe packets into the multicast tree and receivers report $n = N/w$ aggregate feedbacks. Receivers report aggregate feedbacks at precision level 0, i.e., they report whether they observed 0 losses or greater than 0 losses within windows of w consecutive probes (as shown in Eq. (4.1)).

4.6.1 Principle

Before showing how loss rates of links are inferred in arbitrary multicast trees, we illustrate the core principle followed by basic EMLE using a 2-receiver tree. Consider the 2-receiver tree shown in figure 4.1 with receivers A and B, branch node D, and source S. P_A, P_B, and P_{AB} are the passage probabilities of paths DA, DB, and SD respectively. Our goal is to use the n 1-bit aggregate feedbacks reported by A and B and estimate the passage probabilities P_A, P_B, and P_{AB}. Consider the task of estimating the passage probability P_B of path DB. For this, we estimate the quantity $P_{DB}(0|w)$ that is the probability of 0 losses occurring on DB given that w packets are sent on DB. Then, P_B can be computed as

$$P_{DB}(0|w) = (P_B)^w \Rightarrow P_B = \{P_{DB}(0|w)\}^{1/w} \tag{4.4}$$

To estimate $P_{DB}(0|w)$, consider those aggregate feedbacks where A reported 0. Each of these feedbacks corresponds to A having received w probe packets (i.e., 0 losses). Hence, each time that A received w probe packets, these w multicast packets were also sent out on path DB.

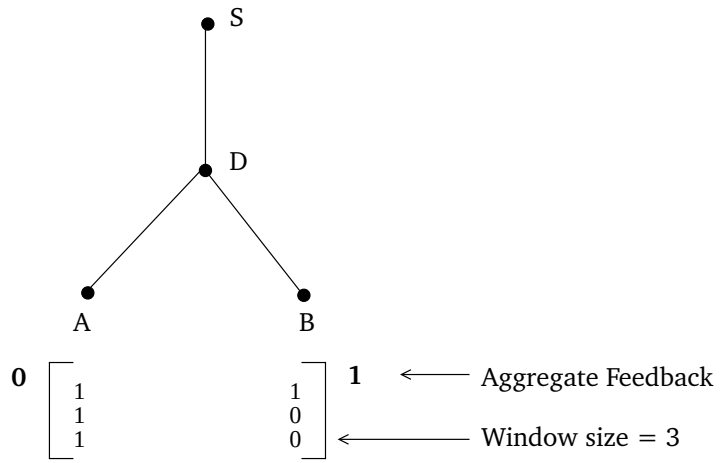


Figure 4.1: Two receiver tree

Thus, the ratio of the total number of feedbacks where both A and B reported 0 to the total number of feedbacks where A reported 0 gives $P_{DB}(0|w)$.

Formally, with usual notation, let n_{ij} denote the number of feedbacks where A and B reported i and j respectively, $i, j \in \{0, 1, *\}$. Thus, $P_{DB}(0|w)$, $P_{DA}(0|w)$, and $P_{SD}(0|w)$ can be estimated as follows.

$$P_{DB}(0|w) = \frac{n_{00}}{n_{0*}}$$
$$P_{DA}(0|w) = \frac{n_{00}}{n_{*0}}$$
$$\frac{n_{00}}{n} = P_{SD}(0|w)\ P_{DB}(0|w)\ P_{DA}(0|w)$$

Subsequently, P_A, P_B, and P_{AB} are computed as in Eq. (4.4).

4.6.2 Loss Inference for general Trees

For general trees, our goal is to utilize the feedbacks reported by *all* receivers to infer passage rates of links in the logical multicast tree. For analysis, we use the following notation similar to the MINC loss estimator. Let V denote the set of all nodes in the logical multicast tree. The set V consists of the source node S, the set of all receivers R, and the set of all branch nodes i.e., $V - R - S$. Let $f(k)$ denote the father of node k and let $d(k)$ denote the set of children of node k. Let $R(k) \subseteq R$ denote the set of receivers in the subtree rooted at node k. Instead of modeling the passage of each probe packet as in MINC, we model the passage of w consecutive probe packets through the multicast tree. We define the following quantities corresponding to each node $k \in V$.

$\mathbf{p_k}$ denotes the passage probability of the path from S to k

$\boldsymbol{\alpha_k}$ denotes the passage probability of a link terminating at node k (quantity to be eventually estimated).

$\mathbf{A_k(0|w)}$ denotes the probability of observing 0 losses on the path from source S to k given that w packets are sent from S.

$\boldsymbol{\gamma_k(0|w)}$ denotes the probability that at least one receiver in $R(k)$ observes 0 losses given that w packets are sent from S.

$\boldsymbol{\beta_k(0|w)}$ denotes the probability that at least one receiver in $R(k)$ observes 0 losses given that w packets are *sent from* $f(k)$.

$\mathbf{m_k(0)}$ denotes the number of times that at least one receiver in $R(k)$ reported the feedback 0.

$\mathbf{N}$, $\mathbf{n}$ N denotes the number of probes sent. n denotes the number of feedbacks reported. $n = N/w$.

The goal is to estimate the first element of loss distribution of paths from source S to each node k in the multicast tree. Subsequently, the passage probabilities of paths (p_k) and links (α_k) in the multicast tree are computed.

For a path from S to a node $k \in V$, its loss distribution element $A_k(0|w)$ is estimated by using the following basic principle: If at least one receiver in the subtree of node k observes 0 losses when w packets are sent from S, then there must have been 0 losses on the path from S to k. The quantities $A_k(0|w)$, $\forall k \in V$ are estimated in two steps as follows:
1. For each node $k \in V$, the quantity $\gamma_k(0|w)$ is estimated by using feedbacks reported by receivers in $R(k)$, i.e.,

$$\gamma_k(0|w) = \frac{m_k(0)}{n}$$

2. Since the probes are sent from S, we have

$$A_S(0|w) = 1$$

For receiver nodes, i.e., $k \in R$, we have

$$A_k(0|w) = \gamma_k(0|w)$$

For branch nodes, i.e., $k \in V - R - S$, we have

$$\gamma_k(0|w) = A_{f(k)}(0|w)\ \beta_k(0|w) \tag{4.5}$$

$$\gamma_k(0|w) = A_k(0|w) \left\{1 - \prod_{d \in d(k)} (1 - \beta_d(0|w))\right\} \tag{4.6}$$

From Eq. (4.5) and Eq. (4.6), we have

$$\gamma_k(0|w) = A_k(0|w)\left\{1 - \prod_{d\in d(k)}(1 - \frac{\gamma_d(0|w)}{A_k(0|w)})\right\} \tag{4.7}$$

Eq. (4.7) is a polynomial of degree $|d(k)|-1$ with unknown variable $A_k(0|w)$. Thus, $A_k(0|w)$ is estimated by finding its root. In [27], authors have shown that the solution of (4.7) exists and is unique in $(0,1)$ provided that $0 < \gamma_k(0|w) < \sum_{d\in d(k)}\gamma_d(0|w)$, which holds.

Subsequently, the passage probabilities of all links in the multicast tree are computed in two steps as follows:

1. For each node $k \in V$, p_k is calculated as

$$p_k = (A_k(0|w))^{1/w} \tag{4.8}$$

2. For each node $k \in V - \{S\}$, α_k is calculated as

$$\alpha_k = p_k/p_{f(k)}$$

For the source node, we have $\alpha_S = 1$.

When window size $w = 1$, the estimation algorithm of basic EMLE reduces to the estimation algorithm of MINC loss estimator and the passage probability of the path from S to k is estimated as $p_k = A_k(0|1)$. The basic version of EMLE does not require any explicit implementation. The quantities defined above are analogous to those used in MINC loss estimator, but have extended meanings. Giving the "new" aggregate feedbacks to an implementation of MINC loss estimator will result in the estimation of quantity $(\alpha_k)^w$ for each link terminating at node k. Quantities α_k are then obtained in a straight forward manner.

4.7 EMLE

The basic version of EMLE is able to process feedbacks of precision level 0. It computes the passage probability of the path from source S to a node k by estimating the first element of the path's loss distribution, i.e., $A_k(0|w)$. The general version of EMLE is able to process feedbacks of all precision levels. If the feedbacks are reported with a precision level of w', EMLE computes the passage probability of the path Sk by estimating the path's loss distribution elements $A_k(0|w), ...A_k(w'|w)$. Before presenting the inference algorithm of the general version of EMLE, we make two observations.

4.7.1 Similarities and differences with MINC delay estimator

The concept of loss distribution of a link is similar to the concept of its delay distribution. The number losses accumulated by a group of w probe packets on an end-to-end path is the

sum of losses accumulated on each individual link in the path. Similarly, the amount of delay accumulated by one probe packet on an end-to-end path is the sum of delays accumulated on each individual link in the path. Therefore, the analysis of EMLE resembles the analysis of MINC delay estimator. In fact, the way EMLE estimates the first element of loss distribution is exactly how the MINC delay estimator estimates the first element of delay distribution. However, the analysis of delay estimator cannot be used to estimate higher elements of loss distribution due the following difference between the two problems. Consider a path SA with two logical links SD and DA. For delay estimation, suppose that S sends a packet to A. The amount by which the packet is delayed on link SD is independent of the amount by which it is delayed on link DA. On the other hand, for loss estimation, suppose that S sends w packets to A. The number of losses accumulated *by the group of w packets* on link SD is not independent from the number of losses accumulated by the group on DA. For instance, if SD introduced one loss, DA cannot introduce w losses since only $w-1$ packets reach D. This difference prevents us from using the analysis of MINC delay estimator to estimate loss distribution elements.

4.7.2 Reduction of Computation

During the estimation of the first element of lost distribution $A_k(0|w)$ for a path Sk, we were required to find the root of a polynomial of degree $|d(k)|-1$ (Eq. (4.7)). We will see in the next section (4.7.3) that the root finding step is required in general to estimate each element of loss distribution. Finding the roots of polynomials of degree larger than two require the use of numerical methods which consume time and yield approximate solutions. To avoid this, we introduce a subtree partitioning procedure which partitions each subtree into exactly two subtrees. This allows us to estimate loss distribution elements using simple linear equations. In the next section (4.7.3), we will show how the partitioning is incorporated into the analysis by changing the definition of certain quantities (β_k's) and by defining additional quantities (γ_k's).

For each branch node k, we partition its subtree T_k, into two subtrees - T_{k_1} and T_{k_2}. For this, we simply pick each child subtree of k and assign it uniformly at random to either subtree T_{k_1} or subtree T_{k_2} (see figure 4.7.2). Note that both T_{k_1} and T_{k_2} are also rooted at k itself. Let $R(k_1)$ and $R(k_2)$ denote the set of receivers in T_{k_1} and T_{k_2} respectively.

4.7.3 Analysis

We assume that aggregate feedbacks are reported with a precision level of w', where $0 \leq w' \leq w-1$. The goal as before is to estimate loss distribution elements of paths from source S to each node k in the multicast tree and then to compute the passage probabilities of links.

For a path from S to a node $k \in V$, its loss distribution elements $A_k(0|w), ...A_k(w'|w)$ are

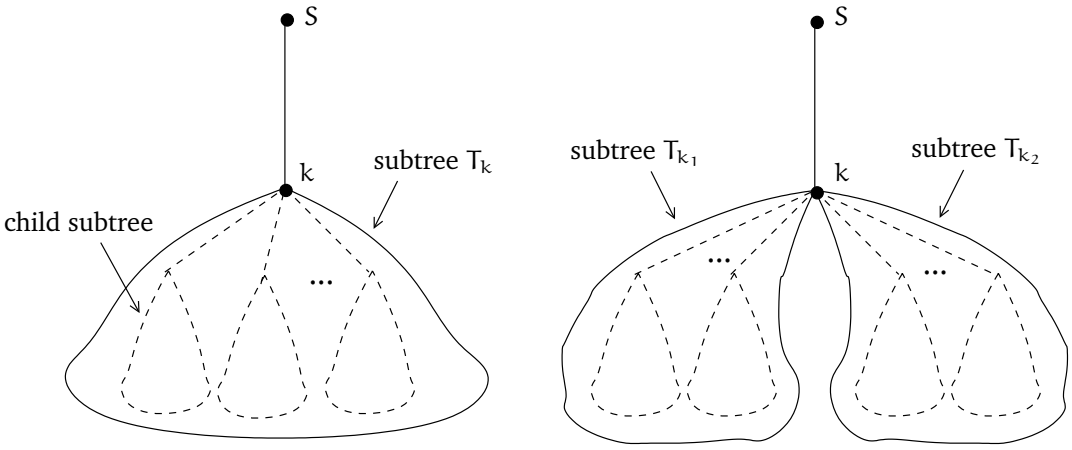


Figure 4.2: Original subtree (left) and the partitioned subtree (right)

estimated by using the following basic principle: If w packets are sent from S and the minimum number of losses observed by receivers in the subtree rooted at k is j, then there must have been at most j losses on the path Sk. Here, $0 \leq j \leq w$.

We define the following quantities corresponding to each node $k \in V$.

$\mathbf{m_k(i)}$, $\mathbf{m_{k_1}(i)}$, $\mathbf{m_{k_2}(i)}$, $\mathbf{m_{k_{1,2}}(i,j)}$ — $m_k(i)$ is the number of times that at least one receiver in $R(k)$ reports at most i losses. $m_{k_1}(i)$ is the number of times that at least one receiver in $R(k_1)$ reports at most i losses. Similarly $m_{k_2}(i)$ is defined in terms of $R(k_2)$. $m_{k_{1,2}}(i,j)$ is the number of times that at least one receiver each in $R(k_1)$ and $R(k_2)$ report at most i and j losses respectively.

$\mathbf{A_k(i|w)}$ — Probability that i packets are lost on the path from S to k given that w packets are sent from S.

$\boldsymbol{\gamma_k(i|w)}$, $\boldsymbol{\gamma_{k_1}(i|w)}$, $\boldsymbol{\gamma_{k_2}(i|w)}$, $\boldsymbol{\gamma_{k_{1,2}}(i,j|w)}$ — $\gamma_k(i|w)$ is the probability that the minimum number of losses received by a receiver in $R(k)$ is at most i given that w packets are sent from S. $\gamma_{k_1}(i|w)$ is the probability that the minimum number of losses received by a receiver in $R(k_1)$ is at most i given that w packets are sent from S. Similarly $\gamma_{k_2}(i|w)$ is defined in terms of $R(k_2)$. $\gamma_{k_{1,2}}(i,j|w)$ is the probability that the minimum number of losses received by a receiver in $R(k_1)$ is at most i and a receiver in $R(k_2)$ is at most j given that w packets are sent from S.

$\boldsymbol{\beta_k(i|w)}$, $\boldsymbol{\beta_{k_1}(i|w)}$, $\boldsymbol{\beta_{k_2}(i|w)}$ — $\beta_k(i|w)$ is the probability that the minimum number of losses received by a receiver in $R(k)$ is at most i given that w packets are *sent from* k. $\beta_{k_1}(i|w)$ is the probability that the minimum number of losses received by a receiver in $R(k_1)$ is at most i given that w packets are *sent from* k. Similarly $\beta_{k_2}(i|w)$ is defined in terms of $R(k_2)$.

With the above definitions, for a branch node $k \in V - R - S$, we have

$$\beta_k(i|w) = 1 - (1 - \beta_{k_1}(i|w))(1 - \beta_{k_2}(i|w)) \tag{4.9}$$

We start by estimating the quantities $\gamma_k,\ \forall k \in V$. For each node $k \in V$, we have,

$$\begin{aligned}
\gamma_k(i|w) &= \frac{m_k(i)}{n}, \quad 0 \le i \le w-1 \\
\gamma_{k_1}(i|w) &= \frac{m_{k_1}(i)}{n}, \quad 0 \le i \le w-1 \\
\gamma_{k_2}(i|w) &= \frac{m_{k_2}(i)}{n}, \quad 0 \le i \le w-1 \\
\gamma_{k_{1,2}}(i,j|w) &= \frac{m_{k_{1,2}}(i,j)}{n}, \quad 0 \le i \le w-1, 0 \le j \le i \\
\gamma_{k_{1,2}}(j,i|w) &= \frac{m_{k_{1,2}}(j,i)}{n}, \quad 0 \le i \le w-1, 0 \le j \le i
\end{aligned}$$

For source S, we have

$$A_S(0|w) = 1$$

For receiver nodes $k \in R$, we have

$$\begin{aligned}
A_k(0|w) &= \gamma_k(0|w) \\
A_k(i|w) &= \gamma_k(i|w) - \gamma_k(i-1|w), 1 \le i \le w'
\end{aligned}$$

For each branch node, i.e. $k \in V - R - S$, the following procedure is used to estimate the loss distribution elements $A_k(0|w), \ldots A_k(w'|w)$ corresponding to the path Sk. The loss distribution elements are estimated recursively starting from $A_k(0|w)$.

Step 0: (Base case)
Quantities calculated before: Nil
Quantities to be calculated in this step:

(i) $A_k(0|w)$

(ii) $\beta_{k_1}(0|w)$, $\beta_{k_2}(0|w)$

The above quantities can be calculated as follows.

$$\gamma_{k_1}(0|w) = \mathbf{A_k(0|w)}\ \beta_{k_1}(0|w) \tag{4.10}$$

$$\gamma_{k_2}(0|w) = \mathbf{A_k(0|w)}\ \beta_{k_2}(0|w) \tag{4.11}$$

$$\begin{aligned}
\gamma_k(0|w) &= A_k(0|w)\ \beta_k(0|w) \\
&= A_k(0|w)\ \left\{1 - (1 - \boldsymbol{\beta_{k_1}(0|w)})\ (1 - \boldsymbol{\beta_{k_2}(0|w)})\right\}
\end{aligned} \tag{4.12}$$

					$\mathbf{i}\|\boldsymbol{w}$
				$i-1\|w$	$\mathbf{i-1}\|\boldsymbol{w-1}$
				$i-2\|w-1$	$\mathbf{i-2}\|\boldsymbol{w-2}$
				$\vdots$	$\vdots$
		$2\|w$	...	$2\|w-i+3$	$\mathbf{2}\|\boldsymbol{w-i+2}$
	$1\|w$	$1\|w-1$	...	$1\|w-i+2$	$\mathbf{1}\|\boldsymbol{w-i+1}$
$0\|w$	$0\|w-1$	$0\|w-2$	...	$0\|w-i+1$	$\mathbf{0}\|\boldsymbol{w-i}$
step 0	step 1	step 2	...	step $i-1$	**step i ?**

Table 4.1: Quantities $\beta_{k_1}(x)$ and $\beta_{k_2}(x)$ calculated in each step are shown. The table shows the values taken by x. Quantities which are calculated in the ith step are shown in bold.

From Eq. (4.10),(4.11), and (4.12), quantities $A_k(0|w)$, $\beta_{k_1}(0|w)$, and $\beta_{k_2}(0|w)$ are obtained in a straight forward manner by solving only linear equations. Note the difference in the estimation of $A_k(0|w)$ using the above equations and the equations (4.5),(4.6), and (4.7) used earlier. With its old definition, β_k for a node k was expressed in terms of the node's child subtrees which could result in a large number of product terms if the degree of node was large. With its new definition, β_k for a node k is expressed only in terms of the node's two partitioned subtrees.

$\vdots$

Step i: Quantities calculated in previous steps $0,...i-1$ are :

(i) $A_k(\ell|w)$, $\forall \ell \in 0,...i-1$

(ii) The sets $\{\beta_{k_1}(\ell|w),...\beta_{k_1}(0|w-\ell)\}$ and $\{\beta_{k_2}(\ell|w),...\beta_{k_2}(0|w-\ell)\}$, $\forall \ell \in 0,...i-1$ (see table 4.1).

Quantities to be calculated in this step are:

(i) $A_k(i|w)$

(ii) The sets $\{\beta_{k_1}(i|w),...\beta_{k_1}(0|w-i)\}$ and $\{\beta_{k_2}(i|w),...\beta_{k_2}(0|w-i)\}$.

We estimate the above quantities in substeps $0,...i$. In substep j, $0 \leq j \leq i-1$, we estimate the elements $\beta_{k_1}(i-j|w-j)$ and $\beta_{k_2}(i-j|w-j)$. In substep i, we estimate quantities $A_k(i|w)$, $\beta_{k_1}(0|w-i)$, and $\beta_{k_2}(0|w-i)$.

Substep 0: Quantities $\beta_{k_1}(i|w)$ and $\beta_{k_2}(i|w)$ are estimated as follows.

$$\begin{aligned}\gamma_{k_{1,2}}(i,0|w) &= A_k(0|w)\ \boldsymbol{\beta_{k_1}(i|w)}\ \beta_{k_2}(0|w)\\ \gamma_{k_{1,2}}(0,i|w) &= A_k(0|w)\ \beta_{k_1}(0|w)\ \boldsymbol{\beta_{k_2}(i|w)}\end{aligned}$$

$\vdots$

Substep $i-1$: Quantities $\beta_{k_1}(1|w-i+1)$ and $\beta_{k_2}(1|w-i+1)$ are estimated as follows.

$$\begin{aligned}\gamma_{k_{1,2}}(i,i-1|w) &= \sum_{\ell=0}^{\ell=i-2} A_k(\ell|w)\ \beta_{k_1}(i-\ell|w-\ell)\ \beta_{k_2}(i-1-\ell|w-\ell)\\ &+ A_k(i-1|w)\ \boldsymbol{\beta_{k_1}(1|w-i+1)}\ \beta_{k_2}(0|w-i+1)\end{aligned}$$

$$\begin{aligned}\gamma_{k_{1,2}}(i-1,i|w) &= \sum_{\ell=0}^{\ell=i-2} A_k(\ell|w)\ \beta_{k_1}(i-1-\ell|w-\ell)\ \beta_{k_2}(i-\ell|w-\ell)\\ &+ A_k(i-1|w)\ \beta_{k_1}(0|w-i+1)\ \boldsymbol{\beta_{k_2}(1|w-i+1)}\end{aligned}$$

Substep i: Quantities $A_k(i|w)$, $\beta_{k_1}(0|w-i)$, and $\beta_{k_2}(0|w-i)$ are estimated using the following equations.

$$\begin{aligned}\gamma_{k_1}(i|w) &= \sum_{\ell=0}^{\ell=i-1} A_k(\ell|w)\ \beta_{k_1}(i-\ell|w-\ell)\\ &+ \boldsymbol{A_k(i|w)\ \beta_{k_1}(0|w-i)}\\ \gamma_{k_2}(i|w) &= \sum_{\ell=0}^{\ell=i-1} A_k(\ell|w)\ \beta_{k_2}(i-\ell|w-\ell)\\ &+ \boldsymbol{A_k(i|w)\ \beta_{k_2}(0|w-i)}\\ \gamma_k(i|w) &= \sum_{\ell=0}^{\ell=i-1} A_k(\ell|w)\ \{1-(1-\beta_{k_1}(i-\ell|w-\ell))\ (1-\beta_{k_2}(i-\ell|w-\ell))\}\\ &+ \boldsymbol{A_k(i|w)}\ \{1-(1-\boldsymbol{\beta_{k_1}(0|w-i)})\ (1-\boldsymbol{\beta_{k_2}(0|w-i)})\}\end{aligned}$$

$\vdots$

Step w':

Substep 0

$\vdots$

Substep w'

When feedbacks are reported at the precision level $w' = w-1$, in the final step $w-1$, $A_k(w-1|w)$ is estimated. Depending on whether all loss distribution elements are estimated or the partial list of loss distribution elements is estimated, either Eq. (4.2) or Eq. (4.3) is used to compute passage probability p_k of path Sk.

After estimating $p_k \; \forall k \in V$, $\alpha_k \; \forall k \in V - \{S\}$ is computed as before, as $\alpha_k = p_k / p_{f(k)}$. $\alpha_S = 1$. In total, EMLE takes $O(|V|w'^2)$ steps to estimate passage rates of all links in the tree when feedbacks are reported at precision level of w'.

We note here that the subtree partitioning concept (section 4.7.2) which is used in the inference algorithm of EMLE is general and can be incorporated in the analysis of both MINC loss and delay estimators. This allows both these estimators to avoid finding roots of polynomials with large degree, thus minimizing their computation time. When window size $w = 1$, EMLE reduces to the MINC loss estimator in which the subtree partitioning concept is incorporated.

4.7.4 Impact of Temporal Dependence

Estimation of passage probability in EMLE consists of two steps: (i) Estimation of loss distribution elements and (ii) Computation of passage probability using loss distribution elements. While modeling, we assumed independence between probes. In practise, there may exist some amount of dependence between consecutive probes, i.e., temporal correlation. The process of estimating loss distribution elements is not affected by dependence of probes within a window. However, the computation of passage probability from loss distribution elements may be affected by temporal correlations.

When receivers report feedbacks with precision level $w' < w - 1$, EMLE estimates a partial list of loss distribution elements and computes the passage probability of paths using Eq. (4.3). Eq. (4.3) assumes that all probes which fall within one window are independent. Thus, in the presence of temporal correlations, estimating the passage probability using Eq. (4.3) introduces errors in estimation.

When receivers report feedbacks with precision level $w' = w - 1$, EMLE estimates all loss distribution elements and computes the passage probability of paths using Eq. (4.2). The advantage of using Eq. (4.2) is that it does not assume independence between probes which fall within one window.

4.8 Experiments

We study the behavior of EMLE through model-based and NS simulations. In model-based simulations, losses on links in the logical multicast tree are created using a time-invariant Bernoulli loss process. In NS simulations, losses on links occur due to buffer overflows as the probe packets compete with background TCP and UDP traffic. We simulate the passage of probes through the multicast tree and use the loss traces observed by receivers to construct their aggregate feedbacks. These aggregate feedbacks are then given to EMLE for loss inference. To construct aggregate feedbacks we use window sizes 1 to 7. For each window size w,

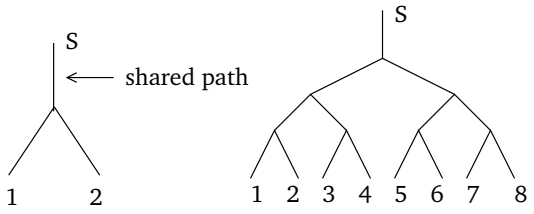


Figure 4.3: 2-receiver (left) and 8-receiver (right) trees used for experiments

we construct aggregate feedbacks with precision level 0 and $w-1$. With feedbacks of precision level 0, EMLE estimates the first element of loss distribution and uses Eq. (4.3) to compute passage probabilities. With feedbacks of precision level $w-1$, EMLE estimates all elements of loss distribution and uses Eq. (4.2) to compute passage probabilities.

To assess the accuracy of estimation, we compute the relative error of a link. The relative error of a link which terminates at node k is defined as

$$RE_k = |(\alpha_k - \widehat{\alpha}_k)/\alpha_k|$$

where α_k is the true passage rate of the link and $\widehat{\alpha}_k$ is the passage rate estimated by EMLE.

We first perform experiments with model-based simulations. We start by performing experiments using a 2-receiver multicast tree and use EMLE to estimate the passage probability of the shared path (figure 4.3). We perform two sets of experiments wherein passage rates of links vary uniformly from $90-95\%$ and $95-99\%$ respectively. We repeat experiments 100 times and plot the measures of location corresponding to the relative error. Figure 4.4 shows the relative error for the shared path for window sizes 1 to 7 with probes varying from 1K to 10K. Each sub-figure shows (i) the average relative error and (ii) median of relative error along with its 5th and 95th percentiles for 10K probes. The sub-figures in the left column show the relative error when passage probability of the shared path is calculated using the first element of loss distribution. The sub-figures in the right column show the relative error when passage probability of the shared path is calculated using all elements of loss distribution. When window size $w = 1$, EMLE reduces to the MINC loss estimator. We observe that, for passage rates $95-99\%$, there is very little difference between EMLE and MINC loss estimator. In general, for passage rates $90-99\%$, the passage probability estimated using all elements of loss distribution is slightly more accurate than the passage probability estimated using only the first element of loss distribution.

Figure 4.5 displays the standard error for estimating the passage rate of the shared path. The standard error is calculated over 100 experiments using a fixed link passage rate (90% first row, 95% on the second row). The standard error reflects the variance of the estimator. The standard error is high when passage rates are low and window sizes are large. The error is generally lower when all elements of loss distribution are used to estimate the passage probability.

The accuracy of EMLE depends on the accurate estimation of loss distribution elements. From the above experiments, we observe that the error and variance in estimation increase when window sizes or loss rates of links increase. When there in an increase in either of these two parameters, the number of feedback samples used to estimate lower lost distribution elements decreases, resulting in a less accurate estimate of the passage probability. For instance, estimation of the first loss distribution element $A(0|w)$ requires feedback samples which correspond to all probes in a window being received (i.e., 0 losses among w probes). The number of such samples decreases with the increase in window size or loss rates of links and thus the passage probability calculated from the first loss distribution element becomes less accurate. The loss distribution elements are estimated recursively starting from the first element. Thus, when window sizes or loss rates of links increase, errors in the estimation of lower elements affect the estimates of higher elements and even the passage probability calculated from all elements of loss distribution becomes less accurate.

Figures 4.6(a) and (b) show the performance of EMLE for the 8-receiver binary tree of figure 4.3, for model-based simulations in which passage rates of links vary from 90-99%. The figure displays the relative error in the estimation of passage probability of one of the internal links in the tree. As the height of the tree increases, the end-to-end loss rates increase. This affects the accuracy of estimation.

Next, we evaluate the performance of EMLE using NS simulations. We use the 8-receiver binary tree of figure 4.3 and setup simulation parameters similar to the original work on MINC [27]. The bandwidth and propagation delay of all links in the multicast tree are set 1.5Mbps and 10ms respectively. Each link is modeled as a FIFO queue with four-packet capacity. The source sends 200-byte multicast probe packets with inter-packet times chosen uniformly at random from 2.5 to 7.5 ms. This simulation setup is a scaled down version of a realistic setting in the Internet. Real router buffers in the internet usually have a capacity which is much larger than 4 packets. The scaled down setup is used to minimize the simulation time. We conducted 100 simulations and during each simulation, a variable amount of background traffic was introduced on each link using a random number of TCP and exponential on-off UDP flows. The resulting link passage rates in these simulations varied from 85% to 95%. The probe losses observed by each receiver were used to construct the aggregate feedbacks as in model-based simulations and were given to EMLE for loss inference.

Figures 4.6(c) and (d) show the performance of EMLE for the 8-receiver binary tree, for NS simulations. The figure displays the relative error in the estimation of passage probability of one of the internal links in the tree. In NS simulations, when the passage probability is estimated using the first element of loss distribution, the relative error is high. This is due to the presence of temporal correlations in NS simulations. The losses on the end-to-end paths from source to receivers are not perfectly Bernoulli. Due to this reason, using Eq.(4.3) to estimate passage

probability yields errors. Figure 4.7 displays the autocorrelation(lag 1) in the loss trace of a receiver present in the 8-receiver tree for model-based and NS simulations. In case of model-based simulations, the loss trace is perfectly bernoulli and thus the autocorrelation stays around 0. In case of NS simulations, due to presence of temporal correlations, the autocorrelation varies between 0.10 and 0.15.

4.8.1 Thinning with MINC loss estimator

By performing loss inference on aggregate feedbacks, EMLE helps in the reduction of feedback bandwidth. The alternate method to reduce feedback bandwidth is to simply report less binary feedbacks. That is, receivers can report binary feedbacks corresponding to a thinned (i.e., sampled) set of probe packets and these binary feedbacks can be given to the MINC loss estimator for loss inference.

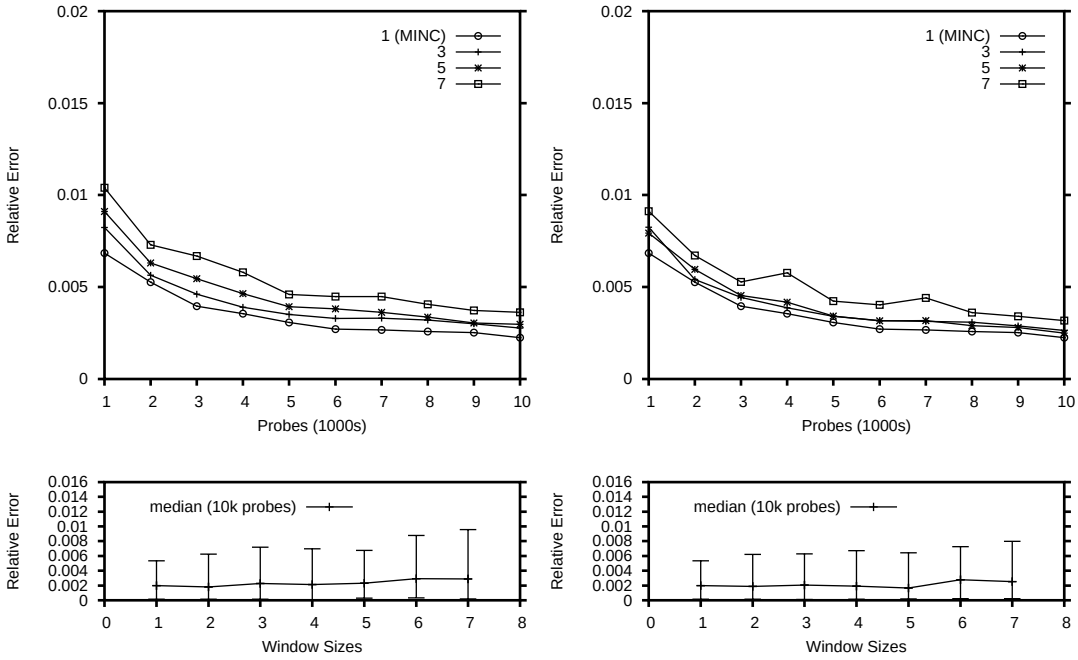
Feedbacks can be thinned to various levels to reduce feedback bandwidth. To thin feedbacks by a factor of t, $0 \leq t \leq 1$, we pick each binary feedback from the original set of feedbacks with a uniform probability of t. In this manner, if the original feedbacks set is of size S, the expected size of the thinned set is t times S.

We compare the passage rates estimated by EMLE using aggregate feedbacks, to the passage rates estimated by MINC loss estimator using thinned binary feedbacks. For this, we thin binary feedbacks in such a manner that the thinned set of feedbacks requires the same bandwidth as the set of aggregate feedbacks. Subsequently, we use the MINC loss estimator for loss inference and repeat experiments performed in the previous section.

In the previous section, we evaluated the accuracy of passage rates estimated using aggregate feedbacks constructed with window size w and precision level 0 and $w-1$. Aggregate feedbacks of precision level 0 require 1 bit per w probes. Aggregate feedbacks of precision level $w-1$ require $\lceil \log_2(w+1) \rceil$ bits per w probes. Corresponding to these two cases, we thin binary feedbacks by factors of $1/w$ and $\lceil \log_2(w+1) \rceil / w$ respectively. As before, we experiment with window sizes 1 to 7.

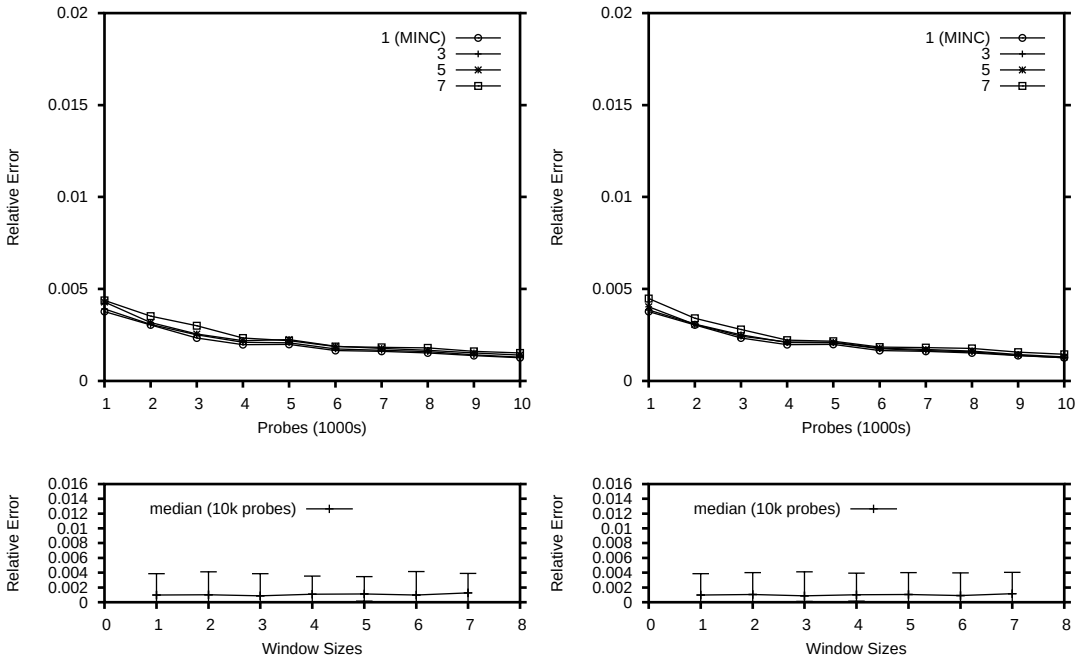
We start with model-based simulations for a 2-receiver tree. As before, we perform two sets of experiments where passage rates of links vary uniformly from 90-95% and 95-99% respectively. Figure 4.8 shows the relative error in the estimation of passage rate of the shared path when feedbacks are thinned by factors $1/w$ (left column) and $\lceil \log_2(w+1) \rceil / w$ (right column). The average and median of relative error are calculated over 100 simulations.

Figure 4.9 shows the standard error in the estimation of passage rate of the shared path using thinned feedbacks. The standard error is calculated over 100 experiments using a fixed link passage rate (90% first row, 95% on the second row). Figure 4.10 shows the relative error in the estimation of passage rate of one of the internal links using thinned feedbacks for an 8-receiver binary tree in case of model-based and NS simulations.



(a) Passage rate 90-95%, using $A(0|w)$

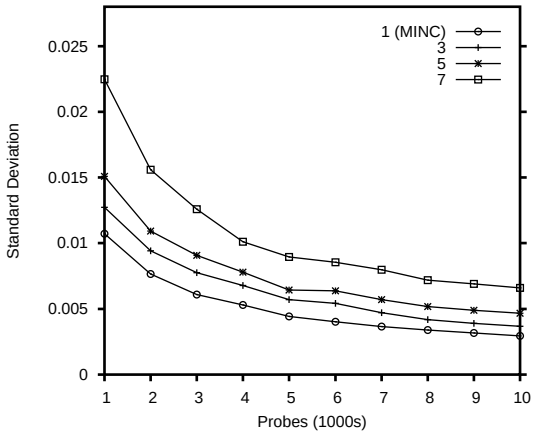
(b) Passage rate 90-95%, using $A(0|w), ...A(w-1|w)$



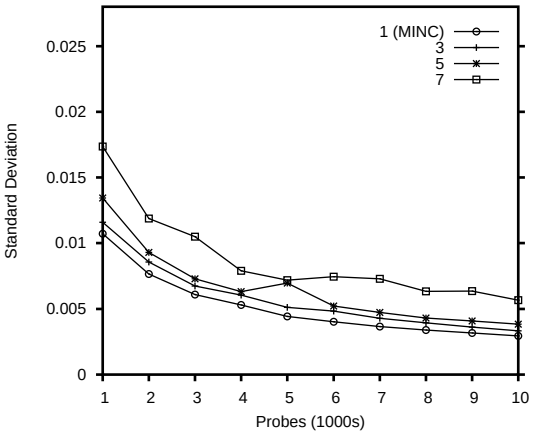
(c) Passage rate 95-99%, using $A(0|w)$

(d) Passage rate 95-99%, using $A(0|w), ...A(w-1|w)$

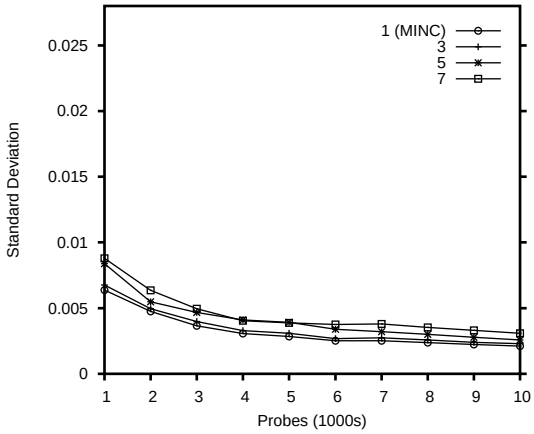
Figure 4.4: EMLE: Relative error of the shared path



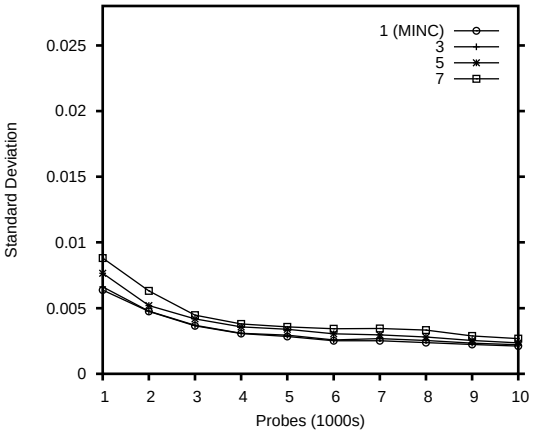
(a) Passage rate 90%: using $A(0|w)$



(b) Passage rate 90%: using $A(0|w), ...A(w-1|w)$

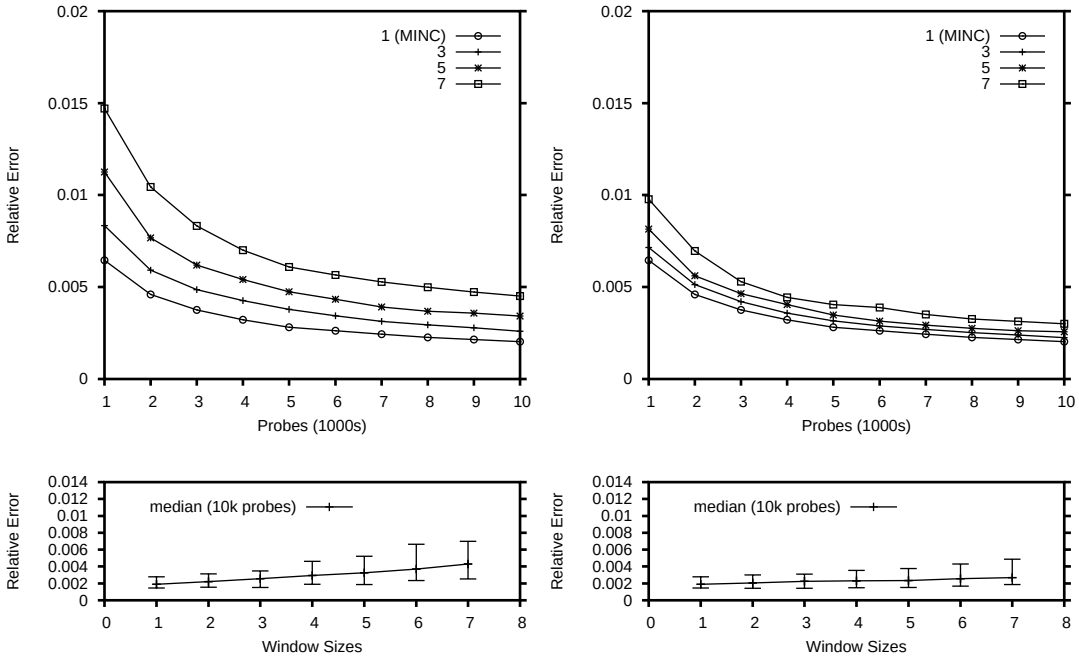


(c) Passage rate 95%: using $A(0|w)$



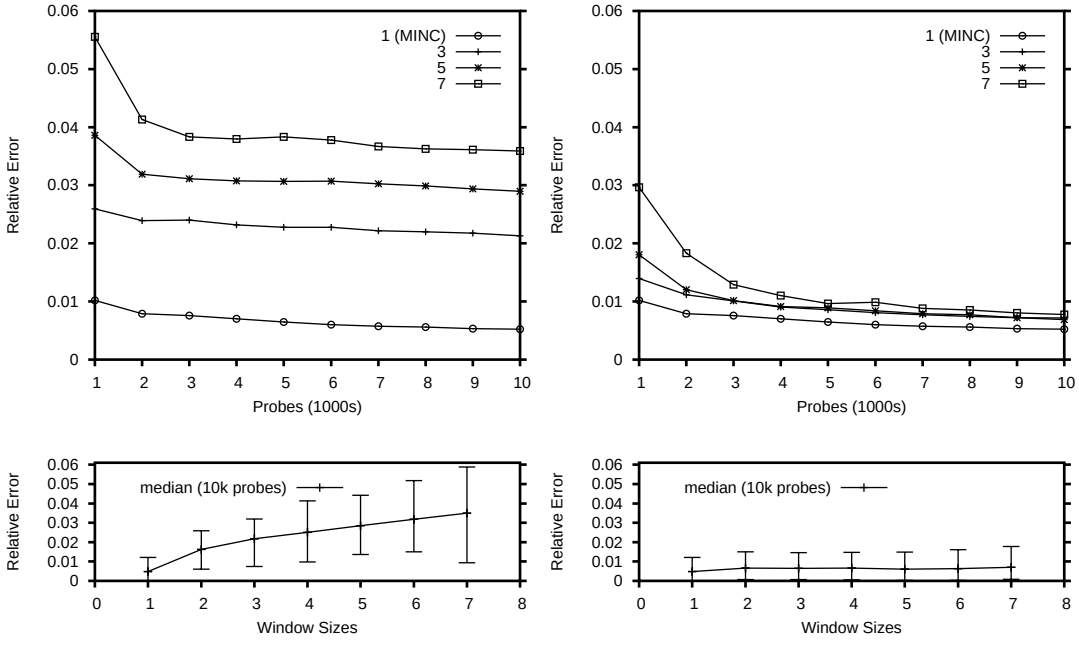
(d) Passage rate 95%: using $A(0|w), ...A(w-1|w)$

Figure 4.5: EMLE: Standard Error of estimate



(a) Model: Using $A(0|w)$

(b) Model: Using $A(0|w), ...A(w-1|w)$



(c) NS: Using $A(0|w)$

(d) NS: Using $A(0|w), ...A(w-1|w)$

Figure 4.6: EMLE: Relative Error of an internal link for 8-receiver binary tree. Results are based on 100 simulations. Passage rates of links varied from $90-99\%$ and $85-95\%$ for model-based and NS simulations respectively

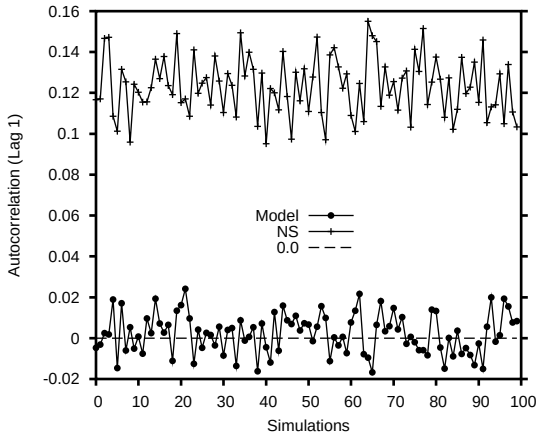
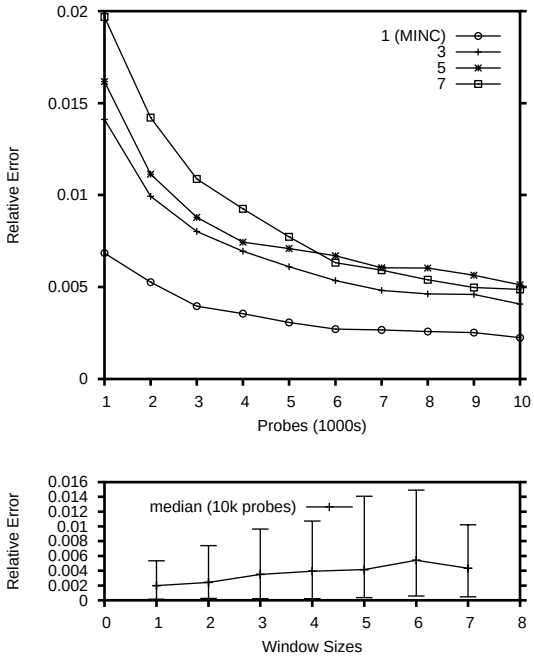


Figure 4.7: Autocorrelation (lag 1) in the loss trace of a receiver present in the 8-receiver tree, for Model-based and NS simulations.

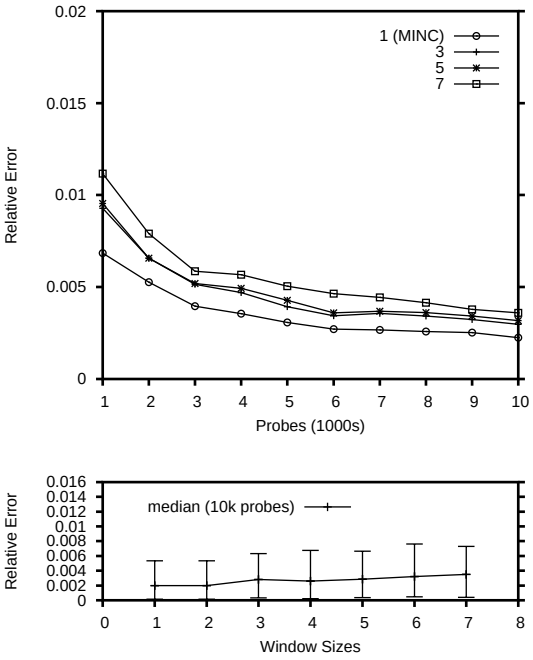
We observe that, as expected thinning by a factor of $\lceil \log_2(w+1) \rceil / w$ which uses more feedbacks yields less errors than thinning by a factor of $1/w$.

We summarize the results of experiments with thinning and EMLE as follows. In model-based simulations, for passage rates between $90\% - 99\%$ and window sizes 1 to 7, we have:
(i) Passage rates estimated by EMLE using aggregate feedbacks of precision level 0 show both lower error and lower variance than passage rates estimated by MINC loss estimator using feedbacks thinned by $1/w$.
(ii) Passage rates estimated using aggregate feedbacks of precision level $w-1$ and feedbacks thinned by $\lceil \log_2(w+1) \rceil / w$ show approximately the same error. As window sizes and loss rates of links increase, passage rates estimated using aggregate feedbacks of precision level $w-1$ show higher variance.
In NS simulations, the thinned binary feedbacks yield better estimates of passage rates than aggregate feedbacks. This is due two reasons: (i) loss rates are higher in NS simulations and (ii) NS simulations exhibit temporal correlations.

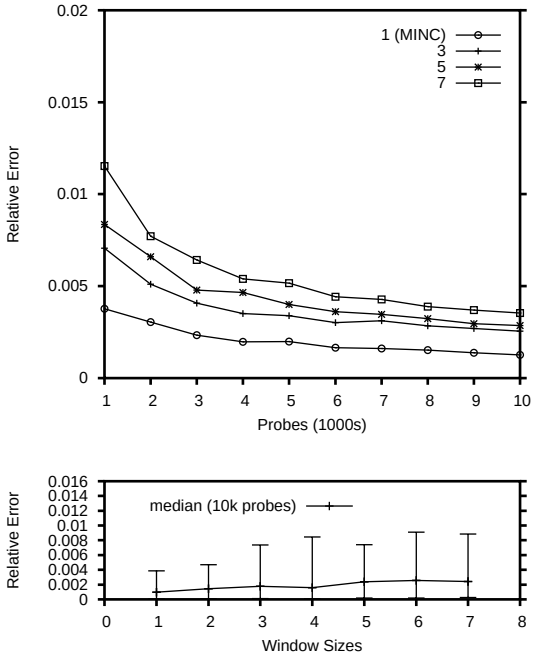
We observe that, although aggregate feedbacks of precision level 0 yield better estimates than binary feedbacks thinned by $1/w$ in model-based simulations, in the presence of temporal correlations they yield high errors. On the other hand, when EMLE uses aggregate feedbacks of precision level $w-1$, as loss rates and window sizes increase, EMLE is unable to fully reap the benefits of more feedback data. This is because loss distribution elements are estimated recursively. As loss rates and window sizes increase, errors in lower loss distribution elements effect the estimates of higher loss distribution elements. Thus, in spite of having more feedback data, the passage rates are not estimated accurately.



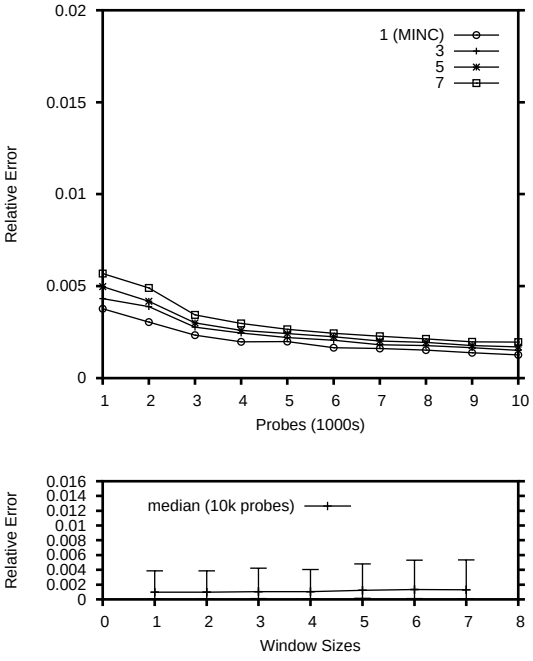
(a) Passage rate 90-95%, Thinning by $1/w$



(b) Passage rate 90-95%, Thinning by $\frac{\lceil \log_2(w+1) \rceil}{w}$

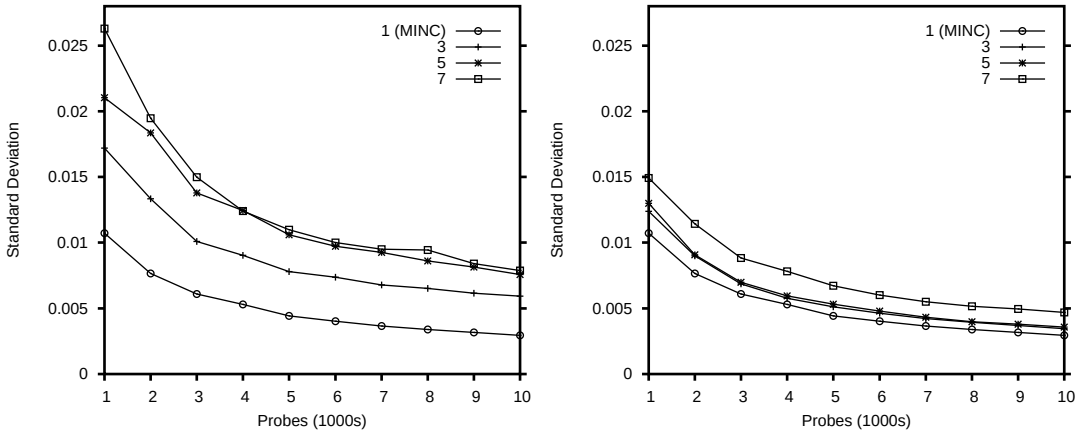


(c) Passage rate 95-99%, Thinning by $1/w$



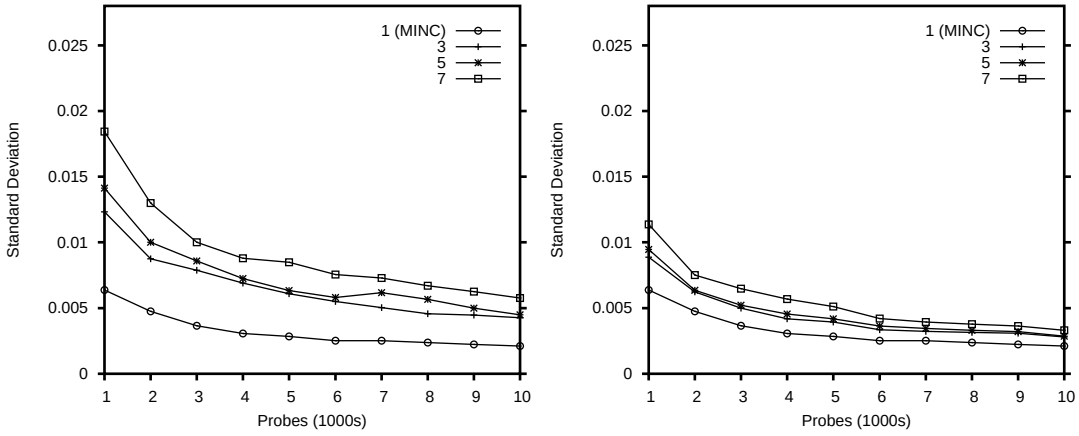
(d) Passage rate 95-99%, Thinning by $\frac{\lceil \log_2(w+1) \rceil}{w}$

Figure 4.8: Relative error of the shared path (MINC loss estimator with thinning)



(a) Passage rate 90%: Thinning by $1/w$

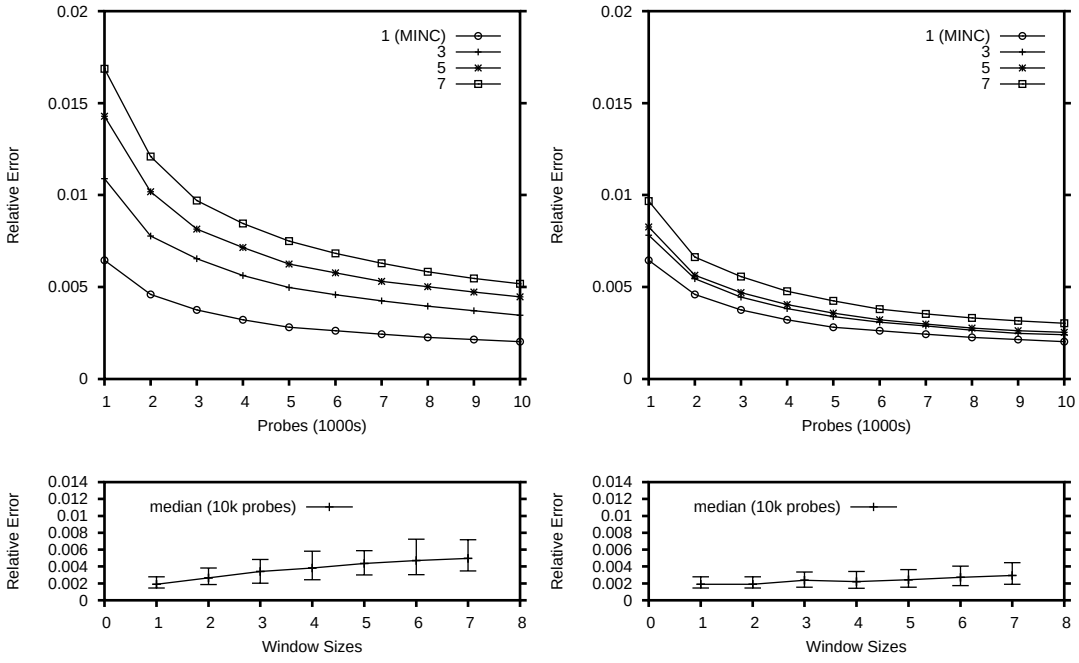
(b) Passage rate 90%: Thinning by $\frac{\lceil \log_2(w+1) \rceil}{w}$



(c) Passage rate 95%: Thinning by $1/w$

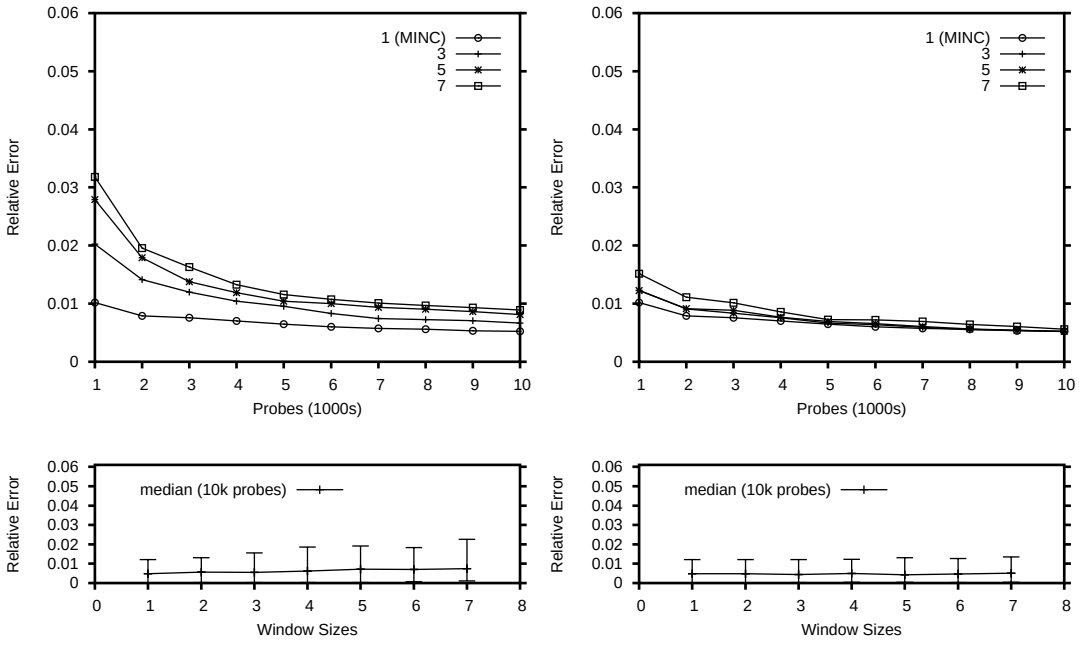
(d) Passage rate 95%: Thinning by $\frac{\lceil \log_2(w+1) \rceil}{w}$

Figure 4.9: Standard Error of estimate (MINC loss estimator with thinning)



(a) Model: Thinning by $1/w$

(b) Model: Thinning by $\frac{\lceil \log_2(w+1) \rceil}{w}$



(c) NS: Thinning by $1/w$

(d) NS: Thinning by $\frac{\lceil \log_2(w+1) \rceil}{w}$

Figure 4.10: MINC loss estimator with thinning: Relative Error of an internal link for 8-receiver binary tree. Results are based on 100 simulations. Passage rates of links varied from $90-99\%$ and $85-95\%$ for model-based and NS simulations respectively

4.9 Discussion

4.9.1 Loss of Feedbacks

While developing EMLE, we did not explicitly model the loss of feedbacks. In other words, we have assumed that the feedback loss process is MCAR (Missing Completely At Random). In MCAR, the probability of a feedback being lost does not depend on the value of any feedback which is either received or lost. In such a scenario, loss inference can be performed by using feedbacks corresponding to the subset of probes where reports from all receivers are available.

If the feedback loss process is MAR (Missing at Random), the probability of feedbacks being lost does not depend on the values of feedbacks which are lost. In this case, the above approach of using a subset of feedbacks does not guarantee a consistent loss estimator. Authors of [37] have suggested that in certain situations, if the feedback reports and the probes follow the same path, correlations could arise between the feedback loss process and the feedback values, resulting in the violation of MCAR assumption. To this end, they model the feedback loss process as MAR and develop a MINC loss estimator using Expectation Maximization Algorithm. We consider incorporating the MAR loss process in EMLE as future work.

4.9.2 Reporting Aggregate feedbacks using RTCP

When receivers of a multicast session use RTCP to report feedbacks, they collectively constrain the RTCP feedback bandwidth below a certain level of data bandwidth. This is done in a distributed manner. Hence each receiver's share of feedback bandwidth is different.

In EMLE, all receivers must use the same window size to report aggregate feedbacks. This reduces the bandwidth needed to report feedbacks for all receivers. The resulting bandwidth may still not be sufficient for some receivers since EMLE cannot support very large window sizes. Therefore, in order to report aggregate feedbacks via RTCP, receivers will need to perform thinning in conjunction with reporting aggregate feedbacks. The coordinated thinning mechanism proposed by authors of [9] can be extended to report aggregate feedbacks. When receivers use window size w, each receiver can report its aggregate feedbacks once every 2^{T} probes by constructing its aggregate feedback using the previous w probes, i.e., $2^{\mathsf{T}}-w+1,...2^{\mathsf{T}}$, where $2^{\mathsf{T}} \geq w$. The thinning exponent T can be chosen independently by each receiver. In this manner, aggregate feedbacks are thinned proportionally by each receiver. Also, the windows of probes for which different receivers report aggregate feedbacks, will overlap. (Other combinations of thinning and constructing aggregate feedbacks are also possible. Our goal here is to point that aggregate feedbacks can also be thinned proportionally by different receivers).

4.9.3 Loss Inference using standard RTCP RR reports

Using EMLE or the MINC loss estimator with thinned feedbacks requires piggy-backing of extra information in compound RTCP packets which are reported periodically by each receiver. Each compound RTCP packet contains an RTCP RR (Receiver Report) packet. An RTCP RR packet already contains aggregate loss information. Each RTCP RR packet reports two quantities - the highest packet sequence number received so far and the cumulative number of packets lost until that sequence number. Therefore, information from two successive RTCP RR packets gives us the number of probes lost within a window of probes. However, in contrast to aggregate feedbacks used by EMLE, windows of probes for which different receivers report feedbacks may not be aligned perfectly. Since RTCP RR loss reports are cumulative, loss of feedbacks only results in the increase of window size of some reports. In [34], authors have shown that it is possible to construct a moment-based estimator to perform loss inference using RTCP RR reports.

4.10 conclusions

The task of performing large scale multicast measurements to infer link loss rates requires the deployment of dedicated infrastructures and is complex. This task can be facilitated if multicast measurements can be reported on RTCP packets. However RTCP imposes constraints on the bandwidth which can be used to report measurements.

In this chapter, we developed the Extended MINC Loss Estimator (EMLE) which is able to perform loss inference using aggregate feedbacks requiring less than 1 bit per probe. An aggregate feedback constructed using window size 7 requires between 15% and 43% of probe bandwidth. EMLE uses aggregate feedbacks to estimate the loss distribution of links and computes their passage rates from their loss distribution. We evaluated the behavior of EMLE through Model-based and NS simulations. In model-based simulations, we observed that for link passage rates between $90-99\%$, the error in estimation remains below 1% for 10K probes. In NS simulations, we observed that due to the presence of temporal correlations, when EMLE estimates passage rates using first element of loss distribution, it yields higher errors. When EMLE estimates passage rates using all elements of loss distribution, the error in estimation remains below 1% for 10K probes even in NS simulations.

During the analysis of EMLE we made observations which can be used to eliminate numerical computations involved in MINC loss and delay estimators.

We compared EMLE to the case where MINC loss estimator is used with thinned feedbacks. We observed that, at high passage rates EMLE yields errors which are lower than or same as the errors yielded by MINC loss estimator with thinned feedbacks. As passage rates decrease,

EMLE yields higher estimation errors. In presence of temporal correlations, when EMLE uses aggregate feedbacks of lower precision levels, it yields higher errors in comparison to the case where MINC loss estimator is used with thinned feedbacks.

Our conclusions are that thinning of feedbacks is a good approach to reducing feedback bandwidth. In model-based simulations, although EMLE estimates more accurately at high passage rates, the results are not significantly better than the case where MINC loss estimator is used with thinned feedbacks. EMLE has two weaknesses: (i) In presence of temporal correlations, aggregate feedbacks of lower precision level which require less feedback bandwidth yield higher estimation errors. (ii) Since loss distribution elements are estimated recursively in EMLE, as link loss rates increase, errors in the estimation of lower loss distribution elements affect the estimates of higher loss distribution elements, leading to less accurate estimates of passage rates.

5

ENCODINGS OF MULTICAST TREES

5.1 Summary

In this work, we present efficient ways of encoding multicast trees. Multicast tree encodings provide a convenient way of performing stateless and explicit multicast routing in networks and overlays. Due to this reason, they help to meet goals of multicast traffic engineering.

We show the correspondence of multicast trees to theoretical tree data structures and give simple lower bounds on the number of bits needed to represent multicast trees. The tree encodings presented in this work can be utilized to represent multicast trees using both node identifiers and link indexes and are based on the well known balanced parentheses representation of tree data structures. These encodings are almost optimal in terms of space and they can be read and processed efficiently. We evaluate the length and forwarding performance of these encodings on multicast trees in generated and real topologies.

5.2 Multicast Traffic Engineering in Overlay Networks

Overlay networks are self-organizing virtual networks built ”on top” of physical networks such as Internet. Examples of overlay networks include RON (Resilient Overlay Networks) [13], OMNI (Overlay Multicast Network Infrastructure) [60, 14], Akamai, and several peer to peer networks such as Chord [61], CAN [62], Pastry [63], and Tapestry [64]. Overlay networks have two primary advantages. Firstly, overlay networks are easily deployable. Due to this reason, services such as multicast which are facing deployment problems in IP networks can be offered, possibly in a less efficient manner, in overlays. For example, in OMNI, Multicast Service Nodes(MSNs) organize themselves into an overlay to form a multicast delivery backbone. End hosts or clients avail themselves of multicast service by subscribing to MSNs. Secondly, overlay networks can also provide certain services which are not available in IP networks. For example, a pair of nodes in the Internet have access to a single default IP path. In RONs, nodes form an overlay to route packets among themselves. RON nodes monitor the quality of Internet paths among themselves and use this information to decide whether to route packets directly over the Internet or by way of other RON nodes. In this manner, RONs are able to make use of alternate paths which are not provided by IP level routing.

The goal of traffic engineering is to ”enhance the performance of a network, at both the traffic and resource levels” [12]. At the traffic level, the objective is to enhance the Qos of traffic streams which pass through the network. This includes aspects such as minimization of packet loss, minimization of delay, maximization of throughput, and routing based on certain constraints. At the resource level, the objective is to optimally utilize network resources. Here, the main objective is to avoid situations where some network resources become over utilized and congested while other feasible resources remain under utilized. Traffic engineering is mainly accomplished by controlling and optimizing the routing function in the network so that traffic can be steered through the network in the most effective manner.

The traffic engineering process broadly consists of two steps which are performed iteratively : (i) Measurement of the operational state of the Network and (ii) Use of measured state to adaptively steer the traffic on the best paths available in the network. The key capability needed to achieve the second goal is the support of routing traffic along specific or explicit paths. In the context of multicast traffic engineering, the capability to support the routing of traffic along specific or explicit multicast trees is needed. Routing all traffic along a fixed shortest path or fixed tree congests that particular path or tree. This adversely affects the performance parameters associated with the traffic when other feasible paths or trees in the network may remain under utilized.

In overlay networks, the task of traffic engineering can be performed at the application level. In commercially owned and operated overlay infrastructures, the algorithms used by overlay

nodes to perform traffic engineering can also be controlled by the overlay service providers. For example, in OMNI, based on existing traffic conditions in the overlay, the MSNs can route multicast traffic on alternate trees to balance load in the overlay. The MSNs can also construct trees which satisfy certain constraints such as trees with minimum average latency for real-time traffic [65]. In peer to peer overlays, the task of traffic engineering can be performed cooperatively by the overlay nodes which are also the end-hosts in the Internet. For instance in RONs, nodes have the potential to perform tasks such as routing based on certain constraints. In peer to peer networks, nodes can utilize the available bandwidth to perform tasks such as routing different layers of multicast video traffic on different trees.

In certain unicast traffic engineering approaches (e.g. Bananas [66]), nodes within a cluster, in addition to computing the default shortest paths among themselves, also compute additional shortest paths. These alternate paths can be explicitly chosen using fixed size packet headers. Thus traffic can be routed on additional paths to achieve goals of traffic engineering. Such an approach does not extend well to multicast. This is because the number of destinations corresponding to different multicast flows is not fixed and is typically much larger than one. Extending the above approach to multicast requires nodes to compute and store a large number of trees corresponding to all possible subsets of destination nodes.

In contrast to unicast routing, multicast routing also needs a per flow multicast state to be maintained at the routers (as discussed in the next section). Construction and maintenance of additional trees for purposes of multicast traffic engineering introduces more state overhead inside the network.

The simplest way to route traffic along explicit paths in a stateless manner is to perform source routing. By embedding multicast trees within data packets, multicast traffic can be routed on specific trees to achieve goals of traffic engineering. To apply this approach to route multicast traffic on trees of reasonably large size, efficient encodings of multicast trees are needed.

5.3 Pitfalls of Per-flow Multicast State

Current IP multicast routing protocols such as DVMRP [67] and PIM [68, 69] install a per group state in the routers. To support these protocols, a router needs to maintain a multicast forwarding table entry corresponding to every multicast group whose distribution tree passes through the router. Since the entries corresponding to different multicast groups cannot be aggregated, the forwarding table size grows with the number of active multicast groups, violating the stateless principle followed by routers. As the number of multicast groups increase, the cost and performance of routers are adversely affected. The state maintenance problem is further aggravated by the fact that even small multicast groups can introduce significant amount of

state into the network. The amount of state introduced by a multicast group in the network depends on the size of its tree. The size of the multicast tree in turn is a function of both the relative locations of the source and various receivers, and the number of receivers itself. A small multicast group consisting of receivers which are spread uniformly in the Internet may introduce more state than a large multicast group concentrated in one region of the Internet. Thus, even a large number of small multicast groups may cause a state explosion. In order to reduce the state overhead at routers, one of the approaches followed is to move the state from the network to the packet [70], i.e., to use a representation of the multicast tree within every data packet. Each router can then function in a stateless manner and perform multicast forwarding by reading and processing packet headers. To use this approach even in small to medium sized multicast groups, a compact encoding of multicast tree is required.

Due to deployment problems in Inter-domain Multicast, several Application Level Multicast protocols have been proposed [71, 72]. However, the problem of per group state maintenance remains even in application level multicast. Scribe [71] and Bayeux [72] are application level multicast protocols for Pastry [63] and Tapestry [64] overlays respectively. Pastry and tapestry are large scale DHT overlays which allow the interconnection and routing between millions of nodes in a scalable manner. To implement multicast in these overlays, both Scribe and Bayeux introduce a per group multicast state within the overlay nodes which are part of the multicast distribution tree. Due to constraints of per group state maintenance, these overlays cannot support a large number of multicast groups (small or big). State maintenance can be a serious issue in overlay nodes since they are constrained by memory and processing power (normally PCs/servers).

5.4 Other Motivations of encodings trees

In Distributed Interactive Applications such as large scale virtual environments (multiplayer games) and distributed interactive simulations, participants act as senders and receivers in several multicast groups. Minimizing the control overhead due to group dynamics is a major design consideration here [73, 74]. In these applications, nodes are clustered according to communication needs and multicast distribution trees are constructed within the clusters. As nodes join and leave, nodes send control messages to each other to repair the multicast distribution trees. On average, the number of messages is linear to the number of nodes in the tree. Since there are a large number of multicast groups with frequent joins and leaves, significant control messages flow in the network. If the multicast trees are encoded within data packets, control overhead is reduced and on-the-fly tree repair is made possible at forwarding nodes [75].

In secure group communication, group keys must be updated when nodes join or leave [76]. When nodes leave, the updated keys cannot be communicated via the old multicast tree since

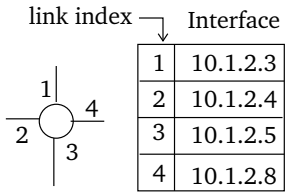


Figure 5.1: Link Index concept: A router/overlay node with its link indexes

the leaving nodes would receive the new key. Multicast tree encoding can be used to communicate the new key via an explicit multicast tree containing only those nodes which need to receive the new key [77].

The above motivations notwithstanding, very little is known about optimal ways of encoding multicast trees. Indeed, there are two proposals which encode multicast trees non optimally [75, 78]. In this work, we show efficient ways of encoding multicast trees within data packets in order to perform stateless and explicit multicast routing. We consider multicast trees which could occur at network or overlay level and show the correspondence of these trees to theoretical tree data structures. Using this correspondence, we give lower bounds on the number of bits needed to encode multicast trees and then present efficient ways of encoding them.

5.5 Requirements of Multicast Tree Encodings

Multicast tree encoding must satisfy two requirements. Firstly, the encoding must be of minimal length since it is passed in *every* multicast data packet and processed by each forwarding node (router or overlay node). Secondly, the forwarding operation must consume minimal time. Each forwarding node processes the encoding to determine the list of downstream nodes to whom the multicast packet must be forwarded. To minimize both the end-to-end packet delay and the per packet processing load on a forwarding node, the encoding must support the execution of this task efficiently.

The networks in which we consider multicast trees can be divided into two categories: (*i*) Networks which can *potentially* support forwarding based on *link Indexes*. The link index of a node with value i refers to its ith link. For example, the link index of a router refers to one of its interfaces. A router can potentially order its link interfaces and refer to each interface by its index in this ordering (figure 5.1). The link index of an overlay node refers to one of its neighbors. Using a link index, a node can be instructed to forward a packet to one of its neighbors. This operation can potentially be supported in IP networks, static overlays such as OMNI, RON, Akamai, and partially upgraded IP networks such as Bananas, where neighbors of a node do not change frequently. (*ii*) In dynamic peer to peer networks (e.g. DHTs, unstructured p2ps) both the degree and the neighbors of nodes change frequently as

nodes join and leave and hence forwarding based on link indexes cannot be supported. In these networks, node identifiers (e.g. IP addresses) of nodes need to be used to encode the multicast tree.

We show encoding of multicast trees using both Link indexes and Node identifiers. The tree encodings we propose are independent of underlying tree construction protocols and are based on *balanced parentheses representation of ordinal trees*. The rest of the chapter is organized as follows. Section 5.6 places related work in context, section 5.7 shows correspondence of multicast trees to tree data structures and section 5.8 presents various tree encodings. Section 5.9 evaluates the length these encodings on real and generated multicast trees and section 5.10 concludes.

5.6 Related work

The encoding of a multicast tree depends on whether it represents the *default* multicast tree supplied by unicast routing framework of the underlying network or whether it represents a *specific or explicit* tree. To represent a specific tree, the multicast tree must be encoded using either the node identifiers or link indexes of the entire tree. To choose the default tree, identities of receivers are sufficient. Default trees can be chosen in any network which supports unicast routing and their main application is reduction of multicast state. On the other hand, explicitly representing trees provides the option of routing multicast traffic as desired, which may be useful to create a multicast routing framework at application level, perform traffic engineering, or multicast state reduction.

In xcast [79], a multicast tree is encoded by listing the IP addresses of receivers in the multicast tree. In IP networks, a packet can be routed to a destination node given its IP address. Due to this reason, explicitly listing the IP addresses of receivers of a multicast group suffices to represent the default multicast tree that exists between the source and the receivers. Forwarding routers perform a routing table lookup for each IP address in the packet and group them based on the common outgoing interface. The multicast packet is then forwarded on the interfaces, each packet containing the corresponding subset of IP addresses. Alternatively, if the last hop routers are allowed to store multicast state, the tree can be represented by listing the IP addresses of last hop routers instead of receivers(xcast+). Xcast has two limitations. Since each IP address consumes 32 bits, xcast encoding is suitable for multicast groups containing a small number of last hop routers (6-10 last hop routers need 24-40 Bytes in IPv4). Secondly, each forwarding router needs to perform k IP lookups, where k is the number of last hop routers in its subtree. Large encodings require more IP lookups which are expensive.

The xcast model is also suited to peer to peer DHTs such as Pastry and Tapestry, where packets are routed in the overlay using node identifiers. In Pastry, a node is represented using a

128 bit identifier. By representing multicast trees using receiver identifiers, the state overhead imposed by small multicast groups can be avoided in Scribe and Bayeux. The xcast model is advantageous in DHTs since they guarantee routing in the presence of joins and leaves by intermediate peers.

In Linkcast [78], a multicast tree is encoded using the link indexes of links in the multicast tree. Each forwarding node reads *its* corresponding link indexes and forwards the multicast packet. The multicast tree which is encoded is a reverse path multicast tree constructed using join messages sent from last hop routers towards the source. Each router receiving the join message is expected to forward its identity along with the link index of the interface which received the join message. Thus the source can encode the multicast tree using link indexes. For forwarding, a pointer in the encoding points to the location of the current router in the encoding. Each router uses the pointer to read its outgoing link indexes and updates the pointer for the next hop routers. Although the number of links in the multicast tree is larger than the number of receivers, link indexes can be represented using fewer bits as compared to 32 bits for the IP address of a node. Also, expensive routing table lookups are avoided in Linkcast. We propose a new encoding for Linkcast which requires less space and two new encodings of multicast trees using link indexes.

In [75], authors propose two tree encodings to represent application level multicast trees occurring within clusters of Distributed Interactive Applications. The multicast trees are encoded using IP addresses of participating nodes. Authors propose two encodings - Per level header encoding and Preorder header encodings. In addition to IP addresses of nodes, these encodings spend significant number of additional bits. In a multicast tree of maximum degree d and n nodes, the per-level header encoding spends $\log d$ bits per node and preorder header encoding spends approximately $\log n$ bits per node. We show an encoding using only 2 additional bits per node.

Related work on application level multicast protocols is vast. These protocols include Narada [80], Yoid [81], HMTP [82], HostCast [83], Bayeux [72], Scribe [71], and Nice [84]. All these protocols, like IP multicast protocols, provide default trees to route multicast traffic among participating peers. OMNI [14, 60] and Scattercast [85] are proposals which advocate the construction of overlay multicast network infrastructures using MSNs and SCXs (scatter-cast proxies) respectively. In [65, 60], authors propose algorithms for the construction of multicast trees which satisfy certain constraints, among the MSNs. In [77], authors propose an algorithm to construct alternate backup multicast trees among overlay nodes to route multicast traffic in case of link failures. In contrast to all of the above work, the work in this chapter focusses on the construction of tree encodings to route multicast traffic on any given explicit tree in a stateless manner.

Related work on the reduction of multicast state includes REUNITE [86] and Hop-by-

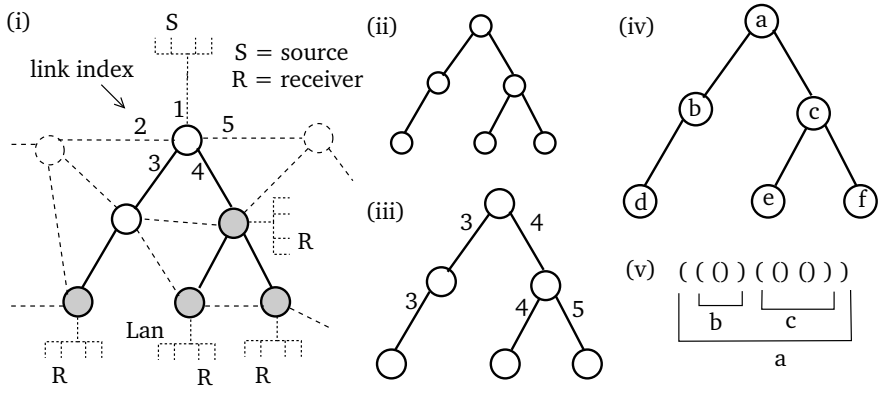


Figure 5.2: (i) Multicast Tree in a Network (ii) Ordinal tree (iii) Cardinal Tree (iv) Arbitrarily labeled tree (v) Balanced parentheses representation of ordinal tree

Hop [87] multicast routing protocols. These protocols reduce multicast state in the network by restricting the state to the branch points of the multicast tree. These protocols then route the multicast traffic via the default unicast paths that exist between branch points of the tree in the underlying network. In our work, since the multicast tree is encoded within data packets, multicast state is eliminated from the network.

The discussion of traffic engineering with MPLS (Multiprotocol Label Switching) [88] is beyond the scope of this thesis.

5.7 Multicast Trees and Tree data structures

Figure 5.2(i) shows a multicast tree occurring in an arbitrary network or overlay. The portion of the multicast tree which needs to be encoded is shown highlighted. The last hop routers, which serve one or more receivers are shown grayed. The link indexes corresponding to one of the nodes are shown. Figure 5.2(ii), (iii) and (iv) show the *Ordinal tree*, *Cardinal tree* and *Arbitrarily labeled tree* data structures corresponding to the multicast tree.

An Ordinal tree is a rooted (a unique node is distinguished as the root), unlabeled tree in which each node has an arbitrary degree. A binary tree is an ordinal tree in which the maximum degree of a node is two. The number of ordinal trees on n nodes can be bounded by $O_n = \binom{2n+1}{n}/(2n+1)$. Thus, $\lg(O_n) \approx 2n$ is an information theoretic lower bound on the number of bits needed to represent ordinal trees ($\lg$ is $\log_2$). Every ordinal tree of n nodes can be represented using $2n$ balanced parentheses. Figure 5.2(v) shows the balanced parentheses representation of the ordinal tree in figure 5.2(ii). This representation is obtained by a preorder (depth first) traversal of the tree, outputting a "(" while visiting a node for the first time and a matching ")" while visiting the node after visiting its subtree (The correspondence of some parentheses to nodes is shown). By representing a "(" by 1 and a ")" by 0, balanced parentheses

can be represented using $2n$ bits.

Figure 5.2(iii) shows the cardinal tree which results when a multicast tree is represented using link indexes. A Cardinal tree of degree d is an unlabeled tree in which each node has d positions for an edge to a child (if a child is at the ith position, the corresponding link gets the label i). There are $C_n^d = \binom{dn+1}{n}/(dn+1)$ cardinal trees of degree d on n nodes. $\lg(C_n^d) \approx (\lg d + \lg e)n$ is an information theoretic lower bound on the number of bits needed to encode these trees. A multicast tree in which the maximum link index is d is a cardinal tree of degree d. Figure 5.2(iii) is a cardinal tree of degree 5. Thus, $(\lg d + \lg e)n$ is a lower bound on the number of bits needed to represent a multicast tree using link indexes.

Figure 5.2(iv) shows the Arbitrarily labeled tree which results when a multicast tree is represented using node identifiers. Node Identifiers a,b, etc are essentially IP addresses. A *labeled* tree of n nodes is an ordinal tree in which each node is labeled from 1 to n. An *arbitrarily labeled tree* is an ordinal tree in which each node has a *unique* arbitrary label. If each node is represented using a k bit label, then there are $A_n^k = \binom{2^k}{n}\, n!\, O_n$ such trees on n nodes. Thus, $\lg(A_n^k) \approx kn + 2n$ is a lower bound on the number of bits needed to represent a multicast tree using node identifiers.

5.8 Multicast Tree Representations using Link Indexes

For ease of explanation, we use the following notation. All nodes in the multicast tree have one incoming link (except the root) and zero or more outgoing links. A *terminal or leaf* node has zero outgoing links, a *relay* node has one outgoing link and a *branch* node has more than one outgoing links. In figure 5.3(i), a, e, h, and d are branch nodes, j, k, m, c, f, and g are leaf nodes and b, i, and l are relay nodes. The links of a multicast tree can be divided into branch and relay links. All outgoing links of a branch node are branch links and all outgoing links of a relay node are relay links. For example, ac is a branch link and lm is a relay link. Let l denote the number of links and n denote the number of nodes, $n = l + 1$. Let $\bar{b}$ denote the number of branch links and $\bar{r}$ the number of relay links, $l = \bar{b} + \bar{r}$. Let b denote the number of branch nodes, r the number of relay nodes, and t the number of leaf nodes, $n = b + r + t$. For tree in figure 5.3(i), the preorder node traversal gives $a, b, e, h, j, k, i, l, m, c, d, f, g$ and the preorder link traversal gives $ab, be, eh, hj, hk, ei, il, lm, ac, ad, df, dg$.

5.8.1 Improvements to Linkcast Encoding

Linkcast consists of two encodings - SBM and DBM. These encodings consist of a series of elements. In SBM, each element is either a link index or a pointer. A pointer is used to point to the representation of the child node in the encoding. In DBM, each element is either a link

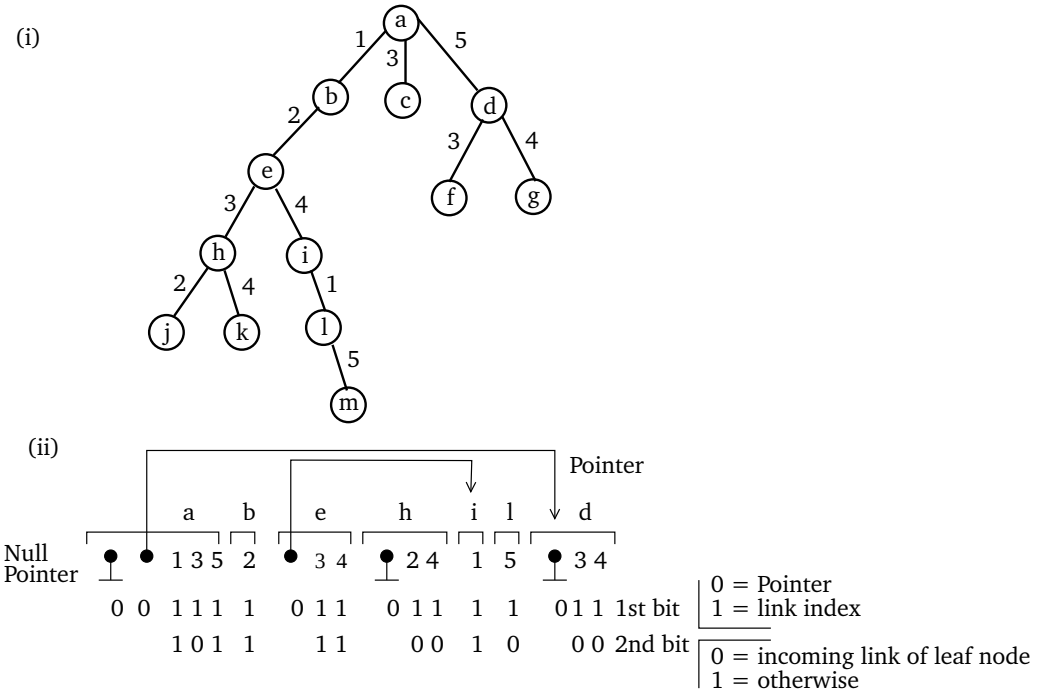


Figure 5.3: (i) Multicast Tree (ii) Link+ encoding

index, pointer or a node demarker. SBM requires $\bar{b}$ pointers. DBM reduces this to $\bar{b}-b$ pointers, but at the expense of introducing $l+1$ node demarkers. Each node demarker requires at least two bits to be differentiated from both link indexes and pointers, consuming $2(l+1)$ bits of space. Utilizing the concept of pointers, we now present a simplified encoding which is shorter than both SBM and DBM and refer to this encoding as Link+. Instead of using node demarkers, our approach is to distinguish between links that terminate at routers which perform forwarding and routers which do not, using one bit per link. One bit is used to distinguish between pointers and link indexes.

To construct Link+ encoding, nodes are visited in preorder. When a node is visited, a string of pointers and outgoing link indexes is outputted. If a node has d outgoing links, then $d-1$ pointers followed by d link indexes are outputted. The $d-1$ pointers point to the output strings of 2nd to dth child nodes. The output string of 1st child node occurs immediately after the output string of current node. Figure 5.3(ii) shows the Link+ encoding for the tree in figure 5.3(i). For example, root node a has three outgoing links, thus 2 pointers followed by three link indexes are outputted. The output string of the first child node b occurs after the output string of a. The 1st pointer points to the output string of 2nd child node c (null). The 2nd pointer points to the output string of 3rd child node d. The first bit in each element is used to distinguish a pointer (0) from a link index (1). The second bit in each link index is used to

```
Forward( Enc, ptr )
if ptr == 0 exit;
if ( Enc[ptr].firstbit() == 0)      // pointer => branch node
   i = ptr;
   child = 0;
   while (Enc[i++] is pointer) child++;  // count pointers
   for (j=1; j <= child; j++)       // read outgoing links
       OLink[j] = Enc[i + j - 1];
       OPtr[j] = Enc[ptr + j - 1];  // pointer passed to child
   child++;
   OLink[child] = Enc[i+child-1];  // last child
   OPtr[child] = i + child;
else  // relay or terminal
   if (Enc[ptr] is terminal)
      child = 0;
   else
      child = 1;
      OLink[child] = Enc[i];
      OPtr[child] = i+1;
```

Figure 5.4: Forwarding Algorithm for Link+

distinguish links which terminate on leaf nodes (0) from those which do not (1).

Routing In Link+, each node receives the pointer to its position in the encoding (as linkcast). The root node receives a pointer to the first element in the encoding. Each forwarding node determines the (*i*) outgoing links to forward the packet (*ii*) positions of child nodes in the encoding in turn to be forwarded to the children. If the position of the current node starts with a pointer, then it is a branch node, else it is a relay node. If a branch node finds d pointers, it reads the subsequent $d+1$ elements to determine its outgoing links. A relay node receives the position of its outgoing link and its child is the next element. If the second bit of an outgoing link is 0, a null pointer is forwarded and thus a leaf node receives a null pointer. The forwarding operation at a node takes time proportional to its out degree. The detailed algorithm is shown in figure 5.4.

In total, the encoding length of Link+ is $(S_l+2)l+(S_p+1)(\bar{b}-b)$ bits. S_l is the number of bits needed to represent the maximum link index in the tree. S_p is the number of bits needed to represent a pointer, $S_p = \lceil \lg(l+\bar{b}-b) \rceil$.

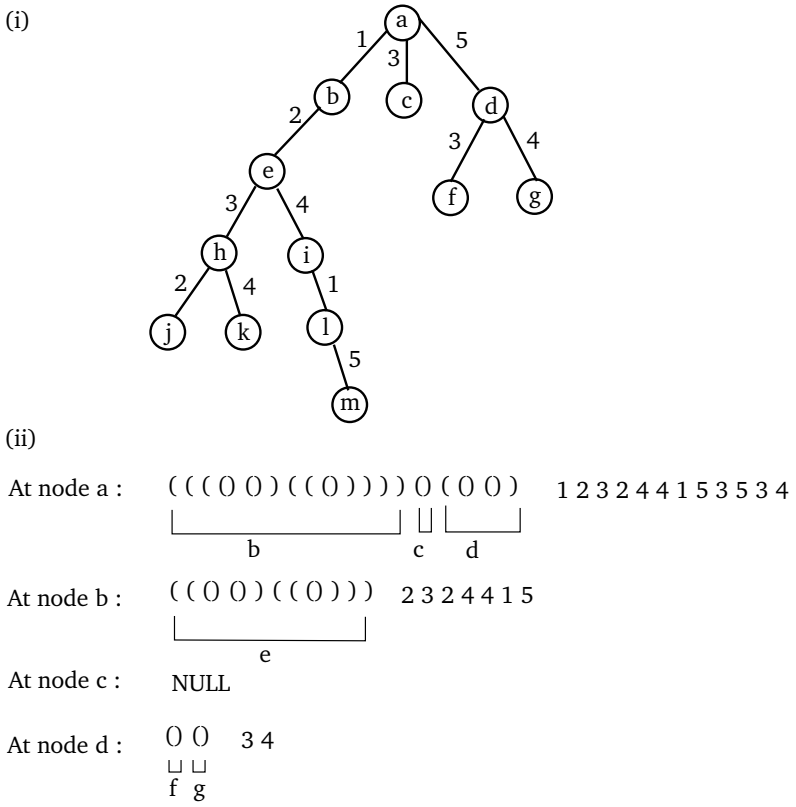


Figure 5.5: (i) Multicast Tree (ii) Link* encoding

5.8.2 Encoding based on Balanced Parentheses

We now present a link index encoding based on balanced parentheses representation. We refer to this encoding as Link*. Link* encoding consists of two parts (*i*) balanced parentheses representation of the tree (*ii*) preorder list of link indexes. The first encoding in figure 5.5(ii) shows the encoding for the tree in figure 5.5(i). The balanced parentheses representation is similar to the balanced parentheses representation of ordinal trees, except that each parenthesis now corresponds to the incoming link of a node, instead of the node itself. The encoding is constructed by a preorder traversal of the tree. When a link is visited for the first time, its link index and a "(" are outputted. When a link is visited after visiting all links in its subtree, a ")" is outputted. This results in $2l$ parentheses and l link indexes. In total, Link* requires $(S_l + 2)l$ bits. S_l as before denotes the number of bits needed to represent the maximum link index.

Routing In Link*, instead of passing a pointer to the position of a node in the encoding, the encoding itself is changed after each forwarding operation. Each node receives only the encoding of its subtree. Figure 5.5(ii) shows the encoding received by the root node a and the encodings it passes to child nodes b, c, and d. A forwarding node reads the balanced

parentheses once to determine the outgoing links and the encodings of subtrees to be passed to the child nodes. The ith open parenthesis "(" corresponds to the link index at the ith position in the list of link indexes within the encoding. For example, in figure 5.5(ii), root node a reads its balanced parenthesis and determines that it has 3 children. The 1st "(" corresponds to child node b, the 9th "(" corresponds to c and the 10th "(" corresponds to d. Thus a's output links are at positions 1, 9, and 10 in the list of link indexes, i.e., link indexes 1, 3, and 5 respectively. The detailed forwarding algorithm is shown in figure 5.6. The forwarding operation of a node takes time proportional to reading the balanced parenthesis representation of its subtree. (This encoding is similar to encoding of Munro, et al [89]. However, they consider efficient bit representations of balanced parentheses for very large trees, finding the parent of a node, size of subtrees, etc which are not needed for multicast trees. We use only the bare minimal representation of cardinal trees).

5.8.3 Improvements to balanced parentheses representation

Several studies [90, 91, 92] have investigated the type of multicast trees which occur in the Internet. Trees occurring in the Internet are constrained by the underlying structure of the Internet. These trees have a large number of relay nodes compared to branch nodes. In Link*, other than the mandatory space of lS_l bits spent to represent the link indexes, 2 bits are spent for each link (for balanced parentheses). We now present a new encoding Link** which reduces the number of bits on those links which terminate on relay nodes and increases the number of bits on those links which terminate on branch nodes. In Link**, 3 bits are spent per incoming link of a branch node or leaf node, and only 1 bit is spent per incoming link of a relay node. In total, Link** requires $(S_l + 2)l + b + t - r$ bits. If $r > b + t$, the encoding spends less than 2 bits per link. In section 5, we shall see that for several trees in the Internet, the number of relay nodes is larger than the sum of branch and leaf nodes.

Link** encoding consists of three parts (*i*) relay bit (*ii*) balanced parentheses (*iii*) preorder list of link indexes. The encoding is similar to Link*. But it is obtained by encoding a virtual tree in which a path of consecutive relay nodes is treated as a single virtual link whose label is the concatenated list of its individual link indexes. Thus, the balanced parentheses represent only branch and leaf nodes. The first encoding in figure 5.7(ii) shows the encoding for tree in figure 5.7(i). For example, the path em is treated as a virtual link with label "415", and the path ae is treated as a virtual link with label "12". To determine the start and end of a virtual link, the first bit in each link index is used to distinguish links which terminate on intermediate relay nodes (1) from those which terminate on branch or leaf nodes (0). The relay bit is used for routing.

Routing The forwarding operation is similar to Link*. Figure 5.7(ii) shows the encoding received by root node a and the encodings it passes to child nodes b, c, and d. The ith open

```
Forward (BPStr, LIStr, n)
if BPStr == NULL exit;
bal = 0;
for i=1 to 2n do    // scan the balanced parenthesis
  if (BPStr[i] == "(")
     open++;
     if bal == 0
        child++;
        OLink[child] = LIStr.link(open);
     else
        LI[child].append(LIStr.link(open)); //encoding passed to children
        BP[child].append("(");
     bal++;
  else
     bal--;
     if bal != 0
        BP[child].append(")");
```

Figure 5.6: Forwarding Algorithm for Link*

parenthesis "(" corresponds to the ith virtual link in the encoding. For example, in the encoding received by a, the 1st "(" corresponds to the 1st virtual link ae and the 5th "(" corresponds to the 5th virtual link em. During forwarding, a node reads the relay bit to check whether it is a relay node or not. The relay bit is 1 when the encoding is received by a relay node and 0 otherwise. The relay bit received by a node is simply the first bit taken from the link index of its incoming link. For example, when a performs forwarding, it sets the relay bit of child node b equal to first bit of link index of link ab. In Link**, only branch nodes read the balanced parentheses and change it. The relay nodes read only the first link index to perform forwarding. The time complexity of forwarding is constant at relay nodes. The time complexity of forwarding at a branch node is proportional to reading the balanced parenthesis of its virtual subtree. The detailed forwarding algorithm is shown in figure 5.8.

5.8.4 Representing trees using Node Identifiers

Both encodings Link* and Link** can be used *as is* by replacing link indexes of links by node identifiers of nodes which terminate on those links. Figure 5.9(i) shows the encoding Link* using node identifiers for tree figure 5.7(i), as received by root node a. The forwarding algorithm remains the same, except that each forwarding node routes the packet to the corresponding

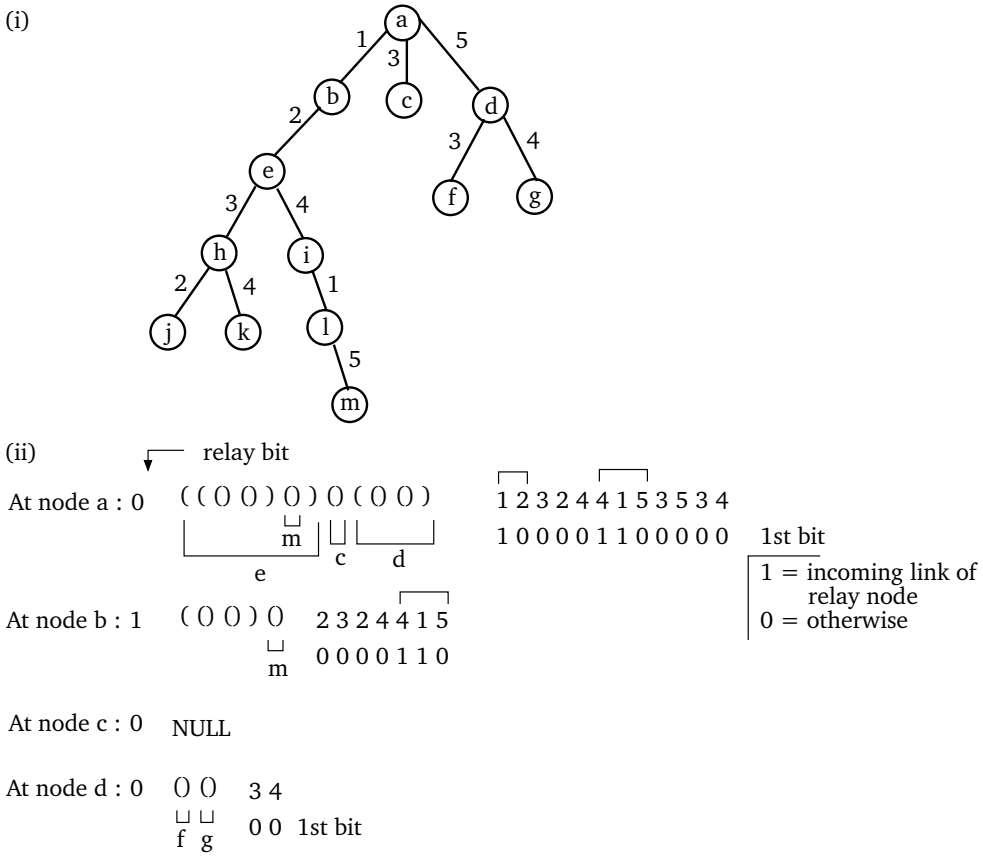


Figure 5.7: (i) Multicast Tree (ii) Link** encoding

next hop nodes instead of sending it on specific outgoing links. Representing trees with node identifiers using Link* or Link** encodings is suited to application level multicast.

5.8.5 Representing trees using both Node Identifiers and Link Indexes

In networks which support unicast routing based on node identifiers, when the default multicast tree is represented using link indexes, it may be advantageous to replace certain long paths of consecutive relay links in the tree with the IP address of the last branch or leaf node in the path. For this reason, we need a representation of tree using both node identifiers and link indexes. For example, if each link index is represented using 6 bits, a path of 10 relay links consumes 60 bits. In Link*, since 2 bits are spent per link, 80 bits are spent in total for this path. Since the default tree is being represented, the path of relay links can be replaced by the 32 bit IP address of the next branch or leaf node.

Both Link* and Link** can be modified to represent certain paths using IP addresses of the terminating node in the path. In Link*, one extra bit is needed per element to distinguish be-

```
Forward (BPStr, LIStr, n, relay)
if BPStr == NULL exit;
if (relay == 1)
    child = 1;
    OLink[child] = LIStr.link(1);
    Relay[child] = OLink[child].first_bit();
    BP[child] = BPStr;
    LI[child] = LIStr.without_firstlink();
    return;
bal = 0;
for i=1 to 2n do /* scan the balanced parenthesis*/
  if (BPStr[i] == "(")
     open++;
     if bal == 0
        child++;
        OLink[child] = LIStr.Vlink(open).link(1);
        LI[child].append(LIStr.Vlink(open).without_firstlink());
        Relay[child] = OLink[child].firstbit();
     else
        LI[child].append(LIStr.Vlink(open));
        BP[child].append("(");
     bal++;
  else
     bal--;
     if bal != 0
        BP[child].append(")");
```

Figure 5.8: Forwarding Algorithm for Link**

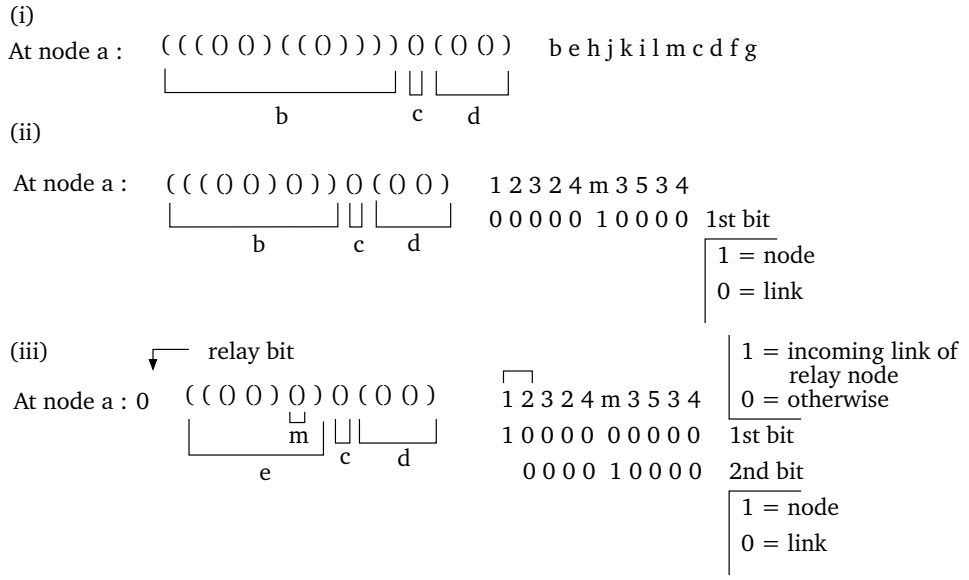


Figure 5.9: (i) Link* encoding using nodes (ii) LN* encoding (iii) LN** encoding

tween a node identifier and link Index. In Link**, one extra bit is needed per branch or leaf node to do the same. We call the corresponding new encodings as LN* and LN** respectively. Figure 5.9(ii) and (iii) show the encodings of the tree figure 5.7(i) using LN* and LN** encoding. The path ae is encoded the usual way using link indexes and the path em is replaced by the node identifier m. In LN* and LN**, it is assumed that intermediate nodes can perform forwarding using both node identifiers and link indexes. If intermediate nodes are IP routers, they need to perform tunneling to route the IP packet to the next branch or leaf node.

5.9 Simulations

In this section, we evaluate the link based encodings when they are used to represent shortest path multicast trees. We conducted tests using various generated and real topologies of the Internet. Here, we present the results for three representative topologies (Table 5.1). `ts1000` is the transit-stub topology generated by GT-ITM [93], `scan` topology is a partial map of the Internet collected by SCAN project [94], and `att` is the router level ISP topology of AT&T collected by Rocketfuel [95]. For each of these topologies, we chose random routers as source and last hop routers and constructed shortest path multicast trees from source to last hop routers. To compute shortest paths, for `scan` and `att`, hop count metric was used and for `ts1000`, the generated weights were used. Figure 5.10 (a),(c), and (e) show the properties of multicast trees for the three topologies (each point is the average of 1000 simulations).

As observed, in all topologies the number of relay nodes is significantly larger than the

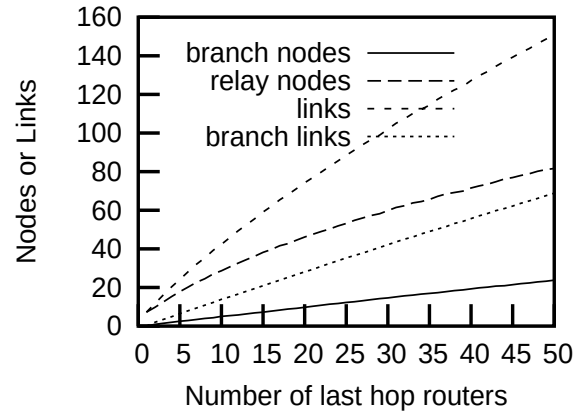
topology	Routers	type
`ts1000`	1040	Generated(GT-ITM)
`scan`	284,805	Real (Internet)
`att`	731	Real (AT&T ISP)

Table 5.1: Topologies used for simulation

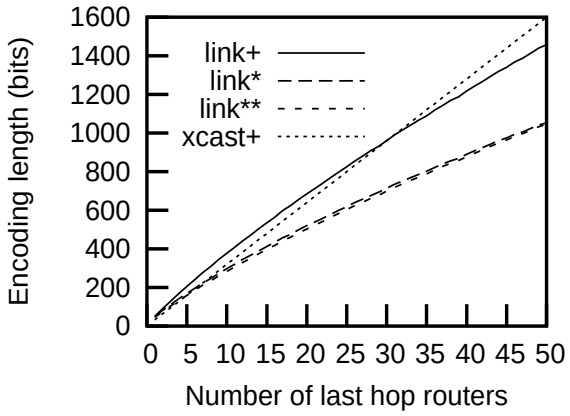
number of branch nodes. However, the total number of branch links is significant and grows linearly with the number of last hop routers. Figure 5.10(b), (d), and (f) show the encoding lengths of Link+, Link*, Link**, and xcast+. The xcast+ encoding length is essentially 32 times the number of last hop routers. For Link encodings, the link index was represented using 5 bits (routers in the Internet rarely have degree larger than 32). The pointer in Link+ was represented using 8 bits. As seen, the Link+ encoding takes more space than both Link* and Link** to represent pointers corresponding to the branch links (linkcast encodings take at least $l + 2$ more bits than Link+, for l links). For inter-domain topologies `ts1000` and `scan`, Link** encoding takes less space than Link* due to the presence of a large number of relay routers. For intra-domain `att` topology, as the number of last hop routers increase (> 25), Link* takes less space than Link** since the proportion of relay routers reduces compared to branch and last hop routers.

The forwarding performance of link based encodings is better than xcast+ since expensive routing table lookups are avoided. The forwarding performance of encodings depends on two factors: (a) Length of encodings read and processed by each node (b) Time taken by the forwarding algorithm to determine the outgoing links to forward the multicast packets. The performance of Link+ forwarding algorithm is better than Link* and Link** since the positions of outgoing links during forwarding operation are obtained directly by reading the pointers. In Link* and Link** the positions of outgoing links are obtained by reading the balanced parentheses. Thus the forwarding algorithm of Link* and Link** takes some extra time compared to Link+. At each forwarding node, this extra time is proportional to the time to read the balanced parentheses representation of the node's subtree. In figure 5.11, the average number of parentheses read per node as the multicast packet is forwarded from the source to a leaf node is shown in case of Link* and Link** encodings. When the tree is encoded using Link**, less parentheses are read since balanced parentheses are used only for branch and leaf nodes. It must be noted here that since each balanced parenthesis is represented using a single bit, the balanced parentheses representation read by each forwarding node is small in length and can be loaded into processor registers and processed at high speeds.

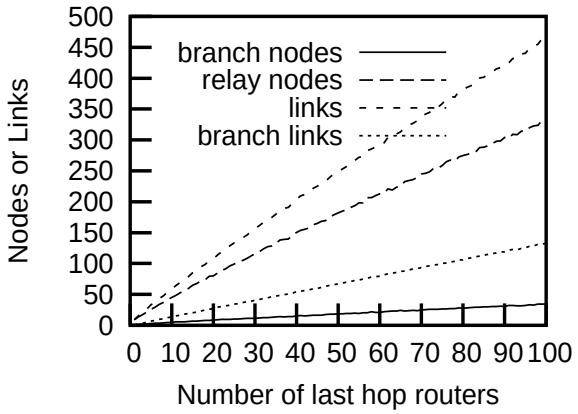
The space and forwarding performance corresponding to node based encodings is analogous to link based encodings, but of larger scale since more bits are spent for a node as compared to



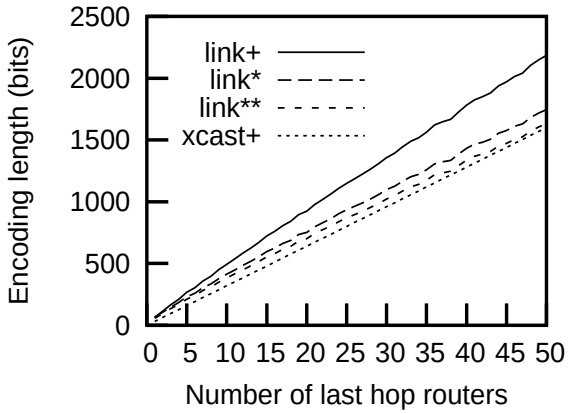
(a) `ts1000`



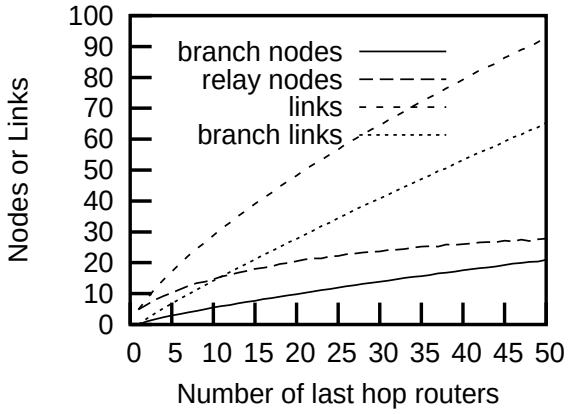
(b) `ts1000`



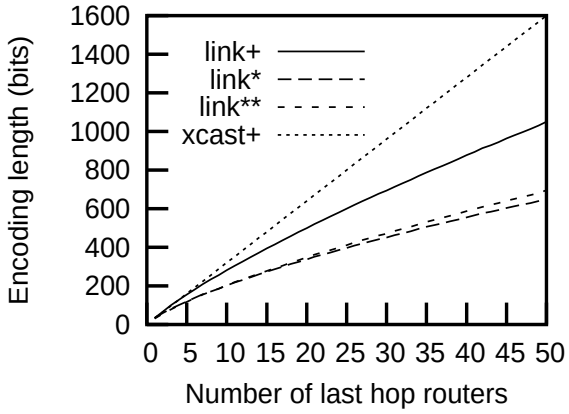
(c) `scan`



(d) `scan`



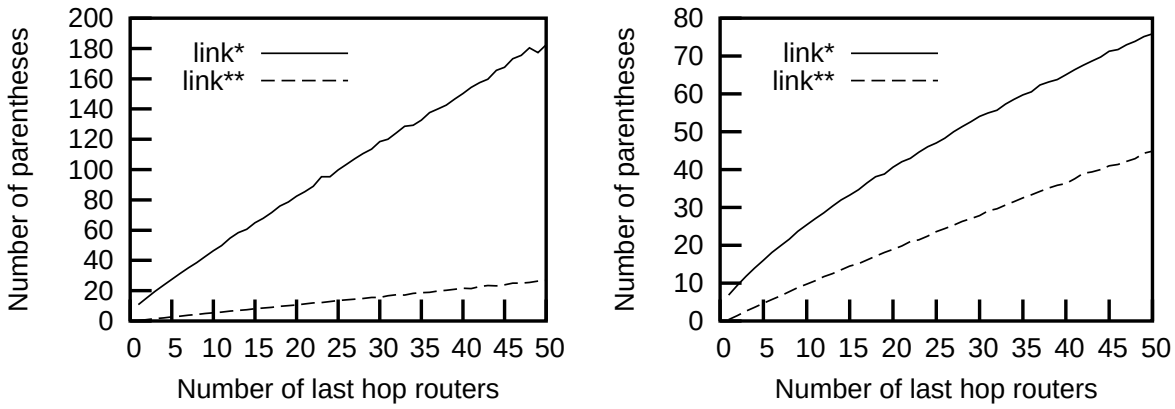
(e) `att`



(f) `att`

Figure 5.10: Properties of multicast trees and their encoding lengths in various topologies

that for a link.



(a) `scan` (b) `att`

Figure 5.11: Average number of parentheses read per node for Link* and Link** encodings

5.10 Conclusions

In this work, we presented efficient ways of encoding specific or explicit multicast trees using link indexes and node identifiers. We presented an improved encoding Link+ and two new encodings Link* and Link** which consume space close to the minimum number of bits needed to represent multicast trees. These encodings can be used to represent multicast trees using either link indexes or node identifiers. We evaluated the space and forwarding performance of these encodings using shortest path multicast trees in real and generated topologies. These encodings can be used for various applications such as multicast state reduction, multicast traffic engineering and application level multicast, and provide a feasible way of representing trees of small and medium sized multicast groups within data packets.

6

CONCLUSIONS

In this thesis, we have addressed problems related to two topics: (i) Inference of Congestion in the Internet and (ii) Multicast Traffic Engineering in Overlay Networks.

Correct inference of congestion is crucial to the functioning of transport protocols in the Internet. With the growth of wireless links in the Internet, transport protocols must differentiate between wireless and congestion losses in order to infer the right level of congestion on end-to-end paths. In chapter 2, we presented AED, an explicit loss differentiation scheme which allows unreliable transport protocols to accurately differentiate between congestion and wireless losses. In AED, agents are deployed at the boundaries of wireless links. These agents mark the cause of losses corresponding to a window of packets, within packets which are not lost, with low overhead. AED can be used in conjunction with protocols such as TFRC. We showed by simulations and modeling how the window size utilized by agents affects the throughput of the TFRC protocol. We showed that even if agents maintain small window sizes, AED performs significantly better than previously proposed end-to-end loss differentiation schemes.

Inferring the level of congestion on individual network links is important in order to monitor and manage networks. This information is useful to network operators and service providers in order to make decisions concerning routing of traffic within their network or the upgrade of certain parts of their network. In chapters 3 and 4, we addressed two problems related to a method of performing network tomography called MINC. MINC infers loss rates, i.e. the level of congestion, on individual network links from end-to-end multicast measurements.

In order to reliably use any information that is inferred from external end-to-end measurements, it is essential to verify the integrity of the measured data. Due to the presence of buggy or malicious multicast receivers, the binary feedbacks which are used by MINC to infer loss rates of network links may be incorrect. Incorrect feedbacks lead to a faulty inference of loss

rates wherein good network links may get classified as having high loss rates and bad network links may get classified as having low loss rates. In chapter 3, we presented a statistical verification algorithm called *ICheck* which can verify the integrity of binary feedbacks collected from multicast receivers. In conformance with the end-to-end nature of network tomography, *ICheck* does not need the knowledge of the multicast tree topology and works even in the presence of colluding receivers. *ICheck* helps to determine if a given set of binary feedbacks collected from multicast receivers would result in a trustworthy inference of link loss rates or not.

ICheck checks the feedbacks of groups of three receivers and reports the number of inconsistencies. Future work in this direction will involve answering the following questions: (i) Can we estimate a global quantity which represents the likelihood that the given feedback data is incorrect (ii) Given the knowledge of multicast tree topology, can the information from different three-receiver tests be aggregated to identify misbehaving receivers (iii) Can a similar test be designed to verify standard RTCP RR feedbacks collected from multicast receivers.

The task of performing large scale multicast measurements to infer link loss rates requires the deployment of dedicated infrastructures and is complex. This can be simplified if receivers of multicast sessions can perform measurements based on their data packets and report these measurements via RTCP. However, since RTCP feedback bandwidth must not exceed 5% of data bandwidth, we need ways of performing MINC loss inference using less feedback bandwidth. In chapter 4, we designed an extended MINC loss estimator(EMLE) which can infer loss rates of links using aggregate feedbacks. Aggregate feedbacks require less feedback bandwidth and we showed that EMLE performs loss inference using these feedbacks without significant loss of accuracy. Another approach to reducing feedback bandwidth is thinning, i.e. to report a sampled set of feedbacks. The thinned feedbacks can then be given to the MINC loss estimator for loss inference. We compared the above two approaches using model-based and NS simulations and concluded that thinning is a better approach to reducing feedback bandwidth. Our work on EMLE also revealed that the MINC loss and delay estimators can be modified to eliminate numerical computations.

Future work on the above problem will involve performing experiments to compare EMLE and the approach of using thinning along with MINC loss estimator, to the recently proposed moment-based estimator [34]. This will involve performing experiments similar to those performed by authors of [9] using an RTP/RTCP emulator to capture realistic network settings. Also, it remains to be determined if EMLE can be extended in a manner similar to MINC loss estimator to incorporate MAR feedbacks loss process.

The goal of multicast traffic engineering is to steer multicast traffic on specific or explicit trees to balance the load in the network and meet certain traffic constraints. Maintaining a single multicast tree to route all the multicast traffic congests paths within the tree while other network paths may remain under-utilized. The task of multicast traffic engineering is feasible in

overlay networks since overlay nodes are generally servers and PCs which can be programmed to perform intelligent and adaptive functions. Performing the same tasks within routers requires their upgrade. In order to perform multicast traffic engineering, we need methods to steer multicast traffic on specific or explicit trees. In chapter 5, we presented efficient ways of encoding multicast trees within data packets. These encodings can be used in overlay networks to route multicast traffic on explicit trees in a stateless manner. These encodings are almost optimal in terms of space and can be read and processed efficiently. Using experiments with multicast trees in real and generated network topologies, we showed that these encodings can be used to feasibly represent trees of small and medium sized multicast groups within data packets. We also showed the correspondence of multicast trees to theoretical tree data structures and obtained simple lower bounds on the number of bits needed to represent multicast trees.

A

Présentation des Travaux de Thèse en Français

Cette thèse s'intéresse à des problèmes liés à (i) l'estimation de la congestion sur Internet et (ii) le trafic multipoint dans les réseaux superposés (*overlays*). Dans la première partie de la thèse nous proposons des méthodes permettant d'améliorer l'estimation de la congestion sur les chemins de bout-en-bout ainsi que sur les liens individuels dans l'Internet. Dans la deuxième partie de la thèse nous proposons un mécanisme pour l'ingénierie de trafic multipoint dans les réseaux superposés.

A.1 Congestion sur Internet

Internet est un réseau à commutation de paquets qui offre un modèle de service de classe unique de type *best effort*. Dans ce réseau, les paquets introduits par différentes machines traversent des liens de communication, attendent brièvement dans les files d'attente des routeurs, et atteignent leurs destinations respectives. Les routeurs qui transmettent les paquets ne fournissent aucune priorité aux paquets et les transmettent généralement dans leur ordre d'arrivée. L'une des caractéristiques majeurs de ces réseaux est le phénomèene de *congestion*. En général

la congestion apparaît lorsque la demande pour une ressource du réseau est plus grande que sa capacité [1]. Si par exemple, dans un intervalle de temps donné, le nombre de paquets devant traverser un lien dépasse la capacité de ce dernier, la congestion apparaît sur ce lien. La conséquence de ce phénomène est que les files d'attente dans les mémoires tampon (*buffers*) des routeurs s'allongent, jusqu'à ce que des paquets arrivent à des tampons pleins et soient simplement perdus par les routeurs. Mis à part la suppression des paquets qu'ils ne peuvent pas prendre en charge, les routeurs ne font rien de particulier pour informer qui que ce soit de la congestion (Notons ici que le mécanisme *ECN* (*Explicit Congestion Notification*) [2], qui est une exception à ce principe, reste faiblement déployé). Lorsque la congestion augmente dans le réseau au-delà d'un certain seuil, la performance du réseau se dégrade. Afin d'utiliser le réseau de manière coopérative au meilleur de sa capacité à transférer des données, il est du devoir des machines, ou plus exactement des protocoles de transport utilisés par les terminaux, de détecter la congestion et de la maintenir sous contrôle.

A.1.1 Estimation de la congestion

Le symptôme de congestion le plus simple à mesurer est la perte de paquet. Si les paquets traversant un chemin dans le réseau sont perdus fréquemment, il est probable que le chemin en question soit congestionné. L'autre symptôme de congestion sur un chemin est l'augmentation de la latence (délai). Si les paquets qui traversent un chemin dans le réseau subissent un délai croissant, il est probable qu'ils passent plus de temps dans les files d'attente en raison de la congestion. Cependant, mesurer le délai et en déduire une estimation de la congestion est une tâche complexe. Dans cette thèse, nous nous intéressons à l'estimation de la congestion à partir du taux de perte de paquets.

Les protocoles de transport comme *TCP* qui transmettent des paquets sur Internet estiment le niveau de congestion de bout-en-bout, et, à partir de cette estimation, adaptent le débit d'émission des paquets dans le réseau. Si ces protocoles n'estiment pas correctement le niveau de congestion, ils risquent de nuire au flux qu'ils transportent ainsi qu'aux autres flux dans le réseau. Un protocole qui estime par erreur trop de congestion n'utilise pas la bande passante

disponible, tandis qu'un protocole qui sous-estime la congestion risque de voler de la bande passante aux autres flux. C'est pourquoi l'estimation correcte de la congestion sur un chemin de bout-en-bout est cruciale pour le fonctionnement des protocoles de transport dans l'Internet.

D'un autre côté, la connaissance du niveau de congestion sur des liens ou chemins spécifiques du réseau est nécessaire pour la surveillance et la gestion des réseaux. Cette information est utile aux opérateurs réseau qui gèrent des réseaux autonomes dans l'Internet. Connaître le niveau de congestion sur des chemins spécifiques dans leur réseau leur permet de juger des performances de leur réseau. Ils peuvent utiliser cette information pour prendre des décisions concernant le routage du trafic dans leur réseau, et pour en améliorer certaines parties. De même, des fournisseurs de service qui offrent des services à des clients situés sur différents réseaux ont besoin de savoir si les chemins qu'ils utilisent sont congestionnés ou non.

Dans la première partie de la thèse nous proposons des solutions qui permettent d'améliorer la qualité de l'estimation de la congestion à la fois sur les chemins de bout-en-bout et sur les liens individuels de l'Internet.

A.1.2 Problèmes et contributions

Pertes de congestion et erreurs de transmission

S'appuyant sur l'implémentation de TCP de Jacobson [3], la plupart des protocoles de transport d'Internet imitent aujourd'hui le comportement de TCP et interprètent chaque perte de paquet comme un symptôme de congestion sur un chemin de bout-en-bout. Avec l'expansion des technologies sans fil, Internet comporte aujourd'hui différents mécanismes de réseaux d'accès sans fil comme les wireless LANs et les réseaux Mobiles 3G, ainsi que quelques liaisons internes sans fil. Les liens sans fil sont sujets aux erreurs de transmission et très souvent, la couche de liaison de données sans fil ne fournit qu'une fiabilité partielle. Dans un tel réseau hybride, les pertes de paquets ne sont plus seulement dues à la conegstion mais aussi aux erreurs de transmission dans les liaisons sans fil. Afin d'utiliser au mieux la bande passante disponible, un protocole de transport doit réduire son débit d'émission uniquement en réponse à des pertes de congestion, et non en réponse à des pertes de transmission sans fil. Cela motive le problème

de la différenciation entre les pertes de congestion et les pertes dues aux erreurs de transmissions, afin que les protocoles de transport puissent estimer la congestion de bout-en-bout avec précision. Le problème a été en premier étudié par rapport à TCP par les auteurs de [4]. Ils ont proposé l'installation d'un agent de surveillance dans les stations de base pour empêcher un émetteur TCP de réduire inutilement son débit en réponse aux erreurs de transmission. Dans cette thèse nous considérons le problème de la différenciation de pertes par rapport aux protocoles de transport non fiables qui transportent des flux multimédia.

Il existe deux approches de la différenciation de pertes : de *bout-en-bout* et *explicite*. Dans la différentiation de pertes de bout-en-bout, le récepteur au niveau transport estime la cause d'une perte de paquet sans aucune aide des éléments internes du réseau. Les mécanismes de différenciation de pertes de bout-en-bout s'appuient sur des éléments comme les délais séparant chaque arrivée de paquets. Cependant, aucun de ces mécanisme ne parvient à différencier assez efficacement les pertes, et se trompent souvent dans la classification des pertes de congestion et de transmission sans fil. Nous proposons un mécanisme précis et explicite de différenciation de pertes appelé AED qui utilise des agents aux extrémités des liens sans fil [5]. Les agents enregistrent dans les paquets non perdus la cause de la perte de paquets de manière intelligente et à faible coût. AED permet aux protocoles de transport de différencier avec précision les pertes de congestion des pertes de transmission sans fil.

Estimation fiable des pertes de paquets

Le niveau de congestion d'une liaison ou d'un chemin précis dans le réseau peut être obtenu par deux approches : (i) en installant des agents de surveillance sur les éléments du réseau, et (ii) par des mesures de bout-en-bout. Lorsque les réseaux à surveiller s'étendent sur plusieurs domaines, il devient quasiment impossible d'installer des équipements spécifiques dans les éléments du réseau pour surveiller chaque lien ou chemin. Même dans le cas d'un unique réseau à administrer, cela exige la mise à jour coûteuse de plusieurs éléments du réseau. De ce fait, les caractéristiques des liens et chemins internes d'un réseau sont souvent estimés à partir de mesures effectuées de bout-en-bout.

Le domaine d'étude de réseaux dans lesquels les caractéristiques internes sont déduites par des mesures bout-en-bout est apparu sous la dénomination de tomographie des réseaux (du fait de sa similitude avec la tomographie médicale) [6]. Une des premières méthodes proposées pour réaliser la tomographie d'un réseau a été l'estimation de pertes *MINC* (*Mutlicast-based Inference of Network internal Characteristics*) [7]. Basée sur des mesures effectuées en multipoint de bout-en-bout, l'estimation MINC permet d'estimer les taux de pertes (c'est-à-dire les niveaux de congestion) sur des liaisons précises du réseau. Pour obtenir cette estimation, la source envoie des paquets sondes dans l'arbre de transmission multipoint et chaque destinataire doit répondre avoir reçu ou non le paquet. La source et les destinataires sont de simples terminaux sur Internet. À partir des rapports de réception (sous forme binaire) des traces collectées auprès de chaque destinataire, l'estimation MINC estime les taux de pertes de chaque liaison de l'arbre de transmission multipoint. Cependant, en raison de la présence de destinataires multipoints défectueux ou malhonnêtes, ces rapports de réception peuvent être incorrects, et ainsi fausser l'estimation des taux de pertes. Une estimation incorrecte peut indiquer de forts taux de pertes sur de bonnes liaisons ou de faibles taux sur des liaisons mauvaises. Dans cette thèse, nous traitons le problème de la vérification des mesures binaires de transmissions multipoint afin d'assurer une estimation cohérente et fiable des taux de pertes sur les liens du réseau.

Nous proposons un algorithme de vérification statistique appelé *ICheck* qui permet de vérifier que les données des rapports de réception récupérés auprès des destinataires multipoints sont cohérents [8]. Cet algorithme reprend les idées de l'estimation de pertes par MINC en exploitant les corrélations inhérentes entre rapports binaires de traces récupérées auprès des différents destinataires d'un arbre de transmission multipoint. Schématiquement, l'algorithme se base sur le principe que les rapports de différents sous-ensembles de destinataires peuvent être utilisés pour estimer le taux de pertes d'une seule liaison de l'arbre de transmission multipoint. De ce fait, en comparant les différentes estimations du taux de pertes d'une liaison, les incohérences entre les rapports de réception peuvent être détectées. Cet algorithme a deux propriétés remarquables : il n'a pas besoin de connaître la topologie de l'arbre de transmission multipoint, et il est capable de détecter des incohérences même en présence de destinataires conspirant pour

fausser les résultats.

Tomographie à faible coût

Afin d'estimer les taux de pertes par des mesures actives de bout-en-bout, des infrastructures dédiées doivent être déployées. Pour des mesures effectuées sur des arbres de transmission multipoint de grande taille, cette tâche est complexe. Elle nécessite l'accès à des sites situés en différents endroits d'Internet et la possibilité d'y installer des logiciels de mesure. Généralement les sites utilisent différents systèmes d'exploitation, ce qui augmente encore la difficulté d'installation et de configuration. Pour faciliter la prise de mesures multipoints de bout-en-bout, les auteurs de [9] ont proposé une architecture passive qui associe l'envoi des rapports de réception avec *RTCP* (*Real-Time Control Protocol*) [10]. Dans cette architecture, les sessions multipoint utilisant *RTP* (*Real-time Transport Protocol*) [10] pour transférer les données peuvent utiliser leur paquets de données pour explorer le réseau et retourner les rapports utiles à l'estimation de pertes MINC sur les paquets RTCP. Cependant, une des contraintes dans ce cas est que les destinataires ne sont pas en mesure d'envoyer les rapports pour chaque paquet, sans quoi la bande passante de retour en RTCP pourrait dépasser 5% de la bande passante des données de la session RTP. Dans cette thèse, nous nous intéressons au problème de l'estimation de pertes MINC à partir de moins de rapports de réception.

Dans l'architecture proposée par les auteurs de [9], les rapports de réception sont limités pour maintenir une faible bande passante des retours RTCP, c'est-à-dire que les destinataires envoient des rapports correspondant à un ensemble réduit (échantillonné) des paquets sondes, et ces rapports sont utilisés pour l'estimation des taux de pertes.

Nous avons conçu un estimateur de pertes MINC étendu (*EMLE–Extended MINC Loss Estimator*) qui permet d'estimer les pertes en utilisant des rapports cumulés [11]. Chaque rapport cumulé est retourné par un groupe de w paquets d'exploration et peut être représenté par moins de w bits. Ainsi, l'estimateur EMLE permet l'estimation de taux de pertes en utilisant moins de bits en retour. Nous comparons l'estimateur EMLE avec un estimateur MINC qui utilise un sous-ensemble des rapports.

A.2 Ingénierie de trafic dans les réseaux multipoints

Cette seconde partie de la thèse est consacrée à l'ingénierie de trafic multipoint dans les réseaux superposés. Le but de l'ingénierie de trafic est "d'améliorer la performance d'un réseau, tant au niveau du trafic qu'à celui des ressources" [12]. L'objectif principal est de diriger le trafic multipoint vers des arbres sélectionnés de telle sorte que les performances attendues soient atteintes et que les ressources réseau soient utilisées de manière optimale.

Les réseaux superposés sont des réseaux virtuels auto-organisés dans lesquels les sites participants forment un réseau au niveau applicatif. On peut citer comme exemples de réseaux superposés *RON* (*Resilient Overlay Networks*) [13], *OMNI* (*Overlay Multicast Network Infrastructure*) [14], Akamai, et plusieurs réseaux pair-à-pair. Dans ces réseaux, les nœuds, généralement des serveurs, peuvent implanter des fonctions réseau intelligentes et configurables au niveau applicatif, que les routeurs sur Internet ne peuvent normalement pas. Bien que les services implantés au niveau applicatif soient moins efficaces, la propriété remarquable des réseaux superposés est qu'ils sont aisément déployables.

A.2.1 Problème et Contribution

En étudiant les caractéristiques des chemins de routage superposés, les nœuds de la surcouche peuvent diriger le trafic dynamiquement pour équilibrer la charge sur les chemins ou routes selon certaines contraintes, et ainsi faire de l'ingénierie du trafic. Un des services qui peut être offert par les réseaux superposés est le routage multipoint. Afin d'utiliser au mieux les conditions du trafic sur le réseau superposé, les nœuds peuvent construire des arbres pour diriger le trafic intelligemment. Pour cela, ils ont besoin de décrire des arbres de transmission multipoints de manière précise dans le réseau superposé. Cependant, les mécanismes de maintenance des différents arbres du réseau superposé qui introduisent un état par arbre sur les nœuds ne passe à l'échelle. De plus, les nœuds du réseau superposé peuvent être limités en termes de capacité mémoire. Le problème du routage multipoint explicite sans états est encore rarement attaqué dans la littérature.

Une approche pour réaliser un routage explicite sans état est le routage multipoint à la source, c'est-à-dire l'encodage de l'arbre de transmission multipoint au sein des paquets de données. Dans cette thèse, nous considérons cette approche et nous avons mis au point des techniques d'encodages efficaces d'arbres de transmission multipoint dans les paquets de données [15]. Nous montrons la similitude entre les arbres de transmission multipoint et les structures d'arbres théoriques, et à partir de cette similitude nous montrons l'existence de bornes inférieures simples sur le nombre de bits nécessaires pour représenter des arbres de transmission multipoint. Nos encodages sont quasi optimaux en termes d'espace mémoire et peuvent être lus et traités efficacement. Ils peuvent être utilisés pour représenter des arbres de transmission multipoint dans des paquets de données afin de router le trafic multipoint sur des arbres de transmission explicites sans utiliser d'états.

A.3 Conclusions

Dans cette thèse, nous avons abordé des problèmes liés à deux sujets: l'estimation de la congestion sur Internet et l'ingénierie du trafic multipoint dans les réseaux superposés.

Estimer avec précision la congestion est un besoin crucial pour le fonctionnement des protocoles de transport sur Internet. Avec le développement des liens sans fil sur Internet, les protocoles de transport doivent distinguer les pertes dues aux erreurs de transmission sur les liens sans fil, des pertes liées à la congestion afin de pouvoir évaluer le réel niveau de congestion de bout-en-bout. Au chapitre 2, nous avons présenté AED, un mécanisme de différenciation des pertes qui permet à des protocoles de transport non fiables de faire la distinction entre les pertes de congestion et les pertes qui se produisent sur les liens sans fils. Dans AED, les agents sont déployés aux extrémités des liens sans fils. Ces agents notifient la cause des pertes correspondant à une fenêtre de paquets, dans des paquets non perdus, avec un faible surcoût en taille mémoire des paquets. AED peut être utilisé en complément à d'autres protocoles tel que TFRC. Nous avons montré que même si la taille des fenêtres était petite, AED se comportait beaucoup mieux que les méthodes de différenciation de pertes existant à l'heure actuelle.

Estimer le niveau de congestion des liens est essentiel pour le contrôle et la gestion des réseaux. Cette information est utile pour les opérateurs réseaux et les fournisseurs d'accès car elle leur permet de prendre des décisions concernant le routage du trafic au sein de leur propre réseau ou bien la mise à jour de certains composants de leur réseau. Aux chapitres 3 et 4, nous avons évoqué deux problèmes liés à un outil de tomographie de réseaux appelée MINC. MINC utilise des mesures multipoints pour évaluer le taux de perte des différents liens et consécutivement, le niveau de congestion.

Afin d'utiliser de manière fiable les informations fournies par les mesures de bout-en-bout, il est essentiel de vérifier l'intégrité des données mesurées. Les données binaires communiquées à MINC en retour lui permettent d'estimer les taux de pertes des liens; mais elles peuvent être erronées du fait de la présence de récepteurs défaillants ou mal intentionnés. Les données incorrectes conduisent à de mauvaises estimations des taux de pertes. De bons liens peuvent alors être répertoriés comme ayant un fort taux de pertes et réciproquement. Au chapitre 3, nous avons présenté un algorithme statistique de vérification appelé *ICheck*. *ICheck* vérifie l'intégrité des données binaires transmises par les récepteurs multipoints. En conformité avec la nature de bout-en-bout de la tomographie réseau, *ICheck* n'a pas besoin de connaître l'arbre multipoint de la topologie et fonctionne même en présence de récepteurs défectueux ou mal intentionnés. *ICheck* aide à évaluer si un certain ensemble de données transmis par les récepteurs multipoints conduira à des estimations de pertes dignes de confiance ou non.

ICheck vérifie les informations fournies par des groupes de trois récepteurs et renvoie le nombre d'incompatibilités. Des recherches futures dans ce domaine tenteront de répondre aux questions suivantes : (i) peut-on définir une quantité représentant la probabilité qu'une donnée reçue soit incorrecte (ii) Connaissant la topologie de l'arbre de transmission multipoint, est-ce que les informations provenant de différents groupes peuvent être regroupées pour identifier les récepteurs fallacieux. (iii) Est-ce que des tests similaires peuvent être réalisés pour vérifier l'intégrité des données RTCP RR (*Receiver Reports*) transmises par les récepteurs multipoints.

Réaliser des mesures multipoints à grande échelle pour estimer des taux de pertes nécessite la mise en place d'infrastructures adaptées et constitue donc une tâche complexe. Celle-ci

peut être simplifiée si les récepteurs des sessions multipoints peuvent effectuer des mesures à partir de leurs propres paquets de données et renvoyer ces mesures via RTCP. Cependant, étant donné que la bande passante utilisée par RTCP ne doit pas excéder 5% de la bande passante des données, nous devons trouver un moyen pour réaliser les estimations de pertes en utilisant moins de bande passante pour récupérer les mesures. Au chapitre 4, nous avons conçu une extension de l'estimateur de perte MINC (EMLE) qui évalue le taux de perte des liens en utilisant des données agrégées. Les données agrégées utilisent moins de bande passante. De plus, nous avons montré que EMLE, en utilisant ces données, réalise les estimations de pertes sans détérioration significative de la précision. Une autre approche pour réduire la bande passante en retour consiste à diminuer la taille des données en ne prenant en compte qu'un échantillon des mesures reçues. Les données ainsi réduites sont transmises à l'estimateur de pertes MINC. Nous avons comparé les deux méthodes ci-dessus en utilisant des modèles et des simulations sur NS et nous avons conclu que la méthode de réduction des données est la plus efficace pour diminuer les besoins en bande passante. Notre travail sur EMLE a aussi révélé que les estimateurs de pertes et de délais MINC peuvent être modifiés pour supprimer les calculs numériques.

Des travaux futurs consisteront à effectuer des expériences pour comparer EMLE et sa méthode de réduction des données, au nouvel estimateur [34] basé sur le moment. Ceci exigera des expériences similaires à celles réalisées par [9] utilisant un émulateur RTP/RTCP pour capturer des données réseaux réalistes. Il reste également à déterminer si EMLE peut être étendu de manière similaire à l'estimateur de pertes MINC pour incorporer le processus de perte des données reçues *MAR*(*missing at random*).

Le but de l'ingénierie du trafic multipoint est d'orienter le trafic sur des arbres spécifiques ou explicites pour équilibrer la charge dans le réseau et vérifier certaines contraintes liées au trafic. Maintenir un seul arbre pour router tout le trafic multipoint risque d'encombrer certains chemins alors que d'autres liens peuvent être sous-utilisés. La tâche de l'ingénierie du trafic multipoint est réalisable dans les réseaux superposés car les noeuds sont généralement des serveurs ou des PC programmables pour réaliser des tâches intelligentes et configurables. Effectuer les

mêmes tâches avec des routeurs nécessite leur mise à jour. En présence de trafic multipoint, nous avons besoin de méthodes pour guider le trafic sur des arbres spécifiques ou explicites. Au chapitre 5, nous avons présenté des manières efficaces pour encoder des arbres multipoints au sein des paquets de données. Ces techniques d'encodage peuvent être utilisées dans les réseaux superposés pour router le trafic multipoint sur des arbres explicites sans utiliser d'états. Cet encodage est quasi optimal en terme d'espace mémoire occupé et peut être calculé et décodé efficacement. En utilisant des expériences avec des arbres multipoints dans des topologies réseaux réelles et simulées, nous avons montré que ces techniques peuvent être utilisés de manière efficace pour encoder dans des paquets de données des arbres correspondant à des groupes multipoints de petites ou de moyennes tailles. Nous avons également démontré la correspondance entre les arbres multipoints et la notion théorique d'arbre en tant que structure de données et nous avons obtenu une borne inférieure au nombre de bits nécessaires pour représenter un arbre multipoint.

BIBLIOGRAPHY

[1] Raj Jain, "Congestion control in computer networks: issues and trends," *IEEE Network*, pp. 24–30, 1990. 1, 104

[2] K. Ramakrishnan, S. Floyd, and D. Black, "The Addition of Explicit Congestion Notification (ECN) to IP," RFC 3168 (Proposed Standard), September 2001. 1, 10, 104

[3] V. Jacobson, "Congestion avoidance and control," in *SIGCOMM '88: Proceedings of the conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, 1988, pp. 314–329. 2, 105

[4] Hari Balakrishnan, Srinivasan Seshan, and Randy H. Katz, "Improving reliable transport and handoff performance in cellular wireless networks," *ACM Wireless Networks*, vol. 1, no. 4, pp. 469–481, 1995. 3, 9, 106

[5] Vijay Arya and Thierry Turletti, "Accurate and Explicit Differentiation of Wireless and Congestion Losses," in *ICDCS Workshop on Mobile and Wireless Networks*, May 2003, pp. 877–882. 3, 106

[6] Y. Vardi, "Network tomography: Estimating source-destination traffic intensities from link data," *Journal of the American Statistical Association*, pp. 365–377, 1996. 3, 27, 107

[7] A. Adams, T. Bu, T. Caceres, N.G. Duffield, T. Friedman, J.Horowitz, F. Lo Presti, S.B. Moon, V. Paxson, and D. Towsley, "The use of End-to-end Multicast Measurements for Characterising Internal Network Behavior," *IEEE Communications Magazine*, vol. 38, no. 5, pp. 152–158, May 2000. 3, 24, 26, 27, 50, 51, 52, 53, 107

[8] Vijay Arya, Thierry Turletti, and Ceilidh Hoffmann, "Feedback Verification for Trustworthy Tomography," in *Workshop on Internet Performance, Simulation, Monitoring and Measurement (IPS-MOME)*, March 2005. 4, 107

[9] R. Caceres, N.G. Duffield, and T. Friedman, "Impromptu measurement infrastructures using RTP," in *IEEE INFOCOM*, June 2002, pp. 1490–1499. 4, 25, 47, 49, 50, 51, 52, 75, 100, 108, 112

[10] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications," RFC 3550 (Standard), July 2003. 4, 8, 9, 25, 29, 50, 51, 108

[11] Vijay Arya, Thierry Turletti, Timur Friedman, Remy Bellino, and N. G. Duffield, "Low Feedback MINC Loss Tomography," in *IEEE INFOCOM Student Workshop*, March 2005. 5, 108

[12] D. Awduche, A. Chiu, A. Elwalid, I. Widjaja, and X. Xiao, "Overview and Principles of Internet Traffic Engineering," RFC 3272 (Informational), May 2002. 5, 80, 109

[13] David G. Andersen, Hari Balakrishnan, M. Frans Kaashoek, and Robert Morris, "The Case for Resilient Overlay Networks," in *Proceedings of 8th Annual Workshop on Hot Topics in Operating Systems*, May 2001. 5, 80, 109

[14] S.Y.Shi and J.S.Turner, "Routing in overlay multicast networks," in *IEEE INFOCOM*, June 2002. 5, 80, 85, 109

[15] Vijay Arya, Thierry Turletti, and Shivkumar Kalyanaraman, "Encodings of Multicast trees," in *IFIP Networking Conference*, May 2005, pp. 992–1004. 5, 110

[16] Sally Floyd, Mark Handley, Jitendra Padhye, and Jorg Widmer, "Equation-based congestion control for unicast applications," in *SIGCOMM '00: Proceedings of the conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, 2000, pp. 43–56. 8

[17] I. Rhee, V. Ozdemir, and Y. Li, "TEAR: TCP emulation at recievers-flow control for multimedia streaming," Tech. Rep., Department of Computer Science, NCSU, April 2000. 8

[18] Deepak Bansal and Hari Balakrishnan, "Binomial congestion control algorithms," in *IEEE INFOCOM*, April 2001, pp. 631–640. 8, 9

[19] Y. R. Yang and S. S. Lam, "General AIMD congestion control," in *Proceedings of 8th ICNP*, November 2000. 8

[20] S. Biaz and N.H. Vaidya, "Discriminating congestion losses from wireless losses using inter-arrival times at the reciever," in *IEEE Symposium ASSET'99*, March 1999. 8, 10

[21] Y. Tobe, Y. Tamura, A. Molano, S. Ghosh, and H. Tokuda, "Acheiving moderate fairness for UDP flows by pathstatus classification," in *25th Annual IEEE Conference on Local Computer Networks*, November 2000, pp. 252–61. 8, 10

[22] S. Cen, P.C. Cosman, and G.M. Voelker, "End-to-end differentiation of congestion and wireless losses," in *Multimedia Computing and Networking (MMCN)*, January 2002, pp. 1–15. 8, 10, 11, 18, 19

[23] E. Kohler, M. Handley, J. Padhye, and S. Floyd, "Datagram congestion control protocol(DCCP)," May 2002, http://www.icir.org/kohler/dcp/. 9, 14

[24] M. Handley, J. Padhye, and S. Floyd, "TCP Friendly Rate Control(TFRC): Protocol specification," April 2002, http://www.icir.org/tfrc/. 9

[25] Giao Thanh Nguyen, Randy H. Katz, Brian Noble, and M. Satyanarayanan, "A trace-based approach for modeling wireless channel behavior," in *Winter Simulation Conference*, 1996, pp. 597–604. 16, 17, 21

[26] F. Li, N. Seddigh, B. Nandy, and D. Malute, "An empirical study of today's internet traffic for differentiated services IP QoS," in *Proceedings of ISCC*, 2000. 22

[27] R. Caceres, N. G. Duffield, J. Horowitz, and D. Towsley, "Multicast-based inference of network-internal loss characteristics," *IEEE Transactions on Information Theory*, vol. 45, pp. 2462–2480, 1999. 24, 27, 43, 46, 50, 52, 58, 66

[28] Francesco Lo Presti, N. G. Duffield, J. Horowitz, and Don Towsley, "Multicast-based inference of network-internal delay distributions," *IEEE/ACM Transactions on Networking*, vol. 10, no. 6, pp. 761–775, 2002. 24, 27, 50, 53

[29] Andrew Whitaker and Richard S. Cox and Steven D. Gribble, "Configuration Debugging as Search: Finding the Needle in the Haystack," in *6th Symposium on Operating System Design and Implementation (OSDI)*, 2004. 25

[30] Dawson Engler and Madanlal Musuvathi, "Model-checking large network protocol implementations," in *Network System Design and Implementation (NSDI)*, 2004. 25

[31] V. Paxson, A. K. Adams, and Matt Mathis, "Experiences with NIMI," in *Passive and Active Measurement workshop*, April 2000. 25

[32] B. Tierney and D. Gunter, "Netlogger: A toolkit for distributed system performance tuning and debugging," Tech. Rep., LBNL, 2002. 25

[33] Sergey Gorinsky, Sugat Jain, and Harrick M. Vin, "Multicast Congestion Control with Distrusted Receivers," in *Networked Group Communication*, 2002, pp. 19–26. 25, 28

[34] N. G. Duffield, V. Arya, R. Bellino, T. Friedman, J. Horowitz, D. Towsley, and T. Turletti, "Network Tomography from Aggregate Loss Reports," in *Submitted to IFIP Performance*, 2005. 25, 76, 100, 112

[35] T. Friedman, R. Caceres, and A. Clark, "RTP Control Protocol Extended Reports (RTCP XR)," RFC 3611 (Proposed Standard), November 2003. 25, 52

[36] N.G. Duffield and F. Lo Presti, "Multicast Inference of Packet Delay Variance at Interior Network Links," in *IEEE INFOCOM*, March 2000, pp. 1351–1360. 27, 53

[37] N. G. Duffield, J. Horowitz, D. Towsley, W. Wei, and T. Friedman, "Multicast-based loss inference with missing data," *IEEE Journal on Selected Areas of Communications*, vol. 20, no. 4, pp. 700–713, 2002. 27, 75

[38] R. Caceres, N.G. Duffield, J. Horowitz, F. Lo Presti, and D. Towsley, "Loss-Based Inference of Multicast Network Topology," in *IEEE Conference on Decision and Control*, December 1999. 27, 30, 38, 53

[39] Sylvia Ratnasamy and Steven McCanne, "Inference of Multicast Routing Trees and Bottleneck Bandwidths Using End-to-end Measurements," in *IEEE INFOCOM*, 1999, pp. 353–360. 27, 30, 38, 53

[40] Claudia Tebaldi and Mike West, "Bayesian inference on network traffic using link count data," *Journal of the American Statistical Association*, pp. 557–596, 1998. 27

[41] Jin Cao, Drew Davis, Scott Vander Wiel, and Bin Yu, "Time-varying network tomography: router link data," *Journal of the American Statistical Association*, vol. 95, pp. 1063–1075, 2000. 27

[42] Jorg Widmer and Mark Handley, "Extending equation-based congestion control to multicast applications," in *SIGCOMM '01: Proceedings of the conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, 2001, pp. 275–285. 28

[43] S. Paul, K.Sabnani, J.C Lin, and S. Bhattacharya, "Reliable Multicast transport Protocol (RMTP)," *IEEE Journal on selected areas in Communication*, vol. 15, no. 3, pp. 407 – 421, April 1997. 28

[44] Luigi Rizzo, "pgmcc: a TCP-friendly single-rate multicast congestion control scheme," in *SIGCOMM '00: Proceedings of the conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, 2000, pp. 17–28. 28

[45] Brett J. Vickers, Célio Albuquerque, and Tatsuya Suda, "Source-adaptive multilayered multicast algorithms for real-time video distribution," *IEEE/ACM transactions on Networking*, vol. 8, no. 6, pp. 720–733, 2000. 28

[46] Steven McCanne, Van Jacobson, and Martin Vetterli, "Receiver-driven layered multicast," in *SIGCOMM '96: Proceedings of the conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, 1996, pp. 117–130. 28

[47] Lorenzo Vicisano, Luigi Rizzo, and Jon Crowcroft, "TCP-Like congestion control for layered multicast data transfer," in *IEEE INFOCOM*, 1998, pp. 996–1003. 28

[48] John W. Byers, Michael Frumin, Gavin B. Horn, Michael Luby, Michael Mitzenmacher, Alex Roetter, and William Shaver, "FLID-DL: Congestion control for layered multicast," in *Networked Group Communication*, 2000, pp. 71–81. 28

[49] D Sisalem and A wolisz, "MLDA: A TCP-friendly congestion control framework for heterogeneous multicast environments," in *Eighth International Workshop on Quality of Service (IWQoS)*, June 2000. 28

[50] J. Vieron, T. Turletti, K. Salamatian, and C. Guillemot, "Source and channel adaptive rate control for multicast layered video transmission based on a clustering algorithm," *EURASIP, Special Issue on Multimedia over IP and Wireless Networks*, 2004. 28

[51] S.Y. Cheung, M.H. Ammar, and X. Li, "On the use of destination set grouping to improve fairness in multicast video distribution," in *IEEE INFOCOM*, March 1996, pp. 553–560. 28

[52] Sergey Gorinsky, Sugat Jain, Harrick Vin, and Yongguang Zhang, "Robustness to inflated subscription in multicast congestion control," in *SIGCOMM '03: Proceedings of the conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, 2003, pp. 87–98. 28

[53] A. Agresti, "A Survey of Exact Inference for Contingency Tables," *Statistical Science*, vol. 7, no. 1, pp. 131–177, 1992. 38

[54] Cyrus R. Mehta and Nitin R. Patel, "ALGORITHM 643: FEXACT: a fortran subroutine for fisher's exact test on unordered r x c contingency tables," *ACM Transactions on Mathematical Software*, vol. 12, no. 2, pp. 154–161, 1986. 40

[55] Mehta C. R. and Patel N.R, "A network algorithm for performing fisher's exact test in rxc contingency tables," *Journal of the American Statistical Association*, vol. 78, no. 382, pp. 427–434, 1983. 40

[56] M. Yajnik, J. Kurose, and D. Towsley, "Packet Loss Correlation in the MBone Multicast Network," in *Global Internet Conference*, November 1996. 43, 46

[57] "MBone Traces," ftp://gaia.cs.umass.edu/pub/yajnik/. 43

[58] Jean-Chrysotome Bolot, "End-to-end packet delay and loss behavior in the internet," in *SIGCOMM'93 : Proceedings of the conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, 1993, pp. 289–298. 46

[59] R. Caceres, N.G. Duffield, S. B. Moon, and D. Towsley, "Inference of internal loss rates in the MBone," in *IEEE/ISOC Global Internet*, December 1999. 51

[60] Sherlia Shi, *Design of Overlay Networks for Internet Multicast*, Ph.D. thesis, Washington University at St. Louis, August 2002. 80, 85

[61] Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, and Hari Balakrishnan, "Chord: A scalable peer-to-peer lookup service for internet applications," in *SIGCOMM '01: Proceedings of the conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, 2001, pp. 149–160. 80

[62] Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, and Scott Schenker, "A scalable content-addressable network," in *SIGCOMM '01: Proceedings of the conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, 2001, pp. 161–172. 80

[63] A. Rowstron and P. Druschel, "Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems," in *Proceedings of IFIP/ACM International Conference on Distributed Systems Platforms*, 2001. 80, 82

[64] Ben Y. Zhao, John D. Kubiatowicz, and Anthony D. Joseph, "Tapestry: An Infrastructure for Fault-tolerant Wide-area Location and Routing," Tech. Rep. UCB/CSD-01-1141, UC Berkeley, 2001. 80, 82

[65] Suman Banerjee, Christopher Kommareddy, Koushik Kar, Samrat Bhattacharjee, and Samir Khuller, "Construction of an efficient overlay multicast infrastructure for real-time applications," in *IEEE INFOCOM*, March 2003. 81, 85

[66] Hema Tahilramani Kaur, Shivkumar Kalyanaraman, A. Weiss, S. Kanwar, and A. Gandhi, "BANANAS: an evolutionary framework for explicit and multipath routing in the internet," *Computer Communication Review*, vol. 33, no. 4, pp. 277–288, 2003. 81

[67] D. Waitzman, C. Partridge, and S.E. Deering, "Distance Vector Multicast Routing Protocol," RFC 1075 (Experimental), November 1988. 81

[68] D. Estrin, D. Farinacci, A. Helmy, D. Thaler, S. Deering, M. Handley, V. Jacobson, C. Liu, P. Sharma, and L. Wei, "Protocol Independent Multicast-Sparse Mode (PIM-SM): Protocol Specification," RFC 2117 (Experimental), June 1997. 81

[69] Stephen Deering, Deborah L. Estrin, Dino Farinacci, Van Jacobson, Ching-Gung Liu, and Liming Wei, "The pim architecture for wide-area multicast routing," *IEEE/ACM Transactions on Networking*, vol. 4, no. 2, pp. 153–162, 1996. 81

[70] Ion Stoica, *Stateless Core: A Scalable Approach for Quality of Service in the Internet*, Ph.D. thesis, Department of Electrical and Computer Engineering, Carnegie Mellon University, December 2000. 82

[71] A. Rowstron, A-M. Kermarrec, M. Castro, and P. Druschel, "SCRIBE: The design of a large-scale event notification infrastructure," in *Networked Group Communication*, 2001. 82, 85

[72] Shelley Q. Zhuang, Ben Y. Zhao, Anthony D. Joseph, Randy H. Katz, and John Kubiatowicz, "Bayeux: An Architecture for Scalable and Fault-tolerant Wide-Area Data Dissemination," in *Proceedings of NOSSDAV*, June 2001, pp. 11–20. 82, 85

[73] Michael R. Macedonia, Michael J. Zyda, David R. Pratt, Paul T. Barham, and Steven Zeswitz, "Npsnet: A Network Software Architecture For Large Scale Virtual Environments," in *Presence: Teleoperators and virtual Environments*, 1994. 82

[74] J. Mark Pullen and David Wood, "Network technology for DIS," in *Proceedings of IEEE*, August 1995, pp. 1156–1167. 82

[75] George Popescu and Zhen Liu, "Stateless application-level multicast for dynamic group communication," in *IEEE International Symposium on Distributed Simulation and Real-Time Applications (DS-RT'04)*, 2004, pp. 20–28. 82, 83, 85

[76] C. W Kong, M. Gouda, and S. LamK, "Secure group communications using key graphs," *IEEE/ACM Transactions on Networking*, vol. 8, no. 1, pp. 16–30, February 2000. 82

[77] Torsten Braun, Vijay Arya, and Thierry Turletti, "A Backup Tree Algorithm for Multicast Overlay Networks," in *IFIP NETWORKING Conference*, May 2005, pp. 1430–1434. 83, 85

[78] Mozafar Bag-Mohammadi1, Siavash Samadian-Barzoki1, and Nasser Yazdani, "Linkcast: Fast and scalable multicast routing protocol," in *NETWORKING*, May 2004, pp. 1282–1287. 83, 85

[79] Rick Boivie et al, "Explicit Multicast (Xcast) Basic Specification, INTERNET DRAFT draft-ooms-xcast-basic-spec-06.txt," 2004. 84

[80] Y. H. Chu, S. G. Rao, , and H. Zhang, "A case for end system multicast," in *Proceedings of ACM SIGMETRICS*, 2000, pp. 1–12. 85

[81] P. Francis, "white paper http://www.aciri.org/yoid/," . 85

[82] B. Zhang, S. Jamin, and L. Zhang, "Host Multicast: A framework for delivering multicast to end users," in *IEEE INFOCOM*, June 2002. 85

[83] Zhi Li and Prasant Mohapatra, "Hostcast: A new overlay multicasting protocol," in *IEEE International Communications Conference (ICC)*, 2003. 85

[84] S. Banerjee, B. Bhattacharjee, and C. Kommareddy, "Scalable application layer multicast," in *SIGCOMM '02: Proceedings of the conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, 2002, pp. 205–217. 85

[85] Y. Chawathe, *Scattercast: An Architecture for Internet Broadcast Distribution as an Infrastructure Service*, Ph.D. thesis, University of California, Berkeley, August 2000. 85

[86] Ion Stoica, T. S. Eugene Ng, and Hui Zhang, "Reunite: A recursive unicast approach to multicast.," in *IEEE INFOCOM*, March 2000, pp. 1644–1653. 85

[87] Luís Henrique Maciel Kosmalski Costa, Serge Fdida, and Otto Carlos Muniz Bandeira Duarte, "Hop by hop multicast routing protocol," in *SIGCOMM '01: Proceedings of the conference on Applications, technologies, architectures, and protocols for computer communications*, August 2001, pp. 249–259. 86

[88] E. Rosen, A. Viswanathan, and R. Callon, "Multiprotocol Label Switching Architecture," RFC 3031 (Proposed Standard), January 2001. 86

[89] David Benoit, Erik D. Demaine, J. Ian Munro, and Venkatesh Raman, "Representing trees of higher degree," in *International Workshop on Algorithms and Data Structures(WADS'99)*, August 1999, pp. 169–180. 91

[90] Robert C. Chalmers and Kevin C. Almeroth, "On the topology of multicast trees," *IEEE/ACM Transactions on Networking*, vol. 11, no. 1, pp. 153–165, 2003. 91

[91] Graham Phillips, Scott Shenker, and Hongsuda Tangmunarunkit, "Scaling of multicast trees: comments on the chuang-sirbu scaling law," in *SIGCOMM'99: Proceedings of Conference on Applications, technologies, architectures, and protocols for computer communication*, 1999, pp. 41–51. 91

[92] Danny Dolev, Osnat Mokryn, and Yuval Shavitt, "On multicast trees: Structure and size estimation," in *IEEE INFOCOM*, March 2003. 91

[93] Ken Calvert, Matt Doar, and Ellen W. Zegura, "Modelling internet topology," *IEEE Communications Magazine*, vol. 35, no. 6, pp. 160–163, June 1997. 95

[94] "SCAN project," http://www.isi.edu/scan/mercator/maps.html. 95

[95] Neil Spring, Ratul Mahajan, David Wetherall, and Thomas Anderson, "Measuring isp topologies with rocketfuel," *IEEE/ACM Transactions on Networking*, vol. 12, no. 1, pp. 2–16, February 2004. 95

RÉSUMÉ

Cette thèse présente des mécanismes pour obtenir de meilleures estimations de la congestion sur Internet. Elle présente également un mécanisme ayant des applications sur le trafic multipoint dans des réseaux overlay. Tout d'abord, nous présentons une méthode de différenciation des pertes qui permet aux protocoles de transport non fiables d'estimer avec précision la congestion de bout-en-bout, en distinguant les pertes liées à la congestion des pertes liées aux erreurs de transmission sur les liaisons sans fil. Ensuite, nous présentons deux contributions relatives à un outil de tomographie de réseau appelé MINC. MINC utilise des mesures multipoints de bout-en-bout pour estimer les pertes et donc la congestion des liaisons internes du réseau. Nous proposons d'une part un algorithme statistique qui vérifie les mesures binaires effectuées en multipoint par MINC pour estimer les pertes. Cet algorithme permet de garantir une estimation fiable des taux de pertes sur les différents liens. D'autre part, nous proposons une nouvelle version de l'estimateur de perte MINC. Notre estimateur utilise les informations multipoints cummulées fournies en retour pour évaluer les taux de pertes sur tous les liens. Il peut être utilisé dans des situations où la bande passante pour transmettre les rapports de reception est faible. Enfin, nous présentons des techniques efficaces pour encoder les arbres de transmission multipoint au sein des paquets de données. Ces techniques d'encodage peuvent être utilisées pour implanter un routage multipoint explicite et sans état dans des réseaux overlay et ont donc des applications dans le domaine d'ingénierie de trafic multipoint.

Mots-clés: mécanisme de différenciation de pertes de paquets, inference de congestion, tomographie de réseau, MINC, ingénierie de trafic multipoint, encodage des arbes multipoint, multipoint explicite

ABSTRACT

This thesis presents methods which help to improve the quality of congestion inference on both end-to-end paths and internal network links in the Internet and a method which helps to perform multicast traffic engineering in Overlay Networks. First, we propose an explicit loss differentiation scheme which allows unreliable transport protocols to accurately infer congestion on end-to-end paths by correctly differentiating congestion losses from wireless losses. Second, we present two contributions related to Multicast-based Inference of Network Characteristics (MINC). MINC is a method of performing network tomography which infers loss rates, i.e. congestion, on internal network links from end-to-end multicast measurements. We propose a statistical verification algorithm which can verify the integrity of binary multicast measurements used by MINC to perform loss inference. This algorithm helps to ensure a trustworthy inference of link loss rates. Next, we propose an extended MINC loss estimator which can infer loss rates of network links using aggregate multicast feedbacks. This estimator can be used to perform loss inference in situations where the bandwidth to report multicast feedbacks is low. Third, we present efficient ways of encoding multicast trees within data packets. These encodings can be used to perform stateless and explicit multicast routing in overlay networks and thus achieve goals of multicast traffic engineering.

Keywords: loss differentiation, congestion inference, network tomography, MINC, multicast traffic engineering, multicast tree encodings, explicit multicast