Robust Congestion Control for IP Multicast

Sergey Gorinsky

Technical Report TR2003-33

Department of Computer Sciences

University of Texas at Austin

Taylor Hall 2.124, Austin, TX 78712, USA

gorinsky@cs.utexas.edu

August 6, 2003

The Dissertation Committee for Sergey Gorinsky

certifies that this is the approved version of the following dissertation:

Robust Congestion Control for IP Multicast

Committee:

Harrick M. Vin, Supervisor

Lorenzo Alvisi

Sanjoy K. Baruah

Michael D. Dahlin

Aloysius K. Mok

K. K. Ramakrishnan

Robust Congestion Control for IP Multicast

by

Sergey Gorinsky, Eng., M.S.

Dissertation

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

Doctor of Philosophy

The University of Texas at Austin

August 2003

In memory of Edsger Wybe Dijkstra,

a year after his sudden and saddening departure

Acknowledgments

Firstly, I would like to thank Harrick Vin for guiding my doctoral studies to the successful completion. My path though graduate schooling started in New Jersey, and Sanjoy Baruah has served as an inspiration and mentor to me ever since. I am grateful to Al Mok for encouraging me to join Texas and helping me with his wise advice from my first days in Austin. A summer internship at AT&T Labs Research under supervision of K.K. Ramakrishnan, Matthias Grossglauser, and Nick Duffield awoke my interest in robust multicast congestion control. The research continued and matured into this dissertation in the friendly atmosphere of LASR (Laboratory for Advanced Systems Research) led by Lorenzo Alvisi, Mike Dahlin, and Harrick Vin. The dissertation would not become a reality without the sustained moral support from my family, especially from my mother Galina, father Vladimir, and wife Andrea. Also, I greatly appreciate all the help provided by Sara Strandtman, Nina Amla, Yuliya Babovich, Emery Berger, Mikhail Bilenko, Doug Burger, Deji Chen, Edsger Dijkstra, Esra Erdem, Anna Gal, Jeff Golden, Eugene Gorbatov, Mohamed Gouda, Xingang Guo, Sam Guyer, Daniel Jimenez, Jasleen Kaur, Ravi Kokku, Xiaozhou Li, Taroon Mandhana, Jean-Philippe Martin, Ramgopal Mettu, Jayaram Mudigonda, Jeff Napper, Amol Nayate, Scott Page, Seth Pettie, Greg Plaxton, Jefferson Provost, Gloria Ramirez, Taylor Riche, Jacob Sarvela, Peter Stone, Robert van de Geijn, Arun Venkataramani, Praveen Yalagandula, Thomas Wahl, Jian Yin, David Zuckerman, and many other people from the Department of Computer Sciences at the University of Texas. I particularly enjoyed my long-term productive collaboration with Sugat Jain on multicast congestion control. These research efforts were substantially enhanced by Yongguang Zhang's expertise in computer security. Finally, I thank the anonymous reviewers of my SIGCOMM 2003, SIGMETRICS 2003, NGC 2002, and other papers for extensive and helpful comments.

Sergey Gorinsky

The University of Texas at Austin

August 2003

Robust Congestion Control for IP Multicast

Publication No. ____________________

Sergey Gorinsky, Ph.D.
The University of Texas at Austin, 2003

Supervisor: Harrick M. Vin

IP multicast is a network service for scalable distribution of data to multiple receivers. Traditional protocols for multicast congestion control rely on trust: each party is assumed to follow guidelines for fair bandwidth sharing. However, with the growth and commercialization of the Internet, the assumption of universal trust is no longer tenable. In this dissertation, we consider a relaxed model where receivers are untrustworthy and can misbehave to acquire an unfairly high bandwidth at the expense of competing traffic. Our experiments with existing multicast protocols show that each of the evaluated protocols is vulnerable to receiver misbehavior.

To take the first step towards robust multicast designs for distrusted environments, we focus on the class of feedback-free protocols where receivers provide no feedback to the sender and control congestion by regulating their subscription levels in the multi-group session. Unfortunately, the mechanism of group subscription offers a misbehaving receiver an opportunity to inflate its subscription level. Such inflated subscription attacks pose a major threat to fairness of bandwidth allocation.

This dissertation is the first to solve the problem of inflated subscription. The presented designs rely on an insight that the ability of a receiver to access a multicast group should be tied with the congestion status of the receiver. First, we address individual attacks where a receiver inflates its subscription with no assistance from other receivers. Our solution guards access to multicast groups with dynamic keys and consists of two independent components: DELTA (Distribution of ELigibility To Access) – a novel method for in-band distribution of group keys to receivers that are eligible to access the groups according to the congestion control protocol, and SIGMA (Secure Internet Group Management Architecture) – a generic architecture for key-based group access at edge routers. DELTA and SIGMA require only minimal generic changes in the edge routers, do not alter the core of the network, and introduce no auxiliary servers. Then, we extend the design to protect multicast congestion control against inflated subscription of colluding receivers. To illustrate that integration with DELTA and SIGMA makes multicast protocols robust to inflated subscription and preserves other congestion control properties, we derive and evaluate robust adaptations of RLM and FLID-DL protocols.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Multicast is a service that distributes data to multiple receivers. A wide range of emerging applications – such as video streaming, multi-party gaming, dissemination of news and emergency alerts – can greatly benefit from this service. A scalable implementation of the multicast service cannot rely on direct unicast communication between the data source and each receiver. To disseminate data to a large population of receivers, the sender relies on a distribution hierarchy of intermediaries that duplicate and forward data to the receivers.

Internet Protocol (IP) multicast [13] refers to multicast implementations that construct the data distribution hierarchy from network routers. The central concept in IP multicast is a multicast group identified by a single address. Each receiver subscribes to a multicast group by submitting the group address to the local edge router via Internet Group Management Protocol (IGMP) [16], and the network uses multicast routing protocols – such as Distance Vector Multicast Routing Protocol (DVMRP) [51], Core Based Trees (CBT) [4], Protocol Independent Multicast (PIM) [12], Multicast extensions to Open Shortest Path First (MOSPF) [32], Source-Specific Multicast (SSM) [23], and Simple Multicast (SM) [40] – to organize its routers into a logical tree for distribution of packets from the sender to the subscribed receivers.

Whereas distribution trees that branch at routers enable the most efficient dissemination of data both in terms of the network bandwidth utilization and delivery time, slow deployment of IP multicast has stirred recently an interest in end-system implementations of the multicast service where end hosts form the data distribution hierarchies [1, 5]. In this dissertation, we primarily focus on IP multicast but discuss the end-system implementations in some detail in Chapter 5.

By itself, IP multicast provides only the basic forwarding functionality – each router forwards multicast packets to its output links that lead to receivers. To be useful, any communication service should also employ congestion control mechanisms ensuring that the generated traffic uses the network resources fairly and efficiently, e.g., by utilizing the bottleneck bandwidth without causing a heavy loss of packets. Furthermore, some applications require reliability mechanisms that overcome packet losses to deliver all the packets to all the receivers.

Traditional unicast communications in the Internet rely predominantly on Transmission Control Protocol (TCP) that supplies both congestion control and reliability [2, 24]. Upon delivery of a data packet, the receiver of a TCP session transmits an acknowledgment packet to the sender. In its turn, the sender monitors and analyzes the stream of acknowledgment packets from the receiver. If the sender concludes that a data packet has been lost, the sender retransmits the packet and reduces the overall transmission rate. As long as the feedback from the receiver does not indicate a packet loss, the sender raises its transmission rate to consume any unutilized bandwidth.

With respect to congestion control, two important features distinguish multicast from unicast:

- *Receiver Multiplicity.* If each receiver of a large multicast session reports its congestion status directly to the sender, the feedback can overwhelm the sender and the network. Consequently, scalable feedback-driven protocols for multicast congestion control employ additional mechanisms to avoid the feedback implosion. Also due to the scalability considerations, the sender does not know the capabilities, identities, or even the number of the receivers in the session.

- *Receiver Heterogeneity.* If a multicast session has receivers with heterogeneous capabilities, transmission at a single rate does not fully accommodate these diverse capabilities. Thus, a single multicast group is often unable to satisfy all the receivers. Multicast congestion control protocols address the receiver heterogeneity by composing a session from several multicast groups and thereby allowing a receiver to align its received rate with its capability via subscription to a suitable subset of the groups. Hence, multicast protocols use the mechanism of group subscription not only to form multicast groups in a scalable manner but also to control congestion.

Below, we review multicast congestion control mechanisms – and their instantiations in prominent protocols – in more detail.

1.1 Multicast Congestion Control Mechanisms

To avoid feedback implosion, feedback-driven multicast protocols limit the amount of information communicated from the receivers to the sender. Aggregation and suppression are two alternative mechanisms for providing the sender with a brief summary of the session congestion status.

In *feedback aggregation*, receivers pass their reports up along the edges of a logical tree rooted at the sender. Internal nodes of the tree reduce the amount of the feedback by consolidating the provided information: each internal node gathers reports from its subtree, compiles their summary, and transmits a new report with the aggregated information towards the root. Various implementations of feedback aggregation have been proposed. Some protocols – such as Reliable Multicast Transport Protocol (RMTP) [39] – build the aggregation tree entirely from receivers. Schemes like Source-Adaptive Multi-Layered Multicast (SAMM) [50] aggregate feedback in routers or other network devices.

In *feedback suppression*, a receiver reports its status directly to the sender. Unlike feedback aggregation, this mechanism does not rely on intermediaries to generate new reports with aggregated information. Instead, feedback suppression filters out those reports that do not refine the current summary of the session congestion status. For example, in TCP-Friendly Multicast Congestion Control (TFMCC) [53] where the congestion summary is the fair rate for the slowest receiver, the sender multicasts its current summary to the session and thereby cancels reports from the receivers with higher fair rates. Multicast of the congestion summary is not the only implementation of feedback suppression. Some protocols – such as Pragmatic General Multicast Congestion Control (pgmcc) [43] – suppress feedback at routers:

a router discards reports that do not refine the feedback forwarded by this router earlier.

To address heterogeneity of receivers, multicast congestion control protocols compose a session from several multicast groups. By joining and leaving the groups through IGMP, each receiver controls its level of participation in the session. In such multi-group protocols, *group subscription* becomes a congestion control mechanism. In fact, Receiver-driven Layered Multicast (RLM) [30], Receiver-driven Layered Congestion control (RLC) [49], Fair Layered Increase/Decrease with Dynamic Layering (FLID-DL) [7], Wave and Equation Based Rate Control (WEBRC) [28] provide no feedback to the sender and control congestion through regulation of group membership.

Fairness of bandwidth allocation in a multi-group session depends on the ability of a receiver to converge to its fair subscription level. To facilitate this convergence, some multicast congestion control protocols incorporate a mechanism for *subscription synchronization*. Once again, there exist different implementations of this mechanism. In RLM, receivers coordinate their actions via so-called shared learning: before subscribing to a group, a receiver announces its intention to the other receivers. RLC and FLID-DL synchronize subscriptions through explicit signals from the sender: a receiver can add a group only upon an increase signal; increase signals are sent less frequently to receivers with higher subscription levels. Receivers in WEBRC coordinate their subscriptions by converging to rates derived from an equation for TCP-friendly throughput [37].

While group membership regulation and feedback-driven transmission adjustment constitute two different paradigms for multicast congestion control, they

Paradigms	Mechanisms	Protocols		
		Single-group	Feedback-free	Multi-group feedback-driven
Feedback-driven transmission adjustment	Feedback generation	RMTP, SAMM, TFMCC, pgmcc		DSG, SIM, MLDA
	Feedback aggregation	RMTP, SAMM		SIM
	Feedback suppression	TFMCC, pgmcc		DSG, MLDA
Group membership regulation	Group subscription		RLM, RLC, FLID-DL, WEBRC	DSG, SIM, MLDA
	Subscription synchronization		RLM, RLC, FLID-DL, WEBRC	DSG, SIM, MLDA

Table 1.1: Classification of multicast protocols according to their congestion control mechanisms

are not mutually exclusive. Combining these paradigms in one design improves fairness and efficiency of bandwidth allocation in heterogeneous multicast environments [14, 22]. Destination Set Grouping (DSG) [10, 11], congestion control with Selective participation, Intra-group transmission adjustment, and Menu adaptation (SIM) [21], and Multicast Loss-Delay based Adaption (MLDA) [47] are multi-group feedback-driven protocols that adjust both membership and transmission rates of the groups.

Table 1.1 classifies the mentioned prominent multicast protocols with respect to their congestion control mechanisms.

1.2 Diversity of Multicast Protocols

As the previous section demonstrates, different multicast protocols employ different mechanisms to control congestion. Congestion control mechanisms are not the only source of diversity in multicast protocols. Below, we classify multicast protocols along four other dimensions: (1) reliability, (2) congestion notification, (3) session structure, and (4) congested state.

Reliability. TCP uses the same mechanism to provide both congestion control and reliability: when a data packet is lost, the sender retransmits the packet and reduces the overall transmission rate. Such a combined design approach does not lend itself straightforwardly to multicast. Whereas total reliability requires recovery from each packet loss, reduction of the transmission rate in response to each packet loss is not always a right strategy for multicast congestion control. Since the forwarding tree of a multicast group can contain a number of branches that lose packets independently, reacting to each loss can lead to a situation where the transmission rate of the group is much lower than the fair value [6]. Consequently, fair and efficient multicast protocols adjust the transmission rate of a multicast group in response to the feedback from a representative receiver, e.g., the receiver with the highest loss rate. To support reliable delivery, such protocols need to employ an additional mechanism.

The heterogeneity of multicast applications is another reason for decoupling congestion control and reliability in multicast. Whereas some applications – such as software distribution – require delivery of all the data to all the receivers, the total reliability is not a must in other domains. For example, a multicast session that disseminates layered video strives to provide each receiver only with the packets from a subset of the data layers – more specifically, from the layers that fit within the fair bandwidth share of the receiver. Furthermore, some multimedia applications can tolerate occasional packet losses and do not need explicit support for reliability as long as congestion control mechanisms keep the loss rate low.

The above considerations explain the diversity of multicast congestion control protocols with respect to their support for reliability. Many protocols – including TFMCC, SAMM, RLM, RLC, FLID-DL, WEBRC, SIM, and MLDA – provide no such support and thus are called *unreliable* [7, 21, 26, 28, 30, 47, 49, 50, 53]. Among those protocols that are *reliable*, techniques for supporting the reliability vary. Some reliable protocols – including RMTP and pgmcc – retransmit lost packets [17, 27, 34, 35, 38, 39, 43]. Others rely on a proactive mechanism of forward error correction: the sender transmits not only packets with the original data but also redundant packets that carry error correction codes; as long as a receiver obtains a substantially large subset of the transmitted packets, the receiver can reconstruct all the original data [9, 49].

Congestion notification. Whereas the purpose of reliability mechanisms is to overcome packet losses, traditional congestion control introduces losses deliberately and uses them as an indication that the transmission has saturated all the available bandwidth. The loss-driven congestion control suffers from a number of disadvantages. For example, when an output link of a drop-tail router becomes fully utilized, the router discards packets only after they fill up the link buffer, and this buffering reduces the responsiveness of congestion control and delays the delivery of data.

An appealing alternative to the loss-driven control is Explicit Congestion Notification (ECN): when the link is saturated, the router sets a bit in the headers of forwarded packets to notify receivers about the congestion explicitly; the notification enables network sessions to avoid losses by decreasing the transmission earlier, before the link buffer is full [42]. Since the recovery from packet losses in a multicast session is more challenging than in unicast, ECN is particularly useful for multicast protocols.

Session structure. Different multi-group protocols impose a different structure on a multicast session and prescribe a different set of rules for group subscription. For instance, in replicated multicast protocols [10, 11], each group of a session delivers the same content at a different rate, and a receiver reacts to congestion by switching from its only subscribed group to a slower one. On the other hand, in layered multicast protocols [7, 21, 26, 28, 30, 45, 47, 49], groups of a session carry cumulative layers of hierarchically encoded data, and a receiver controls congestion by subscribing to an appropriate stack of lower groups. Byers, Luby, and

Mitzenmacher [8] also propose non-cumulative layered multicast protocols where any combination of the groups in a session constitutes a legitimate useful subscription, i.e., a session with N groups offers $2^N - 1$ subscription levels.

Congested state. Some multicast protocols – such as FLID-DL and RLC – define congestion as a single packet loss. Other protocols often ignore occasional loss of packets and consider a receiver to be congested only when its loss rate exceeds a threshold. For instance, the default threshold for each subscription level in RLM is 25% of the packets transmitted to this level. MLDA and WEBRC are examples of protocols that use the TCP-friendly throughput equation to define a lower threshold for each higher subscription level in the session.

1.3 Trust Modeling

The above review of multicast congestion control protocols showed their great diversity with respect to the definition of the congested state, session structure, and mechanisms for congestion notification and reliability. Despite the differences, the multicast protocols share a common feature – their congestion control assumes that each party always adheres to guidelines for fair sharing of the network bandwidth.

Congestion control protocols have rested on this implicit assumption of universal trust since the early days of the Internet. As long as the network served solely as an experimental test bed shared by a close-knit community of scientists and engineers, ignoring the trust issues in bandwidth allocation was perhaps defensible. However, the Internet has grown into a global commercial system where thousands of independent Internet Service Providers (ISPs) compete and cooperate via dozens of protocols to provide a variety of distributed applications to millions of users. Consequently, networking became impersonal – it is common that a user knows neither all the ISPs that provide communication services to the user nor other customers that fulfill their communication needs over the same bottleneck links. Although this very separation promoted the Internet expansion (by freeing the users from the need to be aware of what is happening inside the network), it also weakened incentives for following the etiquette of fair bandwidth sharing. Since the network bandwidth is a valuable asset, some parties involved in Internet communications can be interested in eliciting unfair bandwidth allocations by manipulating Internet protocols. Furthermore, widespread deployment of open-source operating systems provides users with ample opportunities for such misbehavior.

Defining a representative model for trust relationships in the Internet is a hard problem. Below, we discuss three aspects of the trust modeling: (1) trusted base, (2) dynamics of trust relationships, and (3) degree of distrust.

1.3.1 Trusted Base

Universal trust is an extreme point in the trust space. Another extreme is universal distrust, a situation where no entity – an end host, router, server, or link – is trusted. Designing robust Internet protocols under this pessimistic assumption would require from a protocol participant to verify all the information received from the other entities. Even if possible to be realized, the taxing verification would consume computational and communicational resources of the network and consequently undermine the efficiency of communications.

Fortunately, the assumption of universal distrust is not very realistic either. The likelihood of misbehavior varies substantially among Internet constituents. For example, a party that uses the Internet only to email text messages is unlikely to circumvent congestion control protocols because transmitting the messages at unfairly high rates would not usually provide the party with practical benefits from the shorter delivery times; on the other hand, since bandwidth cheating is often the only way for a receiver to obtain a high-resolution live video stream over the modern Internet infrastructure, such a receiver has stronger incentives to misbehave.

Thus, a realistic trust model needs to partition the Internet into a *trusted base* and distrusted elements. For example, [44] presents a model where the trusted base includes all the network infrastructure and information providers; only information consumers are considered untrustworthy. To be robust under such trust models, Internet protocols can employ traditional techniques for supporting the communications within the trusted base and verify only the information provided by distrusted elements. Such protocols can be substantially lighter than the designs derived under the assumption of universal distrust.

1.3.2 Dynamics of Trust Relationships

Some trust models link trustworthiness to functionality, e.g., routers can be viewed as always trusted. Trust relationships in such function-based models are *static*, and robust protocols can use the identity of a network component to decide whether the component is a part of the trusted base. However, one might argue that functionality is not a reliable indicator of trustworthiness, e.g., it seems reasonable to trust routers of a reputable ISP but be distrustful of routers belonging to a newly emerged provider. Moreover, network components can change ownership or be victimized by an intruder. Thus, more elaborate models represent trust as *dynamic* relationships. Under such a model, robust protocols cannot include an entity into the trusted base simply by authenticating the entity. To verify the trustworthiness, they need additional mechanisms that track the behavior of the entity in earlier communications.

1.3.3 Degree of Distrust

Trust modeling involves not only identifying the elements that are untrustworthy but also specifying the degree of the distrust. One alternative is to assume arbitrary deviations from the prescribed behavior. Such a model operates in terms of *binary* trust relationships: an entity either enjoys the absolute trust or has no credibility whatsoever. Once again, the most pessimistic assumption is not a realistic way to characterize misbehavior. Rather than engaging in random actions, a misbehaving entity is more likely to set up coordinated attacks in order to achieve some specific goals. Hence, more realistic models define the degree of distrust for an entity by describing the attacks that the entity can launch. Below, we slice the continuum of possible attacks using the two criteria of *collusion* and *intent*.

First of all, an entity can stage an *individual attack* without any assistance from other distrusted elements. A protocol can acquire robustness against the individual attack by verifying the information provided by the distrusted element and monitoring its actions. This protection can be insufficient when distrusted elements conspire to join their forces and launch a *collusion attack*. For example, if the verification mechanism relies on a majority voting, the colluding elements can dupe the mechanism by submitting multiple copies of an incorrect report. If the protection employs keys, the colluding elements can share the keys to overcome the protection. Also, whereas individual actions of each colluding entity might seem benign, their cumulative impact can be devastating. Thus, it is more difficult to achieve robustness in models that add collusion to the list of possible attacks.

The intent is a criterion enabling us to separate misbehavior of distrusted entities into *denial-of-service attacks* and *self-beneficial attacks*. In denial-of-service attacks, disruption of network services is a sole goal of the misbehavior. Consequently, the attacker strives for highly visible disruptions, and the magnitude of the damage is a measure of its success. The intentional visibility of denial-of-service attacks facilitates their detection – an unusually low level of service is an indicator that the network is potentially under attack. On the other hand, a self-beneficial attack is driven by a desire to gain selfish benefits from the elicited unfair bandwidth allocation. The damage to communications of other parties is collateral and rather undesirable. To avoid detection and thereby preserve the unfairly acquired bandwidth, self-beneficial attacks are interested in keeping a low profile. For example, instead of shutting down the competing traffic, the attacker has incentives to subdue this traffic to a level that the abused parties can falsely interpret as fair. Thus, self-beneficial attacks can be sneakier and more difficult to discern.

Another distinction of denial-of-service attacks is their richer arsenal. To waste bandwidth, an attacker can transmit spurious data or subscribe to multiple sessions even if the attacker has no interest in their content. Such attacks are purely malicious; the attacker itself does not benefit from the wasted bandwidth. Opportunities for self-beneficial attacks are less ample. For example, to acquire an unfairly high bandwidth for obtaining the data within a session, a receiver has to manipulate its congestion control protocol. Since the manipulation opportunities are limited, protection against self-beneficial attacks can be more effective. Whereas defense against denial-of-service is *reactive* and relies on detection and punishment, it is often possible to *prevent* self-beneficial attacks.

In comparison to widely publicized denial-of-service incidents, insidious self-beneficial attacks have stirred much less attention among researchers. On the other hand, sneaky self-beneficial misbehavior is far from harmless. In the Internet, the population of bandwidth-greedy users exceeds greatly the number of hackers interested only in disrupting the communications of others. Even inside large intra-enterprise network environments, selfish misbehavior cannot be discounted. Due to the tangible incentives offered by self-beneficial attacks, the frequency and cumulative impact of such attacks can be much higher. Recent studies of TCP congestion control showed that a misbehaving receiver can substantially increase its throughput at the expense of cross traffic [15, 44]. Thus, even if a small percentage of TCP receivers launches self-beneficial attacks, this misbehavior can severely disrupt network services. In multicast congestion control [18, 19, 20], self-beneficial attacks are even more diverse and dangerous.

1.4 Dissertation Outline

In this dissertation, we examine the impact of relaxing the assumption of universal trust on the design of multicast congestion control protocols. In Chapter 2, we define our trust model and identify potential threats to multicast congestion control. Our evaluation of existing multicast protocols shows that each of the evaluated protocols is vulnerable to at least one of the identified threats. Based on the dominance of feedback-free multicast protocols, we argue that inflated subscription of misbehaving receivers represents the major threat to fairness of network bandwidth allocation. Chapter 3 proposes solutions for the problem of inflated subscription. Our main contribution is an insight that the ability of a receiver to access a multicast group should be tied with the congestion status of the receiver. First, we exploit this insight to provide multicast protocols with robustness against individual attacks where a multicast receiver inflates its subscription with no assistance from other parties. The proposed design guards access to multicast groups with dynamic keys and consists of two independent components: DELTA (Distribution of ELigibility To Access) – a novel method for in-band distribution of group keys to receivers that are eligible to access the groups according to the congestion control protocol, and SIGMA (Secure Internet Group Management Architecture) – a generic architecture for key-based group access at edge routers. Then, we extend the design of DELTA and SIGMA to protect multicast congestion control against inflated subscription of colluding receivers. Chapter 4 applies the designed mechanisms to derive robust versions of RLM and FLID-DL multicast protocols. Finally, Chapter 5 summarizes the contributions of the dissertation.

Chapter 2

Threats to Multicast Congestion Control

This chapter explores the impact of trust assumptions on multicast congestion control. In our trust model, information sources and network providers (and hence network links, routers, and servers) are trustworthy and always adhere to their protocols. The trust relationships are static and exclude only receivers from the trusted base. One justification for this model comes from the observation that a receiver is primarily interested in maximizing its own throughput whereas information sources and network providers are a part of the managed infrastructure and have an interest in treating their customers fairly. Furthermore, the trust in routers is essential for ensuring the fairness of bandwidth allocation because a router has a decisive word in allocating the bandwidth of its output links. Figure 2.1 depicts our trust model by placing the network infrastructure within a sphere of trust. Local interfaces of edge routers are the only points of access for network users. For instance, a receiver

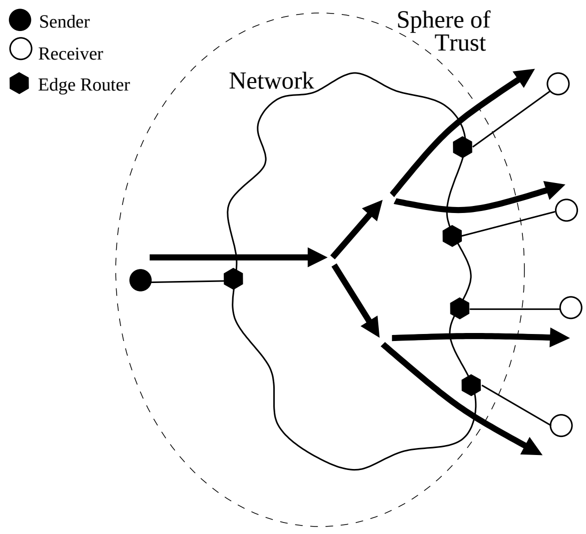


Figure 2.1: Multicast congestion control with distrusted receivers

can subscribe to a multicast group only by communicating with a local router. To characterize the degree of distrust, the model assumes that receivers misbehave only to acquire unfairly high bandwidth at the expense of competing traffic. Thus, a misbehaving receiver launches self-beneficial bandwidth attacks but does not act from pure malice to stage denial-of-service attacks. However, the types of the potential self-beneficial misbehavior include both individual attacks and collusion attacks.

A similar trust model has been studied earlier for TCP congestion control. In that model, the network infrastructure and senders of TCP sessions are trustworthy but the receivers are distrusted and can misbehave to acquire data at an unfairly high rate (see Figure 2.2). [15, 44] demonstrate that by abusing either acknowledgment packets or ECN feedback to the sender, a TCP receiver can inflate the transmission and increase substantially the rate of reliable delivery. In proposed solutions, the sender protects against the misbehavior by verifying the correctness of the congestion

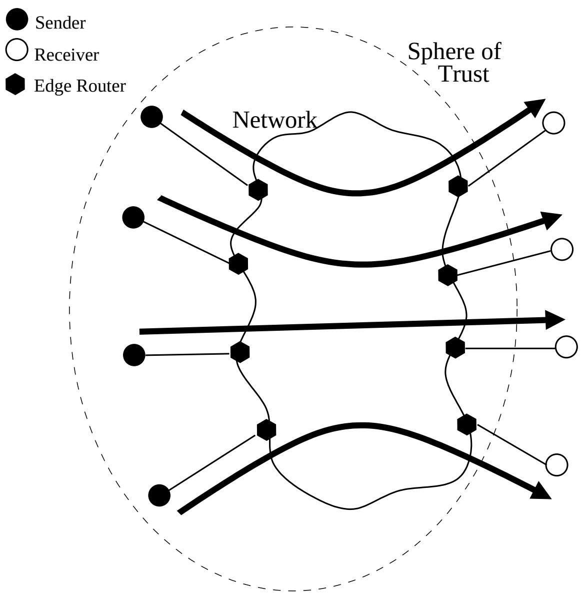


Figure 2.2: Unicast congestion control with distrusted receivers

reports supplied by the receiver.

Is it possible to use the same protection techniques to derive robust protocols for multicast congestion control? As we discussed in Section 1.1, multicast protocols need additional congestion control mechanisms, e.g., a feedback suppression mechanism for avoiding feedback implosion or a group subscription mechanism for addressing receiver heterogeneity. The additional mechanisms give multicast receivers extra opportunities for manipulating congestion control. For example, a misbehaving receiver of a multi-group session can acquire an unfairly high bandwidth by maintaining an unfairly high subscription level; also, a misbehaving receiver can deceive a feedback-driven multicast protocol into an unfairly high transmission by failing to report or by suppressing legitimate reports from other receivers. Note that verification of feedback correctness at the sender does not protect against inflated subscription or incomplete feedback. Thus, unicast-style methods of protection do not make multicast congestion control protocols robust to receiver misbehavior.

In what follows, we analyze multicast congestion control mechanisms and identify their potential susceptibilities to receiver misbehavior. Subsequent experiments with prominent multicast protocols confirm that each of the evaluated protocols is vulnerable to at least one of these threats. Finally, we discuss approaches for eliminating the vulnerabilities and single out inflated subscription of misbehaving multicast receivers as a major threat to fair bandwidth allocation.

2.1 Threat Model

To explore the spectrum of potential vulnerabilities in multicast congestion control, we define a threat as a general pattern of multicast receiver misbehavior that can reward the misbehaver with an unfair bandwidth advantage over other receivers in the network. We compile the list of threats by reexamining the multicast congestion control mechanisms discussed in Section 1.1.

The paradigm of feedback-driven transmission adjustment engages multicast receivers in providing the sender with a summary of the session congestion status. The sender uses this information to adjust its transmission. By distorting the congestion summary, a misbehaving receiver can trick the sender into unfairly high transmission. After the inflated transmission forces well-behaving cross traffic to recede, the misbehaving receiver unfairly acquires the released bandwidth. This general attack of inflated transmission comes in various instantiations that exploit different vulnerabilities in the control mechanisms of the feedback-driven paradigm.

Feedback generation intrinsically resides in receivers: each receiver prepares and transmits reports about its congestion status. To distort the congestion summary, a misbehaving receiver can issue *incorrect reports.* This threat is analogous to receiver misbehavior in unicast congestion control. However, incorrect reports are not the only threat to feedback generation in multicast. *Failure to report* can also boost transmission by distorting the congestion summary.

In feedback aggregation, each internal node of the aggregation tree replaces incoming feedback with a smaller number of aggregated reports. If the aggregation tree consists of receivers, a misbehaving receiver inside the tree can issue *forged aggregated reports* that ignore or falsify information provided to the misbehaver by other receivers.

Feedback suppression uses a report from a receiver to filter out subsequent feedback that does not refine this earlier report. *Manipulation with feedback suppression* through a spurious report can also distort the congestion summary.

In the paradigm of group membership regulation, group subscription allows a receiver to select its subscription level in a multi-group session. Since IGMP does not restrict multicast group membership, a misbehaving receiver can join those groups where transmission exceeds the fair rate for the misbehaver. The unfairly high subscription rewards the misbehaver with an unfairly high throughput after the competing well-behaving traffic recedes. Thus, *inflated subscription* poses a threat to fairness of multicast congestion control.

The mechanism of subscription synchronization coordinates actions of receivers to facilitate convergence to fair subscription levels. If a receiver's decision to join or to leave a group depends on information supplied by another receiver, a misbehaving receiver can manipulate the subscription levels of the others. By *preventing other receivers from subscription*, a misbehaving receiver keeps their subscription levels unfairly low and thus acquires an unfair bandwidth advantage over them.

To sum up the above discussion, we list the six threats of multicast receiver misbehavior that represent potential attacks in our trust model:

1. *Incorrect reports*,
2. *Failure to report*,
3. *Forged aggregated reports*,
4. *Manipulation with feedback suppression*,
5. *Inflated subscription*, and
6. *Prevention of other receivers from legitimate subscription.*

2.2 Evaluation of Vulnerabilities

In this section, we evaluate whether existing multicast protocols are susceptible to the identified threats. For each threat in our model, we evaluate one protocol from Table 1.1. Since our model defines threats with respect to control mechanisms, we select a representative protocol for a threat from the table row for the corresponding mechanism.

We use NS-2 [36] and conduct all our experiments in the same network. Figure 2.3 marks bottleneck links with their capacities. The capacity of each unmarked link is 100 Mbps. All the links have a delay of 10 msec and a buffer for two bandwidth-delay products. Multicast sessions M and N control congestion using

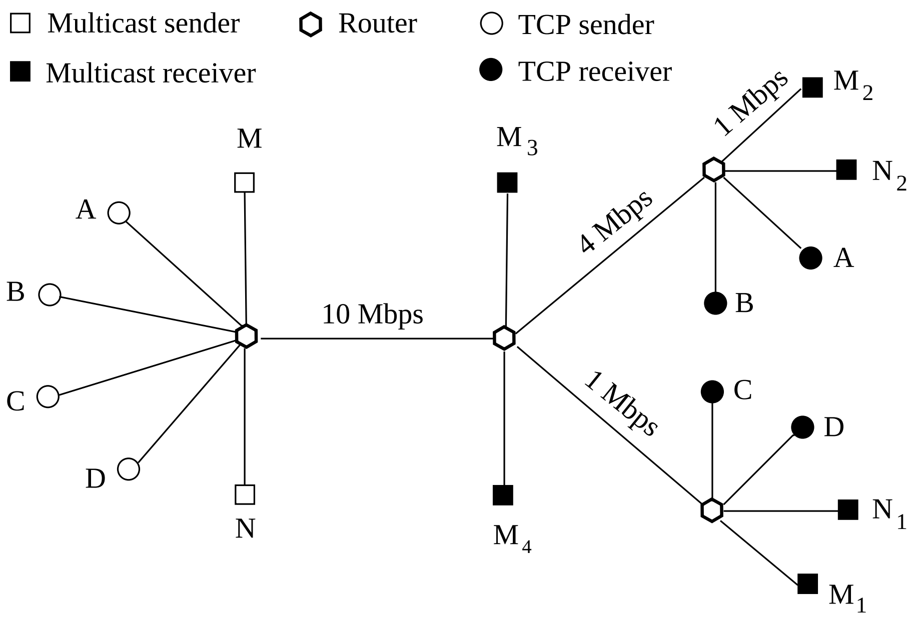


Figure 2.3: Experimental network topology

the evaluated multicast protocol. Session M serves four receivers M_1, M_2, M_3 and M_4 that can misbehave. Well-behaving receivers N_1 and N_2 compose session N. Unicast sessions A, B, C, and D adhere to TCP Reno. Each sender transmits as much data as its protocol allows. The packet size in each session is 1000 bytes.

We run each simulation for 200 seconds. Unless we state explicitly otherwise, a misbehaving receiver starts its attack 100 seconds into the experiment. We measure throughput and loss rates for the misbehaver and other receivers. For reliable protocols, we consider only sequentially delivered data to compute the throughput. In unreliable protocols, the reported throughput reflects all delivered data.

2.2.1 Incorrect reports in TFMCC

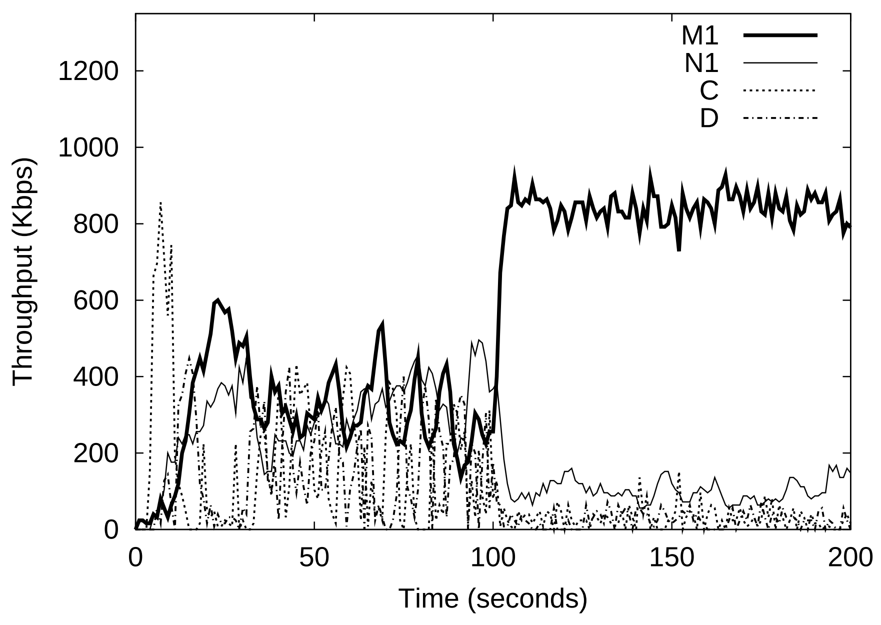
TFMCC [53] is a single-group protocol where each receiver uses an equation for TCP-friendly throughput to calculate its fair rate. The sender adjusts its transmission to the lowest of the fair rates reported by the receivers.

The slowest receiver can attack TFMCC by reporting an exaggerated rate and boosting the transmission. However, the misbehaver does not benefit if the inflated transmission swamps its bottleneck link and causes persistent heavy losses. Also, the misbehavior does not raise the transmission beyond the smallest rate reported by a well-behaving receiver. To profit the most from the attack, the misbehaving receiver can adjust the reported exaggerated rate and maintain the fastest transmission that does not result in congestion.

In our experiment, M_1 is the only misbehaving receiver. The fair rate for M_1 is 250 Kbps. The slowest well-behaving receiver M_2 has a fair rate of 1 Mbps. After 100 seconds, M_1 misbehaves by reporting a rate of 900 Kbps. Figure 2.4 shows that the attack rewards M_1 with a substantial throughput advantage over well-behaving receivers C, D, and N_1. Figure 2.4 also presents the corresponding loss rates.

2.2.2 Failure to report in TFMCC

To attack TFMCC, the slowest receiver can also choose to be silent and boost the transmission to the smallest rate reported by a well-behaving receiver. If the inflated transmission overloads its bottleneck link, the misbehaver detects the persistent losses and discontinues the attack as disadvantageous. In comparison to incorrect reports, failure to report gives the misbehaver less control over the transmission. However, if the sender in TFMCC would verify the correctness of reported rates,



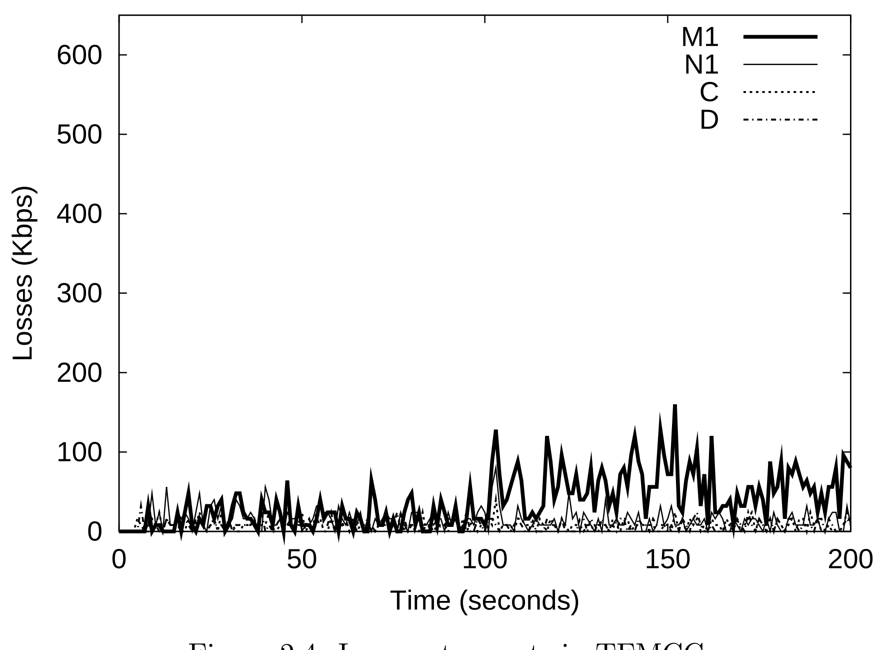


Figure 2.4: Incorrect reports in TFMCC

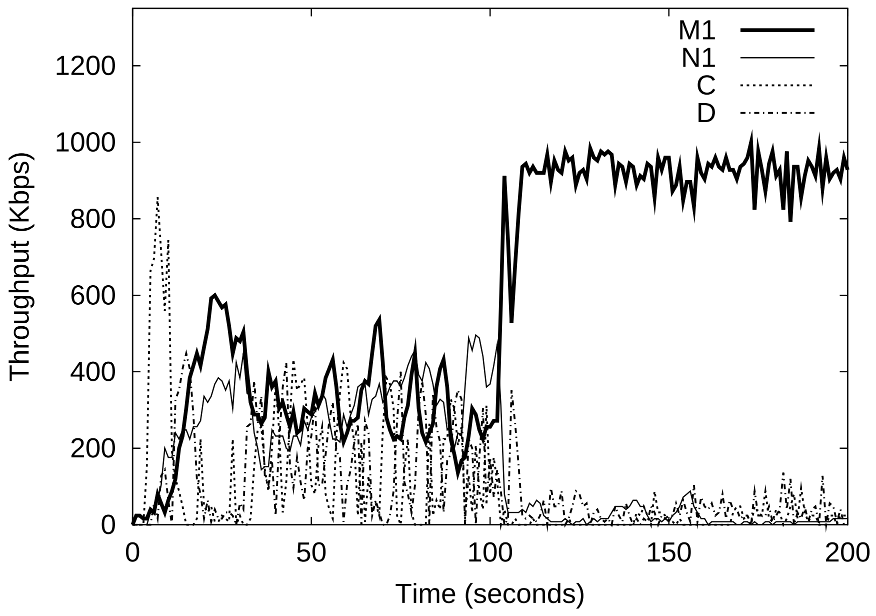
this verification would ward off attacks based on incorrect reports but could not protect against missing reports. Thus, failure to report can spring more potent attacks.

As in the experiment above, M_1 is the only misbehaver. After 100 seconds, M_1 does not report to the sender. Guided by reports from M_2, session M increases transmission to 1 Mbps and subdues the well-behaving cross traffic. Figure 2.5 presents throughput and losses for receivers C, D, N_1, and M_1.

2.2.3 Forged aggregated reports in RMTP

RMTP [39] is a reliable protocol that marks data packets with sequence numbers. Each receiver specifies lost packets in its feedback. RMTP designates some receivers to aggregate feedback from other receivers. Every designated receiver also retransmits lost packets to its children in the aggregation tree. To control congestion, the sender monitors the highest reported loss rate. If this loss rate exceeds a threshold, the sender cuts its transmission to a minimum. While the losses stay below the threshold, the transmission rate grows linearly.

A designated receiver can attack RMTP by failing to relay loss reports from its aggregation subtree. If the ignored reports belong to the slowest receivers, the sender boosts its transmission. In comparison to own distorted feedback, forged aggregated reports reward the misbehaver more and punish the others harsher. In the above attacks on TFMCC, the misbehaver can raise the transmission up to the fair rate for the slowest well-behaving receiver. This increase can be small. In the attack on RMTP, the fastest receiver can govern the transmission by quenching the reports from the slower receivers. Furthermore, the inflated transmission can



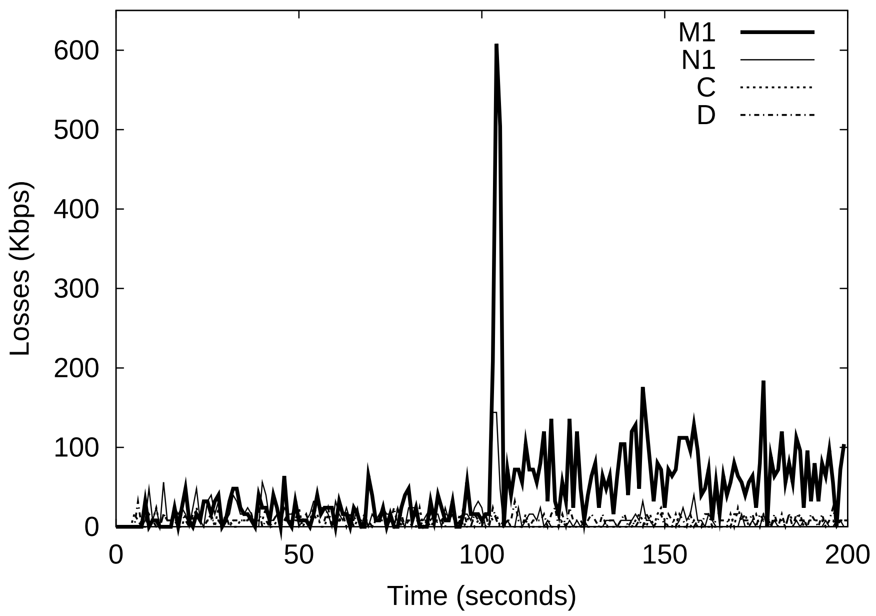


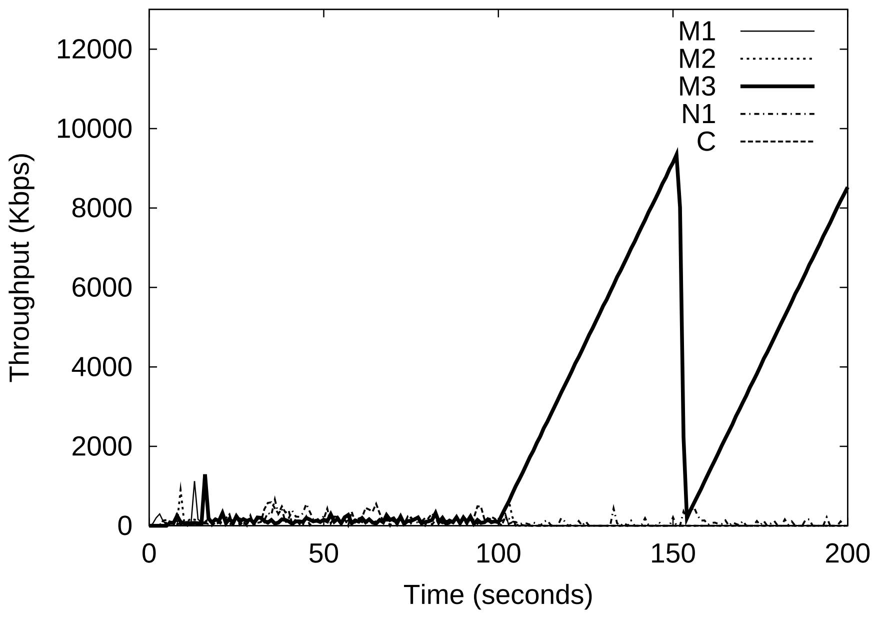
Figure 2.5: Failure to report in TFMCC

penalize the well-behaving receivers with heavy losses. To solidify the damage, the misbehaver can halt reliable delivery for the congested receivers by failing to retransmit the lost data.

We implemented RMTP following the description in [39]. In our experiment, designated receiver M_3 consolidates its feedback with reports from M_1 and M_2. The sender of M receives the aggregated feedback from M_3 and direct reports from M_4. In session N, both receivers report directly to the sender. After 100 seconds, M_3 ignores loss reports from M_1 and M_2. Figure 2.6 shows that this attack raises the transmission rate of M far above 1 Mbps and subdues well-behaving N_1 and C. Whereas M fills the 1 Mbps links with its data, M_1 and M_2 get skyrocketing losses and no throughput because M_3 does not retransmit lost packets to these receivers. Shortly after 150 seconds, the transmission rate of M saturates the 10 Mbps link, falls to a minimum upon a report from M_4, and then inflates again.

2.2.4 Manipulation with suppression in pgmcc

pgmcc [43] is a single-group protocol that employs two types of feedback: NAK and ACK. Based on NAK feedback, the sender picks a receiver to represent the session. This receiver is called an acker and ideally should have the smallest fair rate. Based on ACK feedback from the acker, the sender adjusts its transmission. To support reliable multicast, the sender retransmits lost packets and controls the retransmission rate by a separate mechanism. Upon detecting a packet loss, a receiver transmits a NAK report that includes the sequence number of the lost packet, loss rate, and so-called lead parameter used by the sender to calculate the fair rate for the receiver. To avoid implosion of NAK feedback, pgmcc relies on feedback



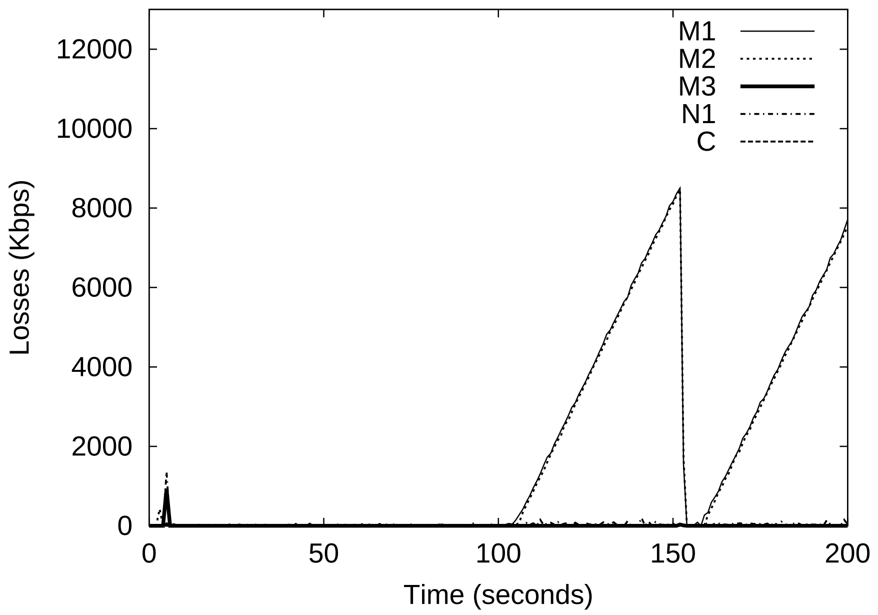
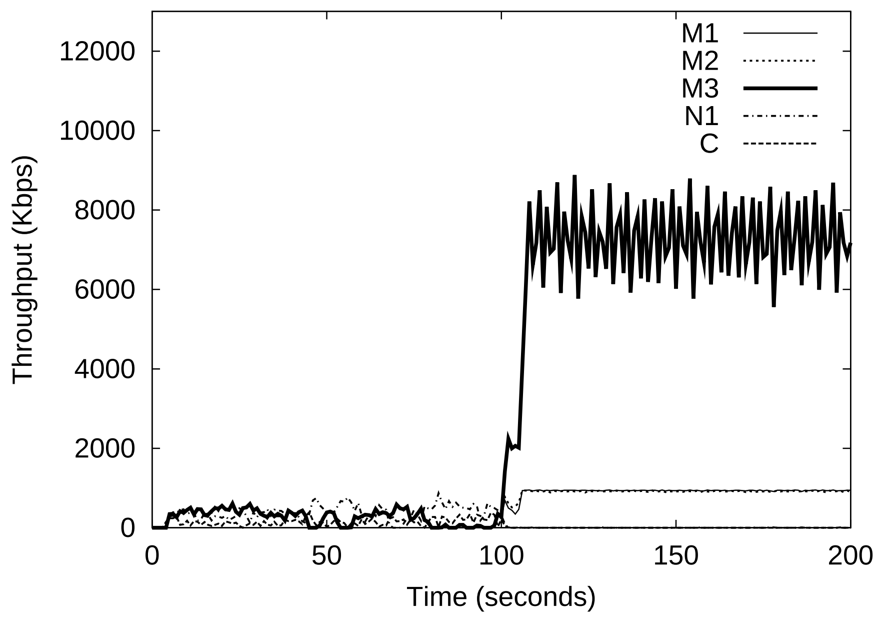


Figure 2.6: Forged aggregated reports in RMTP

suppression at PGM routers [48]. For each sequence number, a PGM router forwards the first NAK report containing this number and discards subsequent reports with the same number. Feedback suppression does not interfere with the acker selection because slower receivers experience higher losses and transmit NAK reports more frequently. Thus, reports with the smallest fair rate have a good chance to reach the sender without being suppressed. Also, feedback suppression is likely to filter out NAK feedback from more capable receivers and thereby exclude them from being considered for the acker position.

A misbehaving receiver can attack pgmcc by issuing a spurious NAK report. To avoid suppression, the spurious report carries an exaggerated sequence number. The report also distorts the loss rate and lead parameter to trick the sender into calculating a tiny fair rate and selecting the misbehaver as the acker. Since the sender identifies the acker in each data packet, the misbehaver knows whether the attack is successful. After the misbehaver becomes the acker, its ACK feedback boosts the transmission to a level that can greatly exceed the smallest fair rate. Unlike in the above attacks where a receiver must be the slowest or designated to benefit from misbehavior, any receiver in pgmcc can fraudulently become the acker and inflate the transmission. Hence, this attack offers the highest probability of success.

In our experiment, all routers suppress NAK feedback. Upon receiving a data packet after 100 seconds, M_3 misbehaves by transmitting a spurious NAK report with 99.99% loss rate, lead parameter of 1, and sequence number $s + 1000$ where s is the sequence number of the received packet. Since the sender is yet to transmit the packet requested in this NAK report, the report triggers no retransmission.



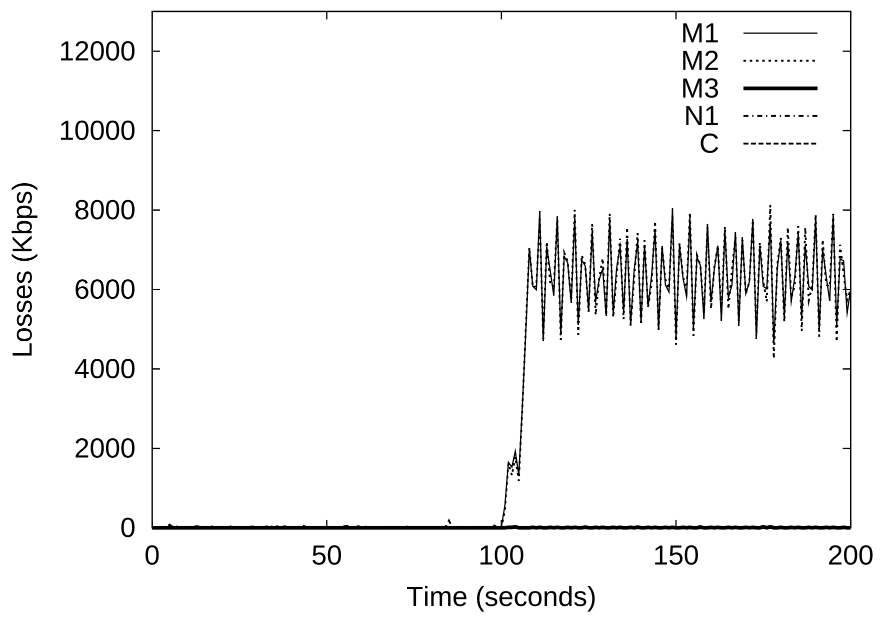


Figure 2.7: Manipulation with feedback suppression in pgmcc

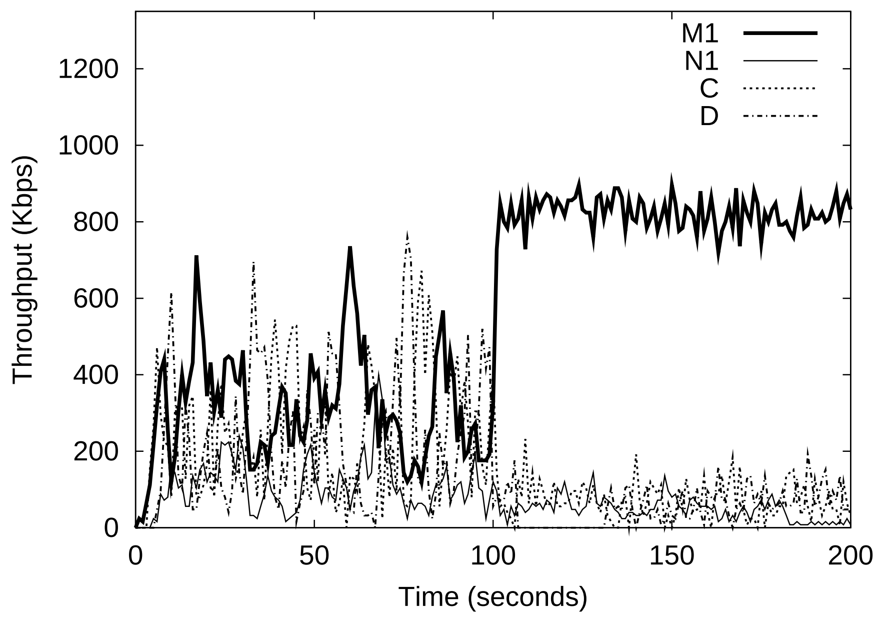
However, the spurious NAK reports establish M_3 as the acker, and its correct ACK feedback inflates the sending rate of M beyond 8 Mbps. Figure 2.7 shows that the inflated transmission stomps throughput of well-behaving N_1 and C to zero and causes huge losses for M_1 and M_2. Although M_1 and M_2 recover from the losses through retransmissions and maintain throughput of 1 Mbps, these receivers fall far behind M_3 in reliable acquisition of data.

2.2.5 Inflated subscription in FLID-DL

FLID-DL [7] is a multi-group feedback-free protocol where the sender encodes data into cumulative layers and uses a separate multicast group for each layer. Every receiver controls congestion by joining and leaving the groups of the session. Since the sender does not know the fair rates of the receivers, the default setting in FLID-DL uses a large number of multicast groups that cover – with a relatively fine granularity – the possible range of the fair rates.

To attack FLID-DL, a misbehaving receiver can join the layers with the cumulative transmission rate just below its bottleneck link capacity. The inflated subscription rewards the misbehaver with unfairly high throughput after the well-behaving cross traffic recedes. To detect the most beneficial subscription, the misbehaver can probe by adding a layer and keeping it only if the enhanced subscription does not cause persistent congestion.

Each FLID-DL session in our experiment has the same parameter settings: the base layer is transmitted at 24 Kbps; data are encoded into 24 layers; the cumulative transmission rate grows multiplicatively with the factor of 1.3 per layer. After 100 seconds, M_1 joins 14 lowest layers of session M, maintains this inflated



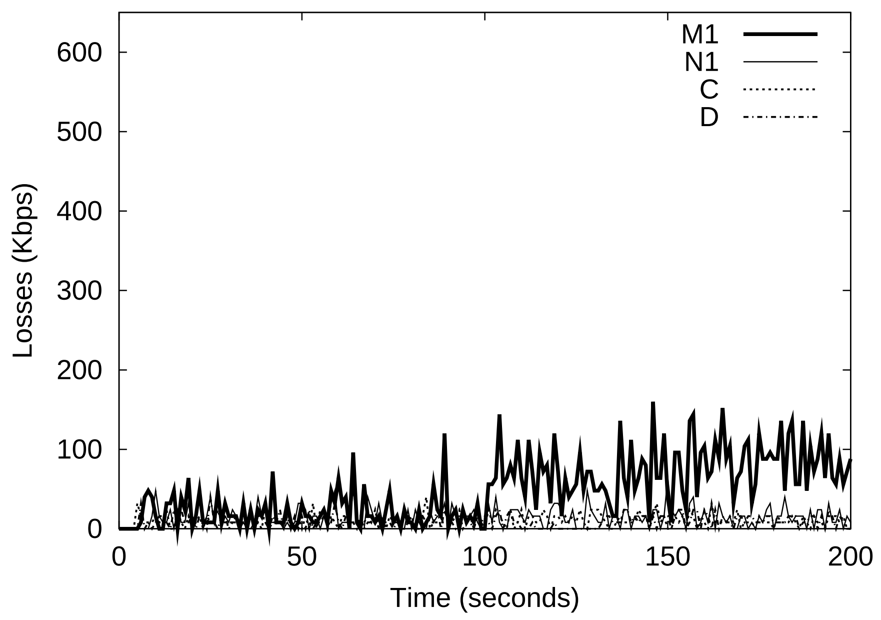


Figure 2.8: Inflated subscription in FLID-DL

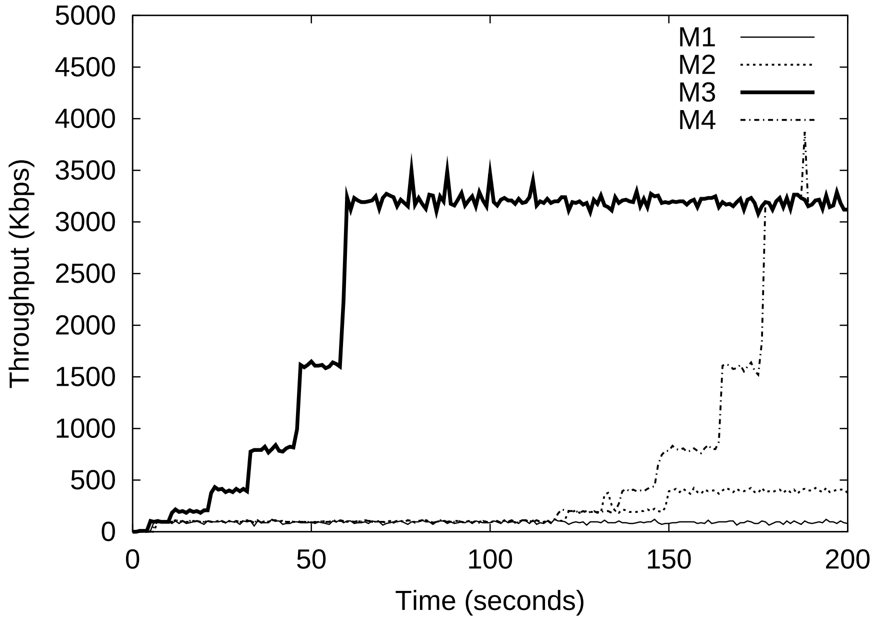
subscription, and raises its throughput to 800 Kbps. Figure 2.8 shows throughput and loss rates for receivers C, D, N_1, and M_1.

2.2.6 Prevention of other receivers from legitimate subscription in RLM

RLM [30] is also a multi-group feedback-free protocol where each group carries one layer of hierarchically encoded data. Every receiver maintains a join timer. When the join timer expires, the receiver adds the group that is immediately above its currently subscribed groups. To synchronize subscriptions, receivers rely on shared learning that sets the following rules:

- Before subscribing to a group, a receiver announces its intention to the other receivers.
- If a receiver observes losses shortly after subscribing to a group, the receiver drops the added group.
- When a receiver that is waiting to join a group receives a join announcement for a lower group, this receiver reschedules its join timer (to avoid derailing the announced join by the losses caused by its own join).

To attack RLM, a misbehaving receiver can periodically act as a newcomer. Its spurious announcements of joining the base layer prevent the other receivers from raising their subscriptions. Unlike the above attacks, this misbehavior gives the receiver an unfair edge over other receivers in the same session but not over receivers in a different session. Also, this attack succeeds only if it starts before the well-behaving receivers reach their fair subscriptions. However, the misbehaver



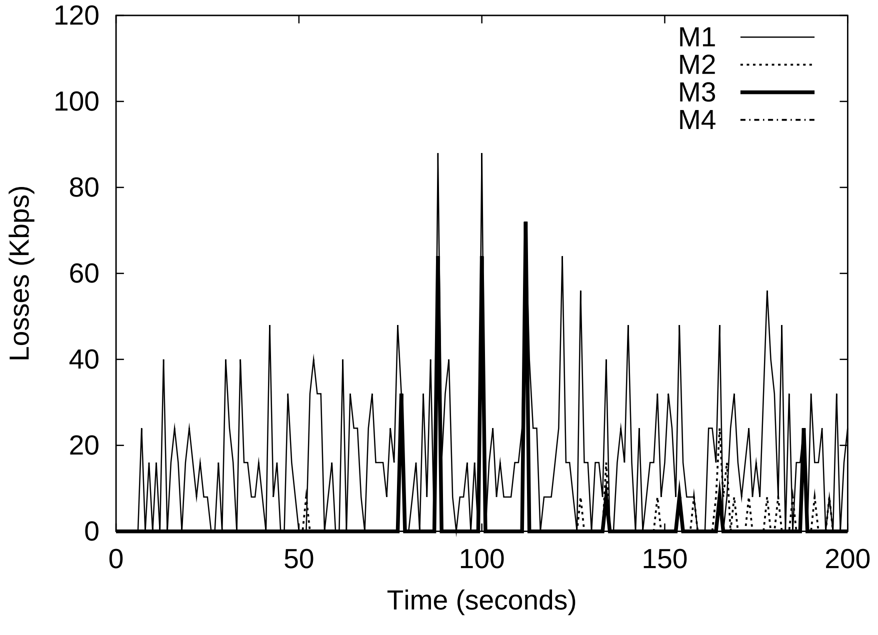


Figure 2.9: Prevention of other receivers from subscription in RLM

can deflate these subscriptions by inflating its own. After the auxiliary misbehavior causes congestion and subdues the other receivers, the misbehaver can keep their subscriptions low. Thus, the receiver can combine these two attacks to maximize its benefits.

In our experiment, each RLM session encodes data into 7 layers, transmits the base layer at 100 Kbps, and doubles the cumulative transmission rate with each layer. Every second until the midpoint of the experiment, M_3 issues a spurious announcement of joining the base layer and thereby limits the subscriptions of M_1, M_2, and M_4 to this layer. After 100 seconds, M_3 stops its attack and allows the other receivers of M to raise their subscriptions. Figure 2.9 presents throughput and loss rates for receivers M_1, M_2, M_3, and M_4.

2.3 Discussion

Section 2.2 shows that each threat in our model victimizes at least one existing multicast protocol. Moreover, we observed that all the protocols from Table 1.1 are vulnerable to receiver misbehavior. Following the threat ordering in our model, Table 2.1 classifies the vulnerabilities of these protocols. Below, we discuss our findings in more detail.

First, let us examine the congestion control paradigm of feedback-driven transmission adjustment. Among the protocols in Table 1.1, SAMM [50] is the only feedback-driven design where a misbehaving receiver does not benefit from its failure to report. In SAMM, the sender transmits all layers of hierarchically encoded data to a single group. Every receiver reports its rate of raw data reception and a count of 1. Feedback is aggregated at routers or auxiliary network devices. An aggregation

Mechanisms	Threats	Vulnerable protocols		
		Single-group	Feedback-free	Multi-group feedback-driven
Feedback generation	Incorrect reports	RMTP, SAMM, TFMCC, pgmcc		DSG, SIM, MLDA
	Failure to report	RMTP, TFMCC, pgmcc		DSG, SIM, MLDA
Feedback aggregation	Forged aggregated reports	RMTP		
Feedback suppression	Manipulation with suppression	pgmcc		
Group subscription	Inflated subscription		RLM, RLC, FLID-DL, WEBRC	DSG, SIM, MLDA
Subscription synchronization	Prevention from subscription		RLM	

Table 2.1: Vulnerabilities of multicast congestion control protocols

node reduces the number of reported rates to one per layer and enhances their counts with the counts of the ignored rates. The sender aligns its layer transmission rates with the reported rates.

The immunity of SAMM to failure to report comes from network support. All routers allocate the bandwidth of their links fairly among competing sessions and assign a larger forwarding priority to a lower layer within a SAMM session. At a bottleneck link, the SAMM session trims its rate to the fair share after the router discards the excessive higher layers. Whereas a misbehaving receiver cannot exceed its fair rate of raw data reception, feedback affects only the layer boundaries within this rate. Failure to report does not improve the alignment of the layer rates with the fair rate of the misbehaver. Hence, *the network can give receivers an incentive to supply feedback*.

Note however that SAMM is vulnerable to incorrect reports. By reporting inflated counts, a misbehaving receiver can elicit layer rates that match its capability exactly but are greatly unfair to other receivers in its session. Thus, *fair link scheduling is insufficient for comprehensive protection against multicast receiver misbehavior*.

Among the three protocols that use feedback aggregation, forged aggregated reports endanger only RMTP because SIM and SAMM aggregate feedback in the network. To protect receiver-based aggregation, a multicast protocol can employ feedback verification: if an aggregation node can detect that reports from its aggregation subtree are incorrect or incomplete, the protocol can curb the transmission to give receivers incentives to aggregate feedback properly.

Four protocols in Table 1.1 rely on feedback suppression but only pgmcc allows a misbehaving receiver to benefit from manipulating this mechanism. Both pgmcc and TFMCC employ feedback suppression to provide the sender with the smallest fair rate. A misbehaver can deceive both protocols by reporting an even smaller rate. In TFMCC where the sender adjusts its transmission to the smallest reported rate, the misbehaver does not benefit from the deception. On the other hand, pgmcc uses the smallest reported rate to select the acker, and the same deception rewards the misbehaver with the acker position and an opportunity to inflate the transmission through correct ACK feedback. Thus, *if protection against receiver misbehavior relies on feedback verification, even the feedback that affects transmission indirectly should be verified.*

Two challenges complicate feedback verification in multicast. First, feedback can be presented in a compressed form such as a rate or an average. Second, not only the sender but also other receivers and network devices can react to feedback.

Let us now consider the paradigm of group membership regulation. Among the multi-group protocols in Table 1.1, only RLM allows a misbehaving receiver to subdue the subscriptions of other receivers. The rest of the protocols is immune to the threat because in these protocols a receiver joins a group without consulting with other receivers. Since RLC, FLID-DL, and other advanced designs enable receivers to synchronize subscriptions without any exchange of messages between the receivers and without any support from the network, *subscription synchronization in future protocols has no reasons to rely on information supplied by receivers.*

All the multi-group protocols are vulnerable to the threat of inflated subscription because a misbehaving receiver is able to join any group. Thus, *protection against inflated subscription can rely on regulation of access to multicast groups.*

Above, we discussed vulnerabilities in different types of multicast congestion control protocols and offered some general suggestions for protecting the protocols against receiver misbehavior. From now on, we will concentrate on the class of feedback-free multicast protocols and derive more detailed designs for robust feedback-free congestion control.

Our interest in feedback-free protocols stems from their current predominance as the most promising approach to multicast congestion control. RLM, RLC, FLID-DL, and WEBRC have formed a successful line of designs leading to a practical solution ready for deployment. The main advantage of this design line is the simplicity of congestion control: since such a protocol provides no feedback from the receivers to the sender, the protocol does not need to incorporate a mechanism that aggregates or suppresses the feedback; instead, the protocol controls congestion via the group subscription mechanism, which IP multicast already supports to enable scalable formation of multicast groups.

Among the two threats that we identified for feedback-free multicast congestion control, prevention of other receivers from legitimate subscription is a vulnerability that has been addressed in advanced feedback-free designs (e.g., by the sender-driven synchronization of subscriptions in RLC and FLID-DL). Thus, the threat of inflated subscription remains the only obstacle for designing robust protocols for feedback-free congestion control. In the next chapter, we propose mechanisms that protect multicast protocols against inflated subscription.

Chapter 3

Robustness to Inflated Subscription

Chapter 2 identified inflated subscription of misbehaving multicast receivers as a major threat to fair bandwidth allocation. Below, we derive mechanisms that make multicast protocols robust to inflated subscription. We focus on preventive solutions and argue that prevention of inflated subscription attacks requires restricted group access. In Section 3.1, we show that existing architectures for group access control – such as Secure IGMP [3] and Gothic [25] – do not protect against inflated subscription because they define the eligibility to access a group based on the *identity*, rather than the *congestion status* of a receiver. With the current Internet infrastructure, inflated subscription belongs to the class of individual attacks because a receiver can join any multicast group at the local router without assistance from other parties. In Section 3.2, we examine these individual attacks and present DELTA and SIGMA, the first solution for the problem of inflated subscription. Our design uses dynamic keys to enforce *congestion-dependent* group access. DELTA and SIGMA require only minimal generic changes in the edge routers, do not alter the core of the network, and introduce no auxiliary servers. We then discuss properties of our scheme and demonstrate its vulnerability to collusion attacks where misbehaving receivers share group keys. Finally, Section 3.3 extends the design to make it robust to collusion attacks.

3.1 Design Requirements

Inflated subscription can be addressed by either discouraging the misbehavior or preventing it altogether. The former approach punishes misbehaving receivers *a posteriori*, e.g., by discriminatory dropping of their future packets [29]. In this dissertation, however, we focus only on mechanisms that *prevent* receivers from inflating their subscription.

Since IGMP does not restrict the ability of receivers to subscribe to multicast groups, a misbehaving receiver can join any multicast group as long as it knows the address of this group. Hence, a natural solution for preventing inflated subscription may appear to be the one that *hides* information about the groups (i.e., multicast group addresses) from ineligible receivers. Unfortunately, such information hiding is difficult to realize in modern networks: since multicast group addresses are employed for routing, receivers can abuse network monitoring and debugging tools – such as MSTAT [33] – to query routers and obtain the addresses of active multicast groups.

Based on these arguments, we conclude that to restrict group subscription only to eligible receivers, a multicast congestion control *must* regulate access to groups. Existing architectures for group access control – such as Secure IGMP [3] and Gothic [25] – rely on receiver authentication. Unfortunately, the identity of a receiver does not reveal any information about its congestion status. Hence, conventional group access control mechanisms are inadequate for preventing inflated subscription. Instead, multicast congestion control protocols need a mechanism where the congestion status of a receiver – rather than its identity – forms a foundation for group access control. This leads us to our first design requirement.

Requirement 1 *To protect against inflated subscription, multicast congestion control protocols must rely on congestion-dependent access control mechanisms.*

Since any form of group access control requires support from the network infrastructure (e.g., routers), deployment considerations lead us to the following requirement.

Requirement 2 *Implementation of access control mechanisms should require minimal modifications of the network infrastructure.*

The minimal infrastructure support requirement suggests that access control mechanisms should be implemented at edge routers without any changes in the network core. In addition to limiting the amount of infrastructure changes, it is essential for the access control functionality to be generic. The infrastructure should support a diverse collection of existing protocols as well as future protocols.

Requirement 3 *The access control functionality supported by the network infrastructure should be independent from details of specific congestion control protocols.*

Achieving the required generality of network support is challenging. As we showed in Section 1.2, different multi-group protocols specify different rules for group subscription. For instance, whereas a receiver in a replicated multicast session reacts to congestion by switching from its only subscribed group to a slower one, cumulative layered multicast protocols instruct a congested receiver to drop the top group among its currently subscribed groups. Furthermore, unlike some protocols that reduce subscription in response to a single packet loss, threshold-based protocols base subscription decisions on the observed loss rate. Also, while some protocols rely on packet loss as a congestion signal, others exploit ECN. The above considerations demonstrate that the right to access a group should be a *protocol-specific function of congestion.*

Finally, although our primary goal is to develop mechanisms that protect multicast congestion control protocols against inflated subscription, a secondary goal is to ensure that these mechanisms have minimal, if any, impact on the overall effectiveness of congestion control. This leads us to our final requirement.

Requirement 4 *Mechanisms for protecting against inflated subscription should preserve the scalability, fairness, efficiency, responsiveness, and other properties of multicast congestion control protocols.*

3.2 Protection against Individual Attacks

Our objective is to design group access control based on the congestion status of the receiver. Direct monitoring of congestion at routers is one option for congestion-dependent access control. For example, edge routers can observe the congestion status of a multicast session and enforce fair subscriptions of local receivers. How-

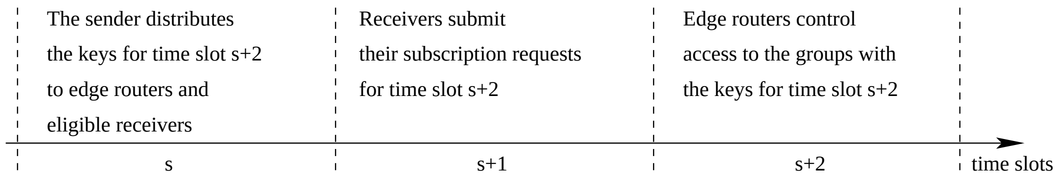

The sender distributes the keys for time slot s+2 to edge routers and eligible receivers
Receivers submit their subscription requests for time slot s+2
Edge routers control access to the groups with the keys for time slot s+2
s
s+1
s+2
time slots

Figure 3.1: Timeline for distribution and usage of keys

ever, such schemes violate our Requirement 3 because they make routers aware of the session, its groups, and its congestion control protocol. Hence, we select an alternative design where keys guard access to groups. To subscribe for a group, a receiver needs to provide a valid key to its local edge router. The edge router verifies the key prior to granting access to the group. The design requires edge routers to obtain, store, and validate group keys. This functionality, however, is independent of a specific congestion control protocol.

Since the network conditions change, keys for congestion-dependent group access should also be dynamic. We define a *time slot* as an atomic duration of group access control. The sender updates group keys once per time slot and distributes the updated keys to edge routers as well as to receivers that are eligible to access the groups during a subsequent time slot. Figure 3.1 depicts the timeline for key distribution and usage: the keys distributed during time slot s control access during time slot $s + 2$. Time slot $s + 1$ gives each receiver enough time to reconstruct the keys and submit them to the local edge router for validation before multicast packets from time slot $s + 2$ start reaching the router.

Since the eligibility to access a group depends on the congestion control protocol, distribution of keys to receivers is protocol-specific. Thus, we separate our design into two independent components: protocol-specific *DELTA (Distribution of ELigibility To Access)* – a method for in-band distribution of group keys to receivers that are eligible to access the groups according to the congestion control protocol, and generic *SIGMA (Secure Internet Group Management Architecture)* – an architecture for key-based group access at edge routers. Below, we present designs of these two components.

3.2.1 Design of DELTA

Despite differences in details, multi-group protocols share some general features. One common notion is a *subscription level* – a subset of the groups that constitutes a legitimate subscription in the session. Each protocol offers a finite number of subscription levels that can be ordered from a *minimal level* to a *maximal level* according to their bandwidth consumption. Although different protocols define the congested state of a receiver differently, they specify three generic rules for fair subscription: (1) *an uncongested receiver can maintain its current subscription level*, (2) *a congested receiver must decrease its subscription level*, and (3) *when authorized, an uncongested receiver can increase its subscription level.*

To enforce these subscription rules, DELTA distributes group keys among multicast packets so that:

1. Only an uncongested receiver can reconstruct updated keys for its current subscription level.

2. A congested receiver can obtain updated keys for a lower subscription level.

3. When authorized, an uncongested receiver can obtain updated keys for a higher subscription level.

Different protocols implement DELTA differently depending on their definitions for a subscription level and congested state. Below, we first present a DELTA instantiation for layered multicast protocols that define congestion as a single packet loss. Then, we discuss DELTA instantiations for other types of protocols.

Example of a DELTA Instantiation

FLID-DL [7] and RLC [49] are prominent representatives of unreliable protocols for cumulative layered multicast where congestion is defined as a single packet loss. In such a protocol, a session consists of multiple groups that carry layers of hierarchically encoded data. We label the groups in the order of their data layers: group 1 carries the base layer, group 2 carries the first enhancement layer, ..., and group N carries the last enhancement layer. Thus, group 1 constitutes the minimal subscription level in the session while the maximal subscription level consists of all N groups. We refer to groups 1 and N, respectively, as the minimal and the maximal groups of the session. The protocol specifies the following subscription rules: (1) *an uncongested receiver can keep its current groups*, (2) *a congested receiver of g groups must drop group g*, and (3) *when authorized, an uncongested receiver of g groups can add group $g + 1$*.

A straightforward transformation of these rules into conditions for in-band distribution of keys would introduce: (∇) a congested receiver of $g + 1$ groups should not obtain an updated key for group $g + 1$, and (Δ) when authorized, an uncongested receiver of g groups should obtain an updated key for group $g+1$. These requirements however contradict when group $g + 1$ is the only group that loses a packet, and group g gets an upgrade authorization: according to (∇), a subscriber to $g + 1$ groups should not obtain an updated key for group $g+1$; on the other hand, since groups 1 through g deliver all their packets, the subscriber should obtain this key according to (Δ). To resolve the contradiction, we allow such a receiver to get the updated key and maintain the subscription to group $g+1$. One can view this resolution as desirable because it helps receivers behind the same bottleneck link to synchronize their subscription levels. Thus, the conditions for key distribution become as follows:

1. An uncongested receiver should obtain updated keys for its current groups.
2. A congested receiver of g groups should obtain updated keys for its lower $g-1$ groups. It can obtain an updated key for group g only if the protocol authorizes an upgrade to group g, and groups 1 through $g-1$ do not lose packets.
3. When authorized, an uncongested receiver of g groups should obtain an updated key for group $g+1$.

Let us now derive a DELTA instantiation that satisfies these conditions. We start with an approach where a single key k_g guards access to group g.

In the absence of an upgrade authorization for group g, only an uncongested receiver of g groups should obtain k_g. To enforce this, the sender can attach a component $c_{j,p}$ to each packet p of every group j such that $1 \leq j \leq g$ and define k_g by applying a function F to the list of these components:

$$k_g = F(c_{1,1}, \ldots, c_{1,n_1}, \ldots\ldots, c_{g,1}, \ldots, c_{g,n_g}) \tag{3.1}$$

where n_j is the number of packets transmitted to group j during the time slot.

If the sender defines keys k_1 through k_N independently without reusing a component of one key as a component for another key, the communication overhead of the key distribution becomes high: each packet of group j needs to carry $N-j+1$ components (one component for each key k_g such that $j \leq g \leq N$). Thus, to minimize the per-packet overhead, each packet p from group j should carry only one component $c_{j,p}$ shared by keys k_j through k_N.

The sharing of components, however, complicates the fulfillment of the distribution conditions. For example, whereas all the components of key k_{g-1} are also components of key k_g, our second condition stipulates that a congested receiver of g groups should obtain k_{g-1} but not k_g. Then, such a receiver should not be able to obtain the shared components by applying the inverse of F to k_{g-1} because this ability would allow the congested receiver to reconstruct k_g via Equation 3.1 when group g does not lose a packet. Therefore, F should be a one-way function.

When the protocol authorizes an upgrade to group g, only an uncongested receiver of $g-1$ groups should obtain key k_g. Thus, a receiver should be able to compute k_g by applying a function H to a list of components $i_{j,p}$ distributed among all packets p of every group j such that $1 \leq j \leq g-1$:

$$k_g = H(i_{1,1}, \ldots, i_{1,n_1}, \ldots\ldots, i_{g-1,1}, \ldots, i_{g-1,n_{g-1}}). \tag{3.2}$$

Since the protocol can authorize an upgrade for each group 2 through N, minimizing the per-packet communication overhead requires that each packet p of group j contains only one component $i_{j,p}$ shared by all keys k_{j+1} through k_N. Then, each component $i_{j,p}$ of key k_{g-1} is also a component of key k_g. Once again, a congested receiver of g groups should not be able to obtain these shared components $i_{j,p}$ by applying the inverse of H to k_{g-1} because this ability would allow the receiver to reconstruct k_g via Equation 3.2 when only groups 1 through $g-2$ lose packets. Hence, H should also be a one-way function.

	Keys	
	with upgrade authorization	without upgrade authorization
Maximal group	$\boxed{\tau}$ $\boxed{\sigma}$	$\boxed{\tau}$
Each intermediate group	$\boxed{\tau}$ $\boxed{\delta}$ $\boxed{\sigma}$	$\boxed{\tau}$ $\boxed{\delta}$
Minimal group	$\boxed{\tau}$ $\boxed{\delta}$	$\boxed{\tau}$ $\boxed{\delta}$

top key: $\boxed{\tau}$ decrease key: $\boxed{\delta}$ increase key: $\boxed{\sigma}$

Table 3.1: Group keys in the presented DELTA instantiations

To reconcile Equations 3.1 and 3.2, a scheme that generates keys and their components must resolve functions F and H into the same value k_g. Since both F and H are one-way functions, no practical algorithm can achieve this goal.

This conclusion leads us to an idea of guarding a group with multiple keys such that any of these keys opens access to the group. Having more than one key per group enables simple instantiations of DELTA by relaxing the dependencies between the distribution conditions.

We instantiate DELTA by using up to three keys per group: (1) top key, (2) decrease key, and (3) increase key (see Table 3.1). The sender communicates these keys to receivers by adding component fields and decrease fields to multicast packets. We define top key τ_g for each group g as:

$$\tau_g = \bigoplus_{j=1}^{g} \bigoplus_{p \in S_j} c_{j,p} \tag{3.3}$$

where $\oplus$ is an XOR operation, S_j is a set of packets sent to group j, and $c_{j,p}$ is a nonce placed by the sender into the component field of packet p from group j. Only an uncongested receiver of g groups can extract all nonces $c_{g,p}$ and use Equation 3.3 to compute key τ_g.

For each group j such that $1 \leq j \leq N-1$, the sender generates the following decrease key δ_j:

$$\delta_j = d_{j+1} \tag{3.4}$$

where d_{j+1} is a nonce put into decrease field of each packet transmitted to group $j+1$. A receiver of g groups can compute keys δ_1 through δ_{g-1} via Equation 3.4 as long as the receiver gets at least one packet from each group 2 through g. If one of these groups loses all its packets, the receiver is forced to reduce its subscription by more than one group. In fact, if group g loses all its packets, and any group 1 through $g-2$ loses a packet, no in-band mechanism can provide a receiver of g groups with an updated key for group $g-1$ without violating the other distribution conditions.

When the protocol authorizes an upgrade to group m where $2 \leq m \leq N$, the sender generates increase key σ_m for group m as:

$$\sigma_m = \bigoplus_{j=1}^{m-1} \bigoplus_{p \in S_j} c_{j,p} \tag{3.5}$$

and thereby enables an uncongested receiver of g groups to reconstruct key σ_{g+1} after receiving all components $c_{j,p}$ from groups 1 through g.

Figures 3.2 and 3.3 present our algorithm for the in-band distribution of keys to receivers. The algorithm has a nice property that the sender precomputes the keys without knowing the number of transmitted packets and then generates components of the keys in real time. Thus, adopting the DELTA instantiation does not change the packet transmission pattern. Besides, the precomputation of the keys allows SIGMA to distribute them to edge routers beforehand.

Instantiations for Other Types of Protocols

The derived DELTA instantiation protects FLID-DL, RLC, and similar unreliable protocols for cumulative layered multicast where congestion is defined as a single

Input	N	number of groups in the session
	S_g	set of packets for group g where $1 \leq g \leq N$
	l_g	last packet for group g where $1 \leq g \leq N$

Algorithm

```
// precomputation of keys and decrease fields
 for  $g = 1, \ldots, N$
         $C_g \leftarrow$ nonce;
 $\tau_1 \leftarrow C_1$;
 for  $g = 2, \ldots, N$
         $\tau_g \leftarrow \tau_{g-1} \oplus C_g$; $\delta_{g-1} \leftarrow$ nonce; $d_g \leftarrow \delta_{g-1}$;
         if  the protocol authorizes an upgrade to group $g$
              then  $\sigma_g \leftarrow \tau_{g-1}$;
// real-time generation of component fields
 if  $p \in S_g$ and $p \neq l_g$  then  $c_{g,p} \leftarrow$ nonce; $C_g \leftarrow C_g \oplus c_{g,p}$;
 if  $p = l_g$  then  $c_{g,p} \leftarrow C_g$.
```

Output	$c_{g,p}$	component field for packet p in S_g where $1 \leq g \leq N$
	d_g	decrease field for group g where $2 \leq g \leq N$
	τ_g	top key for group g where $1 \leq g \leq N$
	δ_g	decrease key for group g where $1 \leq g \leq N-1$
	σ_g	increase key for group g where $2 \leq g \leq N$

Figure 3.2: The sender algorithm in our DELTA instantiation for layered multicast protocols that define congestion as a single packet loss

packet loss. To protect protocols of other types, we extend the presented approach along the four dimensions discussed in Section 1.2: (1) reliability, (2) congestion notification, (3) session structure, and (4) congested state.

Reliability. Reliable multicast protocols overcome losses by transmitting additional packets, e.g., packets with retransmitted data or error correction codes. If a reliable protocol includes these extra packets in its definition for a congested state, DELTA protects the protocol by distributing the keys among both the original and added packets.

Congestion notification. Instantiations of DELTA for loss-driven congestion control can be easily adapted for networks where routers support ECN and

Input	g	current top group
	R_j	set of packets received from group j where $1 \leq j \leq g$

Algorithm
```
for $j = 2, \ldots, g$
        $u_{j-1} \leftarrow$ decrease field from $R_j$;
if the receiver is congested
    then if $g = 1$
            then $n \leftarrow$ null;
            else $n \leftarrow g - 1$;
    else $u_g \leftarrow \bigoplus_{j=1}^{g} \bigoplus_{r \in R_j}$ component field from $r$;
            if the protocol authorizes an upgrade to group $g + 1$
                then $n \leftarrow g + 1$; $u_{g+1} \leftarrow u_g$;
                else $n \leftarrow g$.
```

Output	n	next top group
	u_j	updated key for group j where $1 \leq j \leq n$

Figure 3.3: The receiver algorithm in our DELTA instantiation for layered multicast protocols that define congestion as a single packet loss

mark forwarded packets to indicate congestion explicitly. To extend the protection to ECN-driven multicast protocols, edge routers simply alter the content of the component field in each marked packet. This prevents receivers ineligible for a group from reconstructing the group key.

Session structure. Unlike in layered multicast, each subscription level in a replicated multicast session consists of a single group and provides the same content but at a different rate: minimal group 1 is the slowest, group 2 is the second slowest, ..., and maximal group N is the fastest. Let us now consider a replicated multicast protocol that differs from the discussed protocol for layered multicast only with respect to the subscription rules: (1) *only an uncongested receiver can stay in its current group*, (2) *a congested receiver of group g can switch to group $g-1$*, and (3) *when authorized, an uncongested receiver of group g can switch to group $g+1$*.

Note that the rules allow an uncongested receiver to stay in its current group g even if the protocol authorizes an upgrade to group $g+1$. Then, the receiver can subscribe to both groups. However, the receiver does not benefit from such misbehavior because group g delivers the same content but at a lower quality than group $g+1$. Since our objectives are limited to achieving robustness against self-beneficial attacks, we formulate conditions for the key distribution as follows:

1. Only an uncongested receiver should obtain an updated key for its current group.
2. A congested receiver of group g should obtain an updated key for group $g-1$.
3. When authorized, an uncongested receiver of group g should obtain an updated key for group $g+1$.

We fulfill the conditions with a DELTA instantiation presented in Figures 3.4 and 3.5. The algorithm is basically the same as for layered multicast: the sender computes up to three keys per group and communicates the keys to receivers via component and decrease fields. However, since each subscription level in replicated multicast contains only one group, we redefine top and increase keys for group g as:

$$\tau_g = \bigoplus_{p \in S_g} c_{g,p} \quad \text{and} \quad \sigma_g = \bigoplus_{p \in S_{g-1}} c_{g-1,p}, \tag{3.6}$$

i.e., in terms of components from a single group.

Congested state. Multicast protocols often ignore occasional loss of packets and consider a receiver to be congested only when its loss rate exceeds a threshold. For extending the protection to threshold-based protocols, DELTA can use Shamir's (k,n) threshold scheme [46] to distribute components of key τ_g for subscription level g among all n packets transmitted to this level – the sender uses modular

Input	N	number of groups in the session
	S_g	set of packets for group g where $1 \leq g \leq N$
	l_g	last packet for group g where $1 \leq g \leq N$

```
Algorithm   // *precomputation of keys and* decrease *fields*
             for $g = 1, \ldots, N$
                 $C_g \leftarrow$ nonce; $\tau_g \leftarrow C_g$;
             for $g = 2, \ldots, N$
                 $\delta_{g-1} \leftarrow$ nonce; $d_g \leftarrow \delta_{g-1}$;
                 if the protocol authorizes an upgrade to group $g$
                     then $\sigma_g \leftarrow \tau_{g-1}$;
            // *real-time generation of* component *fields*
             if $p \in S_g$ and $p \neq l_g$ then $c_{g,p} \leftarrow$ nonce; $C_g \leftarrow C_g \oplus c_{g,p}$;
             if $p = l_g$ then $c_{g,p} \leftarrow C_g$.
```

Output	$c_{g,p}$	component field for packet p in S_g where $1 \leq g \leq N$
	d_g	decrease field for group g where $2 \leq g \leq N$
	τ_g	top key for group g where $1 \leq g \leq N$
	δ_g	decrease key for group g where $1 \leq g \leq N-1$
	σ_g	increase key for group g where $2 \leq g \leq N$

Figure 3.4: The sender algorithm in our DELTA instantiation for replicated multicast protocols

arithmetic, picks a polynomial $q(x)$ of degree $k-1$:

$$q(x) = \tau_g + a_1 x + \ldots + a_{k-1} x^{k-1}, \tag{3.7}$$

where $a_1, \ldots, a_{k-1}$ are random coefficients, and puts one component c_p into each packet p:

$$c_p = (p, q(p)) \tag{3.8}$$

where $p = 1, \ldots, n$. Only a receiver that obtains at least k out of the n packets can find the coefficients of $q(x)$ by interpolation and then reconstruct the key as:

$$\tau_g = q(0). \tag{3.9}$$

Input	g	current group
	R_g	set of packets received from group g

Algorithm

```
if  the receiver is congested
    then  if  $g = 1$
              then  $n \leftarrow$ null;
              else  $n \leftarrow g - 1$;
                    $u_{g-1} \leftarrow$ decrease field from $R_g$;
    else  $u_g \leftarrow \bigoplus_{r \in R_g}$ component field from $r$;
          if  the protocol authorizes an upgrade to group $g + 1$
              then  $n \leftarrow g + 1$; $u_{g+1} \leftarrow u_g$;
              else  $n \leftarrow g$.
```

Output	n	next group
	u_j	updated key for group n

Figure 3.5: The receiver algorithm in our DELTA instantiation for replicated multicast protocols

In layered multicast, subscription levels share groups. Unfortunately, Shamir's scheme does not enable a reuse of the components from lower subscription levels. The reliance on independent components is a potential source of a prohibitively high communication overhead. Design of more efficient threshold schemes that reuse components remains an open research problem.

3.2.2 Design of SIGMA

Whereas instantiating DELTA enables multicast protocols to distribute group keys to eligible receivers, group access control also needs a mechanism for distributing the keys to edge routers. As per Requirement 3 from Section 3.1, the functionality of edge routers should not depend on details of congestion control protocols. In particular, edge routers should run protocol-independent code to obtain and store keys as well as to enforce appropriate group access. In this section, we present

SIGMA (Secure Internet Group Management Architecture) – a generic architecture for key-based group access control at edge routers. Below, we discuss the two tasks of SIGMA: (1) distribution of keys to edge routers, and (2) multicast group management.

Generic Distribution of Keys to Edge Routers

Our threat model assumes that the network infrastructure is trustworthy and always adheres to protocols. SIGMA exploits this assumption for distributing keys to edge routers via special multicast packets where tuples bind the address of each group with the keys for accessing the group during a time slot. For example, when the layered multicast protocol described in Section 3.2.1 does not authorize an upgrade to an intermediate group g, SIGMA communicates a tuple that links the address of group g with top key τ_g and decrease key δ_g; if the protocol authorizes the upgrade, the tuple for group g also contains increase key σ_g. The network-layer headers of the special packets carry a bit instructing edge routers to intercept the packets without forwarding to local interfaces. Edge routers run the same protocol-independent code for intercepting the special packets and storing the address-key tuples. To ensure reliable delivery of the addresses and keys to edge routers, SIGMA uses forward error correction.

Multicast Group Management

Multicast group management in SIGMA is challenging because keys change every time slot. When a receiver proves its right to join a new group, some time may pass before the network starts forwarding packets from the added group to the local edge router. Besides, after the packets start reaching the receiver, their first complete time slot s can enable the receiver to obtain the group key for time slot $s+2$ but not for time slot $s+1$ (see Figure 3.1). To allow an uncongested receiver to maintain its uninterrupted subscription to the new group, the edge router marks the local interface as expecting the group. After packets from the added group start reaching the edge router, the router forwards them to the interface unconditionally for two complete time slots.

Admission of a new receiver into a session is a challenge because DELTA provides updated keys only to current receivers. SIGMA admits new receivers by allowing a receiver to join the minimal group without a key: the receiver simply sends the local edge router a session-join message that contains the address of the minimal group (see Figure 3.6(a)). However, if the new receiver fails to submit a valid key within two complete time slots of unrestricted access to the minimal group, the edge router stops forwarding the group packets for at least one time slot. This prevents a receiver ineligible even for the minimal group from maintaining an uninterrupted subscription to the session.

With respect to the other group management functions, SIGMA resembles existing schemes for key-based group access control. In what follows, we describe how SIGMA implements these functions.

Subscribing to a group. A receiver subscribes to a group for a time slot by sending the local edge router a subscription message that specifies the time slot and address-key pair for the group. Before granting access to the requested group, the edge router verifies validity of the submitted key. Figure 3.6(b) shows the general format of subscription messages. To ensure reliable subscription, the edge

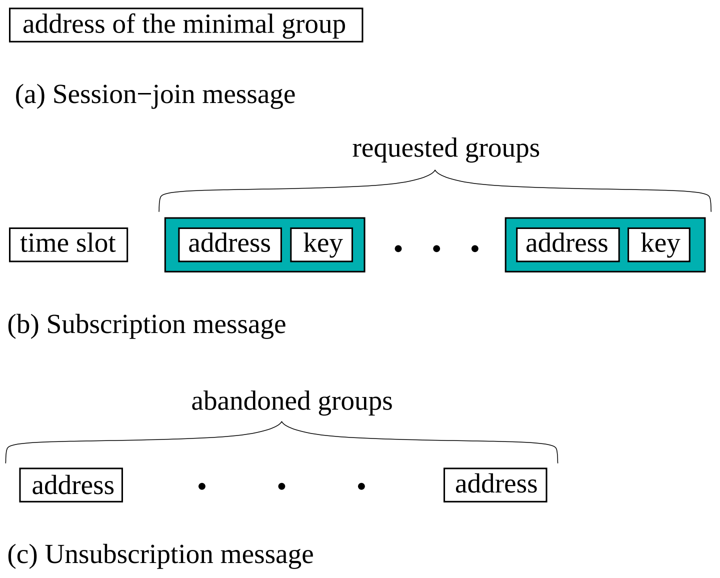


Figure 3.6: SIGMA messages sent by receivers

router acknowledges each subscription message. If a receiver does not receive an acknowledgment for its subscription message, the receiver retransmits the message. To reduce traffic on the local interface, a receiver does not transmit its subscription message if the edge router has acknowledged an earlier message from another receiver that reported the same address-key pairs.

Unsubscribing from a group. Dynamic keys ensure that failure to provide a valid key for a group results in leaving the group. For example, a congested receiver is forced to drop a group within two time slots after congestion. To allow a receiver – e.g., an uncongested receiver parting with its session altogether – to leave groups even quicker, SIGMA also offers an explicit unsubscription message that contains the addresses of the abandoned groups (see Figure 3.6(c)). When a receiver leaves a group, its unsubscription message should not harm other receivers subscribed to the group legitimately on the same interface. The remaining receivers preserve the group subscription by submitting a subscription message that supplies a valid key for the group.

Incremental Deployment of SIGMA

Each edge router that replaces IGMP with SIGMA notifies local receivers about its support of SIGMA. If an edge router does not support SIGMA, local receivers of a multicast session protected with DELTA and SIGMA interact with the router via IGMP and ignore DELTA packet fields and SIGMA special packets. Such receivers still can inflate their subscription and acquire unfairly high bandwidth. However, even a partial deployment of SIGMA edge routers is beneficial – these routers prevent their local receivers from inflated subscription.

3.2.3 Properties of DELTA and SIGMA

Revisiting the Design Requirements

We start an assessment of our design by revisiting the requirements from Section 3.1 and arguing that DELTA and SIGMA meet these requirements.

Congestion-dependent group access control. While SIGMA guards access to groups with dynamic keys, DELTA distributes the keys only to receivers that are eligible to access the groups according to the congestion control protocol. DELTA instantiations protect protocols of different types: unreliable and reliable, loss-driven and ECN-driven, layered and replicated, reacting to a single loss and based on a threshold for the loss rate.

Minimal changes in the network. Any architecture for key-based group access control must enable edge routers to obtain and store keys as well as to enforce appropriate group access. SIGMA adds only this minimal required functionality into edge routers. Furthermore, DELTA and SIGMA need no support from core routers and introduce no servers into the network infrastructure.

Generality of network support. To support DELTA and SIGMA, edge routers run code that is independent from details of protected congestion control protocols.

Preservation of congestion control properties. The presented algorithms impose no limitations on packet transmission. The sender precomputes group keys and then generates their components in real time. Consequently, adopting DELTA does not require from a protocol to change its transmission pattern. In Chapter 4, we experimentally verify that DELTA and SIGMA also preserve other congestion control properties of protected protocols.

Security Properties

In this section, we discuss security properties of the protection offered by DELTA and SIGMA.

Maintaining the trusted base. Our design assumes that the network infrastructure is trustworthy. DELTA and SIGMA implementations can realize this assumption by using conventional techniques for: (a) authentication to prevent a misbehaving receiver from posing either as a sender or as a router [41, 52], and (b) hop-by-hop or edge-to-edge encryption to protect against snooping on network links [31].

Protection against attacks on SIGMA. As long as a local interface provides an edge router with a valid key for a group, the router forwards packets of the group to the interface. A misbehaving receiver ineligible to access the group can send the edge router numerous random keys in a hope that one of these keys is correct. If a valid key consists of b bits, the probability to gain the group access by guessing the key is $y/2^b$ where y is the number of address-key pairs that the receiver is capable of communicating to the edge router for the time slot. To address this attack, the edge router can count different keys submitted for the group and interpret a large tally as a possible indicator of the attack.

Protection against attacks on DELTA. To acquire a forbidden key, a receiver can seek vulnerabilities in the DELTA implementation. For example, the receiver can attempt to guess a missing component of the key. In the DELTA instantiations presented in Section 3.2.1, keys and components have the same size, and the component guessing gives no advantage over guessing the key directly.

So far in this chapter, we treated inflated subscription as belonging to the category of individual attacks because a multicast receiver in the current Internet infrastructure can inflate subscription by abusing IGMP without any assistance from other receivers. DELTA and SIGMA protect multicast protocols against these individual attacks: SIGMA guards access to groups with keys, and DELTA distributes the keys only to receivers that are eligible to access the groups according to the congestion control protocol. However, our design is vulnerable to collusion attacks where receivers of a multicast session conspire to pass keys from a more capable receiver to a less capable receiver that is unable to reconstruct these keys on its own. Below, we extend DELTA and SIGMA to make the design robust to inflated subscription of colluding receivers.

3.3 Robustness to Collusion Attacks

The above design of DELTA and SIGMA assumes that a receiver can obtain a group key only from DELTA fields of multicast packets delivered to the receiver. This assumption does not hold in the presence of colluding receivers because a receiver can also acquire keys or their components from more capable receivers. Thus, achieving robustness against such collusion attacks requires from each local interface to guard a group with a different set of keys.

Extending the design to support interface-specific keys while continuing to fulfill the requirements from Section 3.1 is a serious challenge. To be as scalable as our original design, its extension should not put the sender in charge of generating all the interface-specific keys. Then, the network infrastructure (i.e., routers) should absorb the burden of creating the interface-specific keys. However, generation of group keys at routers makes the router functionality protocol-specific and thereby violates the condition of minimal generic support from the network. Therefore, extension of the protection against inflated subscription to the class of collusion attacks is a subject to a trade-off between degrading the design scalability and diluting the genericness of required network support. In what follows, we explore this trade-off by outlining two design extensions: whereas the first approach maintains the scalability at the expense of the genericness, the second solution preserves the genericness by limiting the scalability.

Let us start with the extension that preserves the scalability of the original design. In DELTA and SIGMA, the sender runs a randomized algorithm R to generate a group key and its components. To communicate the key to eligible receivers, the sender puts the components of the key into DELTA fields of multicast packets. For example, Section 3.2.1 presented the DELTA instantiation for layered multicast protocols where a top key is picked randomly and split – using an XOR operation – into random component fields attached to all the packets of the current subscription level; also, each packet of an upper group carries the same random decrease field that contains a decrease key for the group below.

To make group keys interface-specific, the sender can relegate the computation of the keys and their components to the edge routers. Instead of using SIGMA to provide the edge routers with the same key k for group g, the sender communicates via SIGMA a description of algorithm R and its parameters, e.g., the addresses of groups that are supposed to carry the components of key k as well as the number of packets the sender has transmitted to each of these groups during the time slot. Then, for every local interface i, the edge router runs the randomized algorithm R to create key k_i for group g, generate components of k_i, and insert them as DELTA fields into forwarded multicast packets. The edge router does not have to generate components in advance for each packet transmitted by the sender. Instead, the generation of the components can be done in real time, when and if the packet reaches the edge router.

The functionality of receivers does not change – if the DELTA fields of delivered multicast packets provide enough components, the receiver reconstructs the key and submits it to the local edge router for validation. If a missing component prevents a congested receiver from reconstructing the key on its own, the receiver cannot obtain the key or its missing component from more capable receivers because those receivers are subscribed to the session through different interfaces that guard access to the group with independently selected keys. Thus, the generation of personalized keys at the edge routers protects multicast congestion control protocols against inflated subscription of colluding receivers.

The protection, however, comes at a price of additional network support. In comparison to the original design where the edge routers run the same simple code to extract group keys from SIGMA packets, the edge routers in the extended design perform more diverse actions (including creation of keys, generation of their components, and insertion of the components into forwarded multicast packets) using different algorithms for multicast congestion control protocols of different types. Thus, the edge routers run a more complex parameterized code where the parameters of the randomized algorithm are supplied by the sender via SIGMA. These parameters can be classified into two categories:

1. Parameters characterizing the type of the multicast congestion control protocol along such dimensions as the session structure (layered, replicated, ...), congestion notification (loss-driven, ECN-driven, ...), and congested state definition (single packet loss, loss rate threshold, ...). Parameters from this first category instruct the edge router about the algorithm to be used for generating keys and their components.

2. Parameters that characterize the specific multicast session of the protocol, e.g., the addresses of groups comprising the session, the number of packets transmitted to each of these groups during a particular time slot, and the threshold for the tolerable loss rate.

Together with the size, the generality of the parameterized code is a concern. Even if the code supports all the existing types of multicast protocols, there remains a possibility that a new protocol type will emerge in the future and require a different algorithm for generating group keys and their components. Then, supporting this new algorithm will require changes in edge routers. Hence, the outlined design extension provides robustness against collusion attacks by weakening the genericness of needed network support.

Let us now derive an alternative extension that preserves the minimal generic router functionality of DELTA and SIGMA. The original design limits the required network support to the following tasks at edge routers:

1. Generic change of DELTA fields in forwarded packets (used for protection of ECN-driven protocols).

2. Generic reception (via SIGMA), storage, and application of group keys.

To protect against collusion attacks without requiring any support beyond these two functions, edge routers can apply interface-specific mask m_i to the DELTA fields of all packets forwarded to local interface i:

$$f_i = G(f, m_i) \tag{3.10}$$

where f is a value put into a DELTA field by the sender, f_i is the content of the same DELTA field after the edge router modifies the field, and G is a one-way function that prevents receivers of the modified packet from determining f or m_i. Due to the masking, received packets contain interface-specific values in their DELTA fields. Consequently, applying the key reconstruction algorithm to these values produces interface-specific keys. If a receiver cannot reconstruct its interface-specific key for a group because of congestion, the receiver cannot access the group even if uncongested receivers from other interfaces reveal their keys for the same group (because the revealed keys are valid only for the corresponding interfaces).

Hence, the masking provides robustness against inflated subscription of colluding receivers.

Similarly to the original design, the involvement of edge routers into group access enforcement should be limited to generic reception, storage, and application of keys. Then, generation of all the interface-specific keys should be done at the sender. To compute these keys, the sender also needs to know the masks associated with each edge router. Thus, the sender in the extended design generates all the interface-specific masks and keys and communicates them to the edge routers. Whereas the masks can be updated relatively infrequently without jeopardizing the security of the offered protection, the sender should provide the edge routers with new sets of keys once per time slot. This limits the scalability of the masking-based extension to DELTA and SIGMA.

In this section, we demonstrated that extending DELTA and SIGMA to be robust against collusion attacks is subject to a fundamental trade-off between degrading the scalability and diluting the genericness of required router support. Achieving an optimal balance between the scalability and the amount of network support is a topic for future research.

Chapter 4

Derivation of Robust Multicast Protocols

In Chapter 3, we designed mechanisms for achieving robustness against inflated subscription. By incorporating these mechanisms, feedback-free protocols for multicast congestion control acquire immunity to receiver misbehavior. Below, we derive and evaluate robust versions of two prominent feedback-free protocols: FLID-DL and RLM. Whereas RLM is an example of threshold-based congestion control schemes, FLID-DL represents protocols that define congestion as a single packet loss.

4.1 Robust Adaption of FLID-DL

4.1.1 Protocol Description

Fair Layered Increase/Decrease with Dynamic Layering (FLID-DL) [7] is the third protocol in the series of feedback-free designs for multicast congestion control. Similarly to its predecessors RLM and RLC, the protocol is unreliable and uses multiple multicast groups to distribute cumulative layers of hierarchically encoded data. Each receiver controls congestion by joining and leaving the data layers. FLID-DL offers the receivers a relatively fine granularity of bandwidth allocation by employing, in the default setting, a large number of layers such that each of the enhancement layers consumes a small amount of the network bandwidth. The protocol defines congestion as a single packet loss and instructs a congested receiver to reduce its subscription level by dropping the top subscribed layer. An uncongested receiver is allowed to increase its subscription level (by adding the layer that is immediately above the top subscribed layer) only upon an explicit signal from the sender. Higher layers carry increase signals less frequently. By selecting the frequencies of the increase signals appropriately, FLID-DL enables each receiver to converge to its fair subscription level without assistance from other receivers.

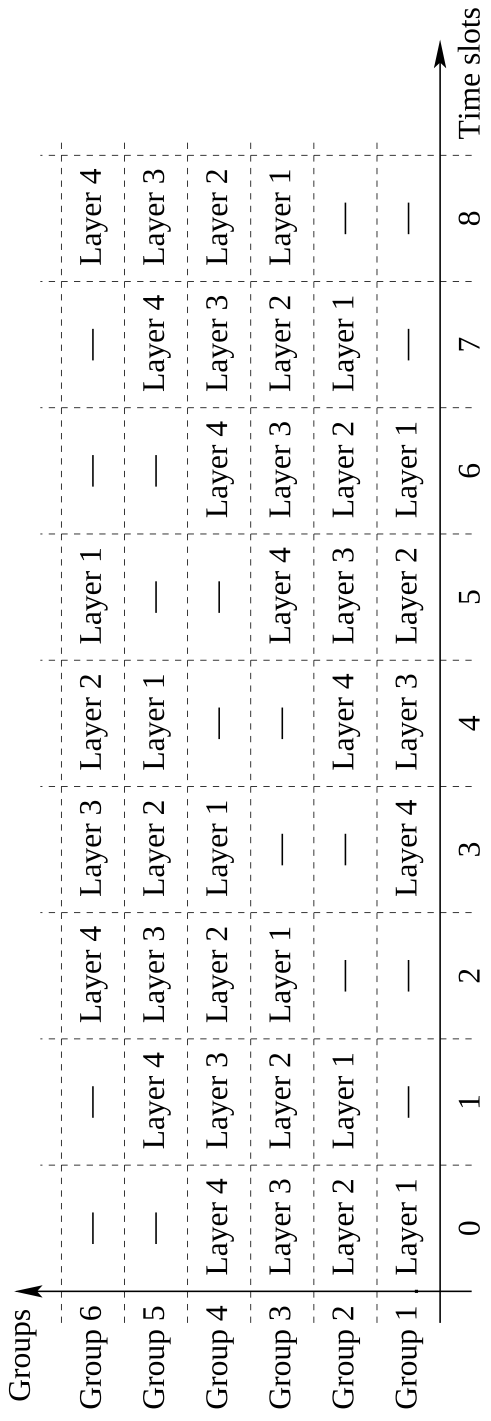
The main novelty in the design of FLID-DL is a clever mechanism for overcoming the large latency associated with leaving a multicast group via IGMP. After a receiver sends an IGMP message asking for unsubscription from a group, up to 9 seconds can elapse before the local router stops forwarding the group packets to the receiver [7]. The large leave latency hampers the responsiveness of congestion control in RLM and RLC where a congested receiver relies on IGMP to abandon the top subscribed layer. FLID-DL solves the problem of the large leave latency without changing IGMP (and thus without modifying the network infrastructure). The solution exploits an insight that the assignment of data layers to multicast groups does not have to be static. In RLM, RLC, and other schemes with the static assignment, leaving a layer is synonymous to leaving the group assigned to this layer.

On the other hand, FLID-DL assigns layers to groups dynamically so that multiple groups take turns delivering the layers. Each of the groups carries a layer for a *time slot* of duration T where $L > T \geq J$, L is the maximum leave latency, and J is the maximum join latency. As soon as the delivery of a layer switches from group A to group B, a receiver of group A leaves the layer even though the receiver cannot – due to the IGMP leave latency – leave group A promptly. Consequently, this technique of *dynamic layering* enables FLID-DL to reduce the temporal granularity of congestion control from L to T.

Overcoming the large leave latency requires additional groups. Let s be the smallest integer such that $s \cdot T \geq L$. A naive implementation of dynamic layering would allocate a separate block of $s + 1$ groups to every layer: each of these groups would operate with a period of $s + 1$ time slots; the group would carry the layer for 1 time slot followed by s slots of idleness (which would give receivers enough time to leave the group before its next turn to deliver the layer). Then, delivery of N layers would require $(s + 1)N$ multicast groups.

To reduce the overhead of dynamic layering from $s \cdot N$ groups to s groups, FLID-DL exploits an observation that a receiver subscribed to layer l during a time slot is supposed to receive layer $l - 1$ during the subsequent time slot, i.e., the group that delivers layer l during time slot t can be reused to deliver layer $l - j$ during time slot $t + j$ where $1 \leq j \leq l - 1$. Hence, FLID-DL implements dynamic layering as follows:

- during time slot t, layer l is carried by group $1 + ((l + t - 1) \bmod (N + s))$ where $1 \leq l \leq N$, and
- the remaining s groups idle during the time slot.



Groups	0	1	2	3	4	5	6	7	8
Group 6	—	—	Layer 4	Layer 3	Layer 2	Layer 1	—	—	Layer 4
Group 5	—	Layer 4	Layer 3	Layer 2	Layer 1	—	—	Layer 4	Layer 3
Group 4	Layer 4	Layer 3	Layer 2	Layer 1	—	—	Layer 4	Layer 3	Layer 2
Group 3	Layer 3	Layer 2	Layer 1	—	—	Layer 4	Layer 3	Layer 2	Layer 1
Group 2	Layer 2	Layer 1	—	—	Layer 4	Layer 3	Layer 2	Layer 1	—
Group 1	Layer 1	—	—	Layer 4	Layer 3	Layer 2	Layer 1	—	—

Figure 4.1: Example of dynamic layering in FLID-DL

In this implementation, each of the $N+s$ groups operates with a period of $N+s$ time slots; an s-slot interval of idleness follows an N-slot interval of transmission; during the transmission interval, the group carries one layer per slot in the decreasing order from layer N to layer 1. Figure 4.1 presents dynamic layering for a FLID-DL session with four data layers (i.e., $N = 4$), time slot duration T of 500 milliseconds, maximum join latency J of 10 milliseconds, and maximum leave latency L of 1 second (i.e., $s = 2$).

With dynamic layering, the group subscription rules for receivers in FLID-DL become as follows:

1. Each receiver must unsubscribe from the group that carries the base layer.

2. An uncongested receiver can keep its current layers by subscribing to group $1 + (g \bmod (N + s))$ where g is the group that carries the current top layer of the receiver.

3. A congested receiver of l layers must drop layer l by not subscribing to any new group.

4. When authorized, an uncongested receiver of l layers can add layer $l + 1$ by subscribing to group $1+((g+1) \bmod (N+s))$ where g is the group that carries its current top layer.

4.1.2 Adaptation

As we showed in Chapter 2, a misbehaving FLID-DL receiver can ignore the above rules and inflate its subscription to acquire data at an unfairly high rate. To protect against the misbehavior with DELTA and SIGMA, we derive a robust protocol

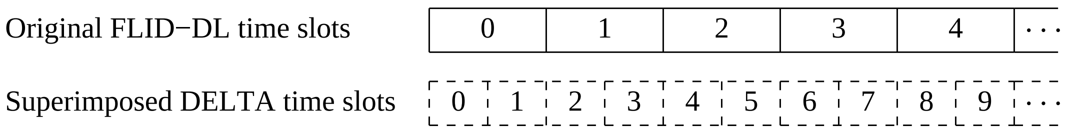


Figure 4.2: Integration of DELTA with FLID-DL

FLID-DS (Fair Layered Increase/Decrease with DELTA and SIGMA) by integrating FLID-DL with a DELTA instantiation for layered multicast. Preserving the responsiveness of FLID-DL congestion control and handling the dynamic assignment of data layers to multicast groups are two main challenges in adapting FLID-DL to FLID-DS.

First of all, both FLID-DL and DELTA use time slots, although for different purposes. In FLID-DL, a time slot determines the temporal granularity of congestion control. In DELTA, a time slot specifies for how long a group key remains valid. Because SIGMA enforces group access with responsiveness of two DELTA time slots, we set the time slot duration for DELTA in FLID-DS to a half of the time slot duration for FLID-DL and superimpose two DELTA time slots on each FLID-DL time slot (see Figure 4.2). This enables FLID-DS to inherit the temporal granularity of FLID-DL congestion control.

To address dynamic layering, FLID-DS defines and communicates group keys as follows. During each DELTA time slot t, the sender distributes component fields and decrease fields among the layers of multicast packets. Receivers use these fields to reconstruct group keys for the DELTA time slot $t + 2$. For each layer l, top key τ_g for group g that will carry layer l is defined as:

$$\tau_g = \bigoplus_{j=1}^{l} \bigoplus_{p \in S_j} c_{j,p} \tag{4.1}$$

where $\oplus$ is an XOR operation, S_j is a set of packets sent to layer j, and $c_{j,p}$ is a nonce placed by the sender into the component field of packet p from layer j. For each layer j such that $1 \leq j \leq N-1$, decrease key δ_g for group g that will carry layer j is defined as:

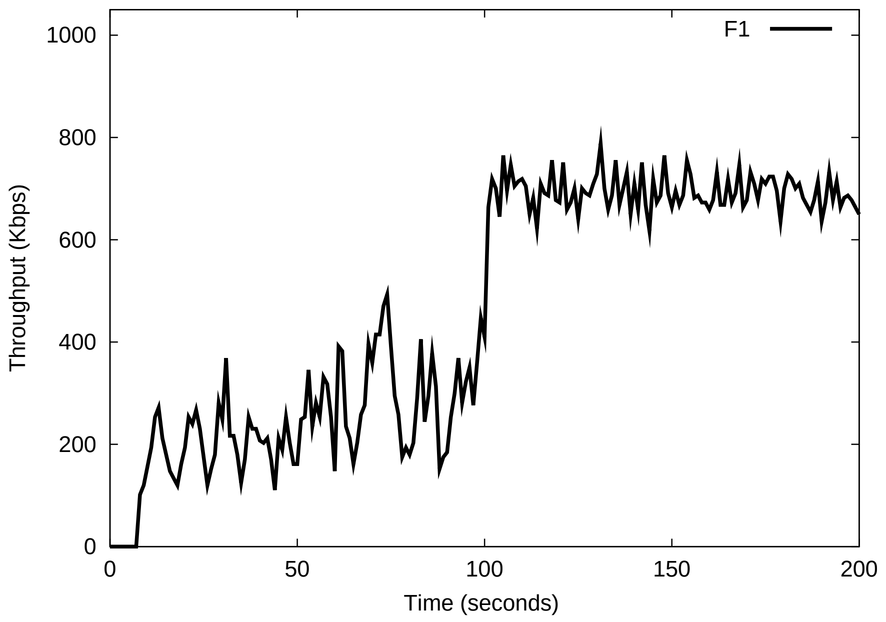
$$\delta_g = d_{j+1} \tag{4.2}$$

where d_{j+1} is a nonce put into decrease field of each packet transmitted to layer $j+1$. When the protocol authorizes an upgrade to layer m where $2 \leq m \leq N$, increase key σ_g for group g that will carry layer m is defined as:

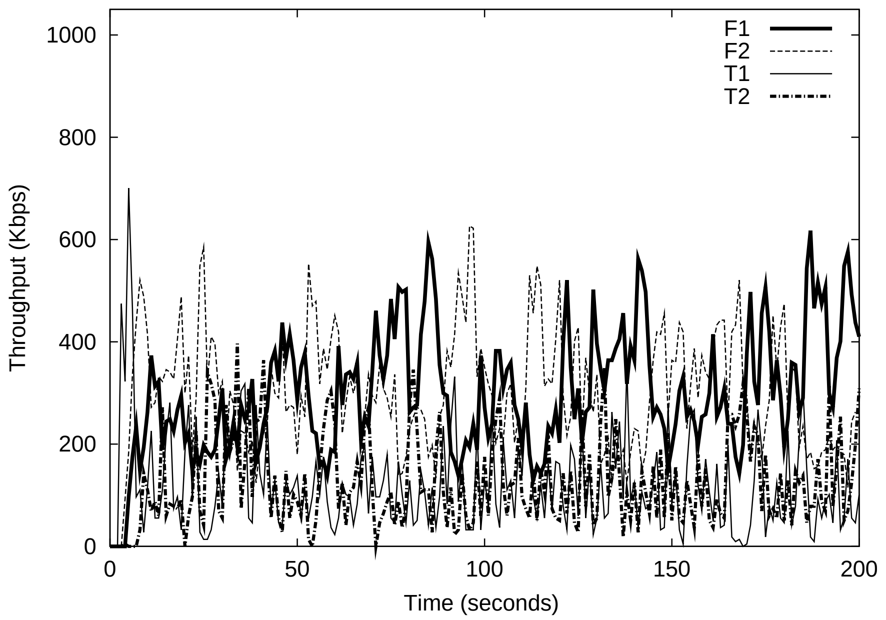
$$\sigma_g = \bigoplus_{j=1}^{m-1} \bigoplus_{p \in S_j} c_{j,p}. \tag{4.3}$$

4.1.3 Evaluation

Now, after having derived FLID-DS, let us evaluate its properties. We conduct experiments in a single-bottleneck topology in NS-2 [36]. Unless stated otherwise, the experimental settings are as follows. Multicast (FLID-DL, FLID-DS) and unicast (TCP Reno, on-off CBR) sessions compete for the bandwidth of the only bottleneck link which is the middle link on a three-link path of each session. The fair bandwidth share for each session is 250 Kbps. The bottleneck link has a propagation delay of 20 ms. Each of the other links has a propagation delay of 10 ms and a capacity of 10 Mbps. The buffer space for each link is equal to two bandwidth-delay products. Every multicast session consists of 10 groups. The minimal group transmits at a rate of 100 Kbps. The cumulative transmission rate of the session grows multiplicatively with a factor of 1.5 per group. The time slot duration for FLID-DL is set to its default value of 500 ms [7]. Hence, the time slot duration for DELTA in FLID-DS is 250 ms. All data traffic uses 576-byte packets.



(a) Vulnerability of FLID-DL to inflated subscription



(b) Immunity of FLID-DS to inflated subscription

Figure 4.3: Protection of FLID-DL with DELTA and SIGMA

Preventing Inflated Subscription

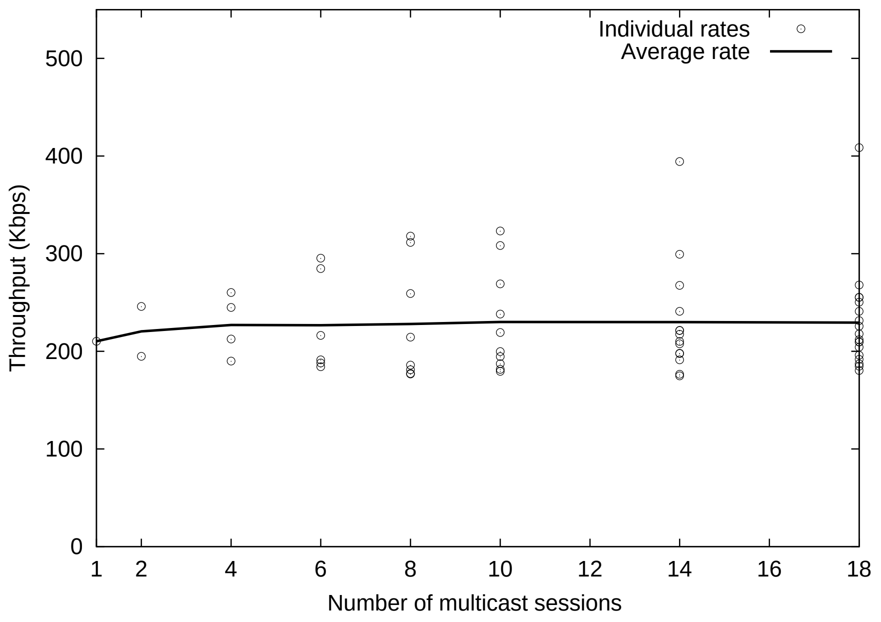
First, consider a setting where receivers F1 and F2 from different FLID-DL sessions share the 1 Mbps bottleneck link with two TCP Reno [2] receivers T1 and T2. After 100 seconds into the simulation, receiver F1 starts to misbehave and inflates its subscription in violation of the protocol. As Figure 4.3(a) illustrates, such a misbehavior boosts the throughput of F1 to 690 Kbps at the expense of well-behaving receivers F2, T1, and T2. We repeat this experiment when the multicast sessions use FLID-DS instead of FLID-DL. Although F1 tries to inflate its subscription after 100 seconds, DELTA and SIGMA preserve – as Figure 4.3(b) shows – the fair bandwidth allocation.

Preserving Congestion Control Properties

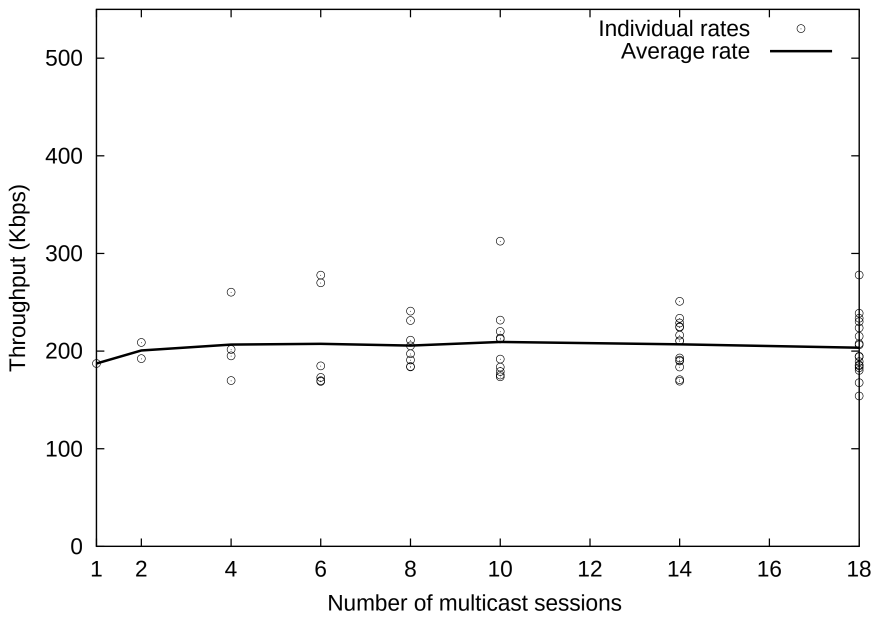
Now, let us investigate whether FLID-DS preserves other congestion control properties of FLID-DL.

Impact on throughput. In this series of experiments, we compare FLID-DL and FLID-DS with respect to the average throughput of a multicast receiver. Each experiment lasts 200 seconds. We vary the number of multicast (FLID-DL or FLID-DS) sessions from 1 to 18. For the only receiver of each multicast session, we measure its throughput over the experiment duration.

First, we examine the multicast sessions in the absence of cross traffic. Figures 4.4(a) and 4.4(b) report individual throughput and average throughput (averaged over the number of sessions) for FLID-DL and FLID-DS receivers respectively. Figure 4.5(a) shows that the receivers achieve similar average throughput in FLID-DL and FLID-DS.

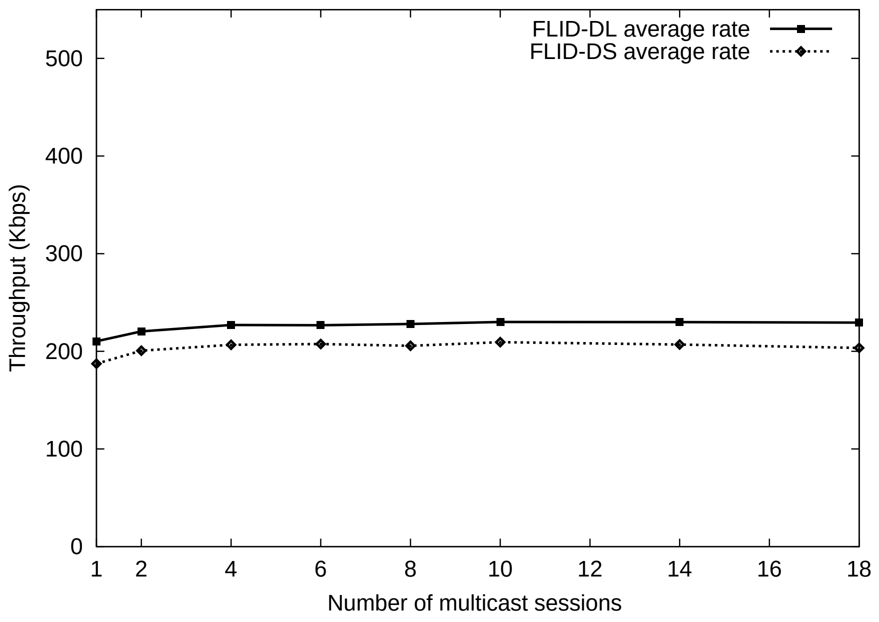


(a) Throughput with FLID-DL

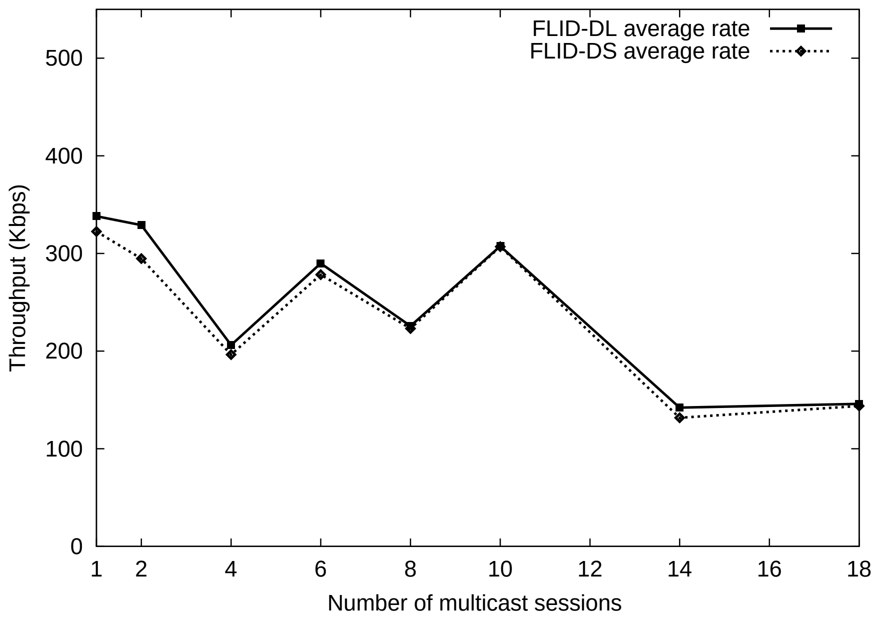


(b) Throughput with FLID-DS

Figure 4.4: Impact of DELTA and SIGMA on FLID-DL throughput



(a) Average throughput without cross traffic



(b) Average throughput with cross traffic

Figure 4.5: Preservation of FLID-DL average throughput

Then, we experiment in a setting where the number of TCP sessions is the same as the number of multicast sessions. In addition, the bottleneck link also serves an on-off CBR session. During an on-period, the CBR session transmits at a rate equal to 10% of the bottleneck link capacity. Each on-period or off-period lasts 5 seconds. Figure 4.5(b) shows that whereas the bandwidth allocation for multicast traffic depends on the number of sessions, receivers of FLID-DL and FLID-DS sessions have similar average throughput.

Responsiveness. To study the impact of DELTA and SIGMA on the responsiveness of multicast congestion control, we consider a setting where the bottleneck link is shared only by a multicast (FLID-DL or FLID-DS) session and an on-off CBR session. The CBR session transmits its data at a rate of 800 Kbps during the time interval between 45 seconds and 75 seconds. Figure 4.6 shows that FLID-DS preserves the responsiveness of the original FLID-DL protocol.

Heterogeneous round-trip times. In FLID-DL, average throughput of a receiver is relatively independent from the round-trip time of the receiver. In this experiment, we verify that DELTA and SIGMA preserve this property. In the considered setting, the only multicast (FLID-DL or FLID-DS) session has 20 receivers. The bottleneck link has a propagation delay of 5 ms. The propagation delays of the other links are chosen so that the round-trip times of the receivers spread uniformly between 30 ms and 220 ms. Figure 4.7 plots average throughput of the receivers as a function of their round-trip times. The average throughput of the FLID-DS receivers is almost constant and remains close to the average throughput of the FLID-DL receivers.

Subscription convergence. When multiple receivers of the same FLID-DL

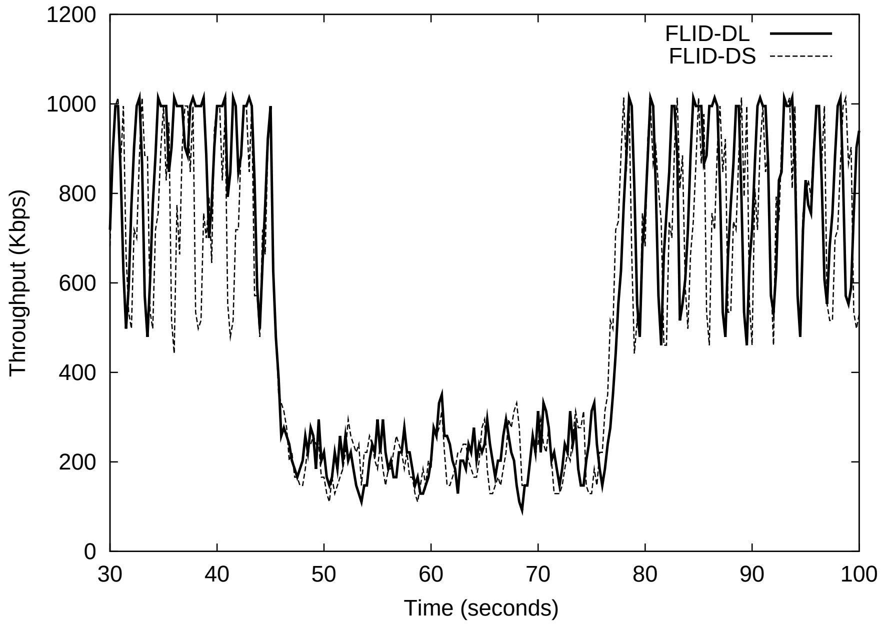


Figure 4.6: Preservation of FLID-DL responsiveness

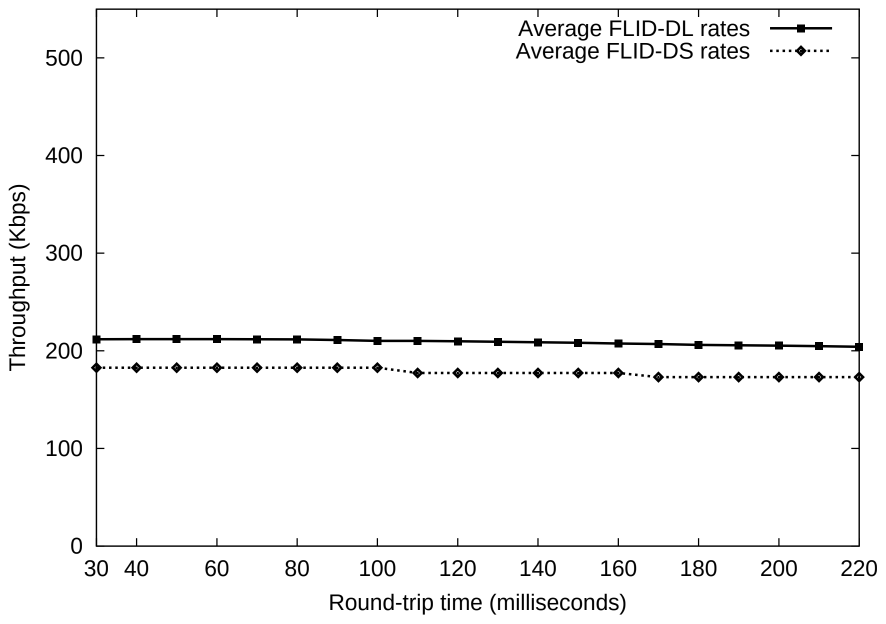


Figure 4.7: Impact of heterogeneous round-trip times

session share a bottleneck link, the receivers converge to the same subscription level even if they join the session at different times. In our experiment, the only multicast (FLID-DL or FLID-DS) session has 4 receivers. The receivers join the session at times 0, 10 seconds, 20 seconds, and 30 seconds respectively. As Figures 4.8(a) and 4.8(b) indicate, the receivers converge to the same fair subscription both in FLID-DL and FLID-DS.

Overhead

In the context of FLID-DS, we now analyze the overhead of communicating the group keys to receivers and edge routers via DELTA – as described in Section 3.2.1 – and SIGMA.

Let the base layer of a FLID-DS session transmit data at rate r. If the cumulative transmission rate grows multiplicatively with the factor of m per layer, the session transmits data at cumulative rate R:

$$R = r \cdot m^{N-1} \tag{4.4}$$

where N is the number of layers in the session.

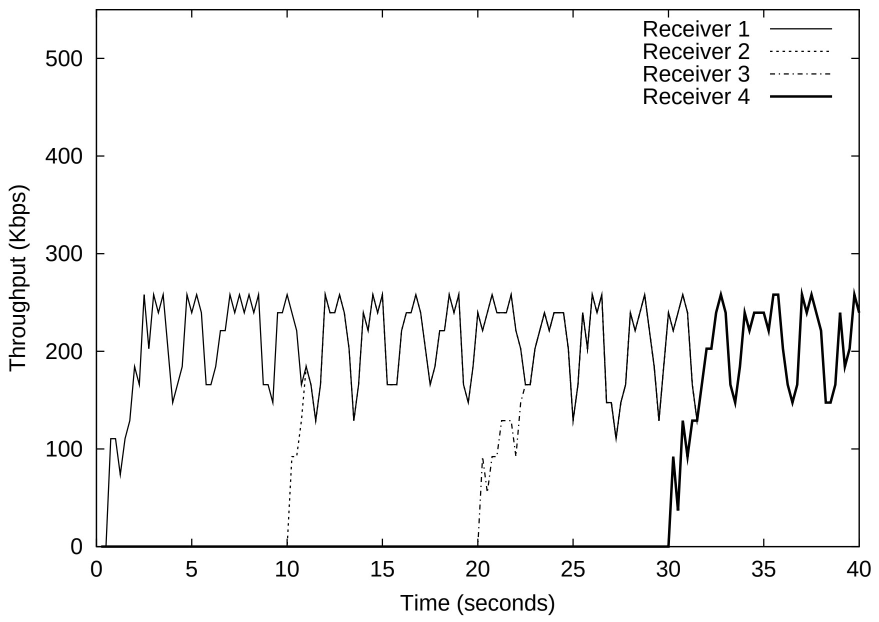
If each packet carries s bits of data, then the session transmits in average

$$P = \frac{R \cdot t}{s} = \frac{r \cdot t \cdot m^{N-1}}{s} \tag{4.5}$$

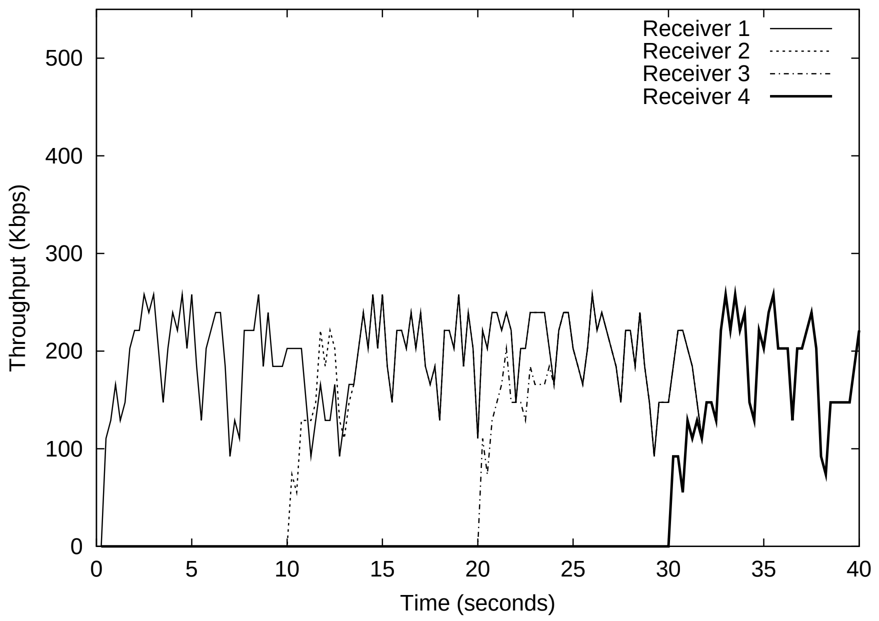
packets during a DELTA time slot of duration t, and

$$p = \frac{r \cdot t}{s} \tag{4.6}$$

of these packets belong to the base layer 1.



(a) Subscription convergence with FLID-DL



(b) Subscription convergence with FLID-DS

Figure 4.8: Preservation of FLID-DL subscription convergence

DELTA communicates the keys to receivers by adding a b-bit component field to each packet and b-bit decrease field to every packet of layer g such that $2 \leq g \leq N$. Therefore, communication overhead O_Δ imposed by DELTA equals:

$$
\begin{aligned}
& O_\Delta \\
= \quad & \{ \text{ ratio of the DELTA bits to the data bits } \} \\
& \frac{(2P-p)b}{R \cdot t} \\
= \quad & \{ \text{ Equations 4.4, 4.5, and 4.6 } \} \\
& \frac{(\frac{2r \cdot t \cdot m^{N-1}}{s} - \frac{r \cdot t}{s})b}{r \cdot t \cdot m^{N-1}} \\
= \quad & \{ \text{ simplification } \} \\
& \Big(2 - \frac{1}{m^{N-1}}\Big)\frac{b}{s}.
\end{aligned}
$$

SIGMA communicates the keys to edge routers via special packets. For each DELTA time slot, these packets deliver an l-bit slot number and one address-key tuple per layer. Every tuple contains a 32-bit multicast group address and b-bit top key. Each of the tuples for layers 1 through $N-1$ also includes a b-bit decrease key. Besides, when the protocol authorizes – with an average frequency of f_g per time slot – an upgrade to the group that will carry layer g, the tuple for this layer carries a b-bit increase key. To ensure reliable delivery of this information, the sender uses forward error correction that increases the bit requirements by the factor of z. The headers of the special packets consume a total of h bits. Hence, communication overhead O_Σ imposed by SIGMA is equal to:

$$
\begin{aligned}
& O_\Sigma \\
= \quad & \{ \text{ ratio of the SIGMA bits to the data bits } \}
\end{aligned}
$$

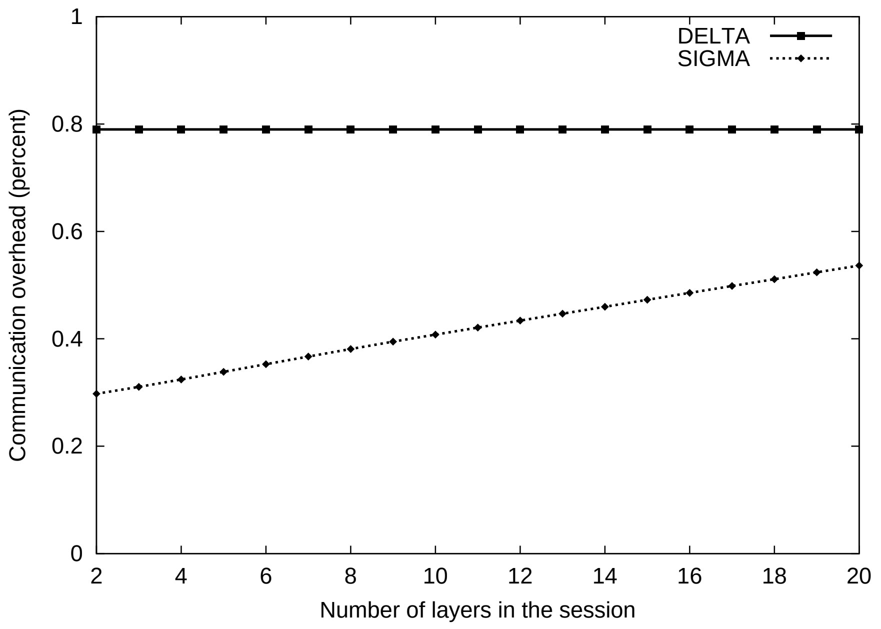
$$\begin{aligned}
&\frac{\Big(l + (32+b)N + b(N-1) + b\sum_{g=2}^{N} f_g\Big)z + h}{R \cdot t} \\
= \quad &\{\text{ Equation 4.4 and simplification }\} \\
&\frac{\Big(l + 32N + b(2N - 1 + \sum_{g=2}^{N} f_g)\Big)z + h}{r \cdot t \cdot m^{N-1}}.
\end{aligned}$$

To quantify the derived expressions, we experiment with a FLID-DS session that transmits its 500-byte data packets (i.e., $s = 4000$) at cumulative rate R of 4 Mbps. Transmission rate r of the base layer is 100 Kbps. Keys and their components are 16 bits long. A slot number consists of 8 bits. Error correction overcomes 50% packet loss. After setting R, r, and N, we determine m from Equation 4.4. During the experiments, we record the observed values of f_g, z, and h.

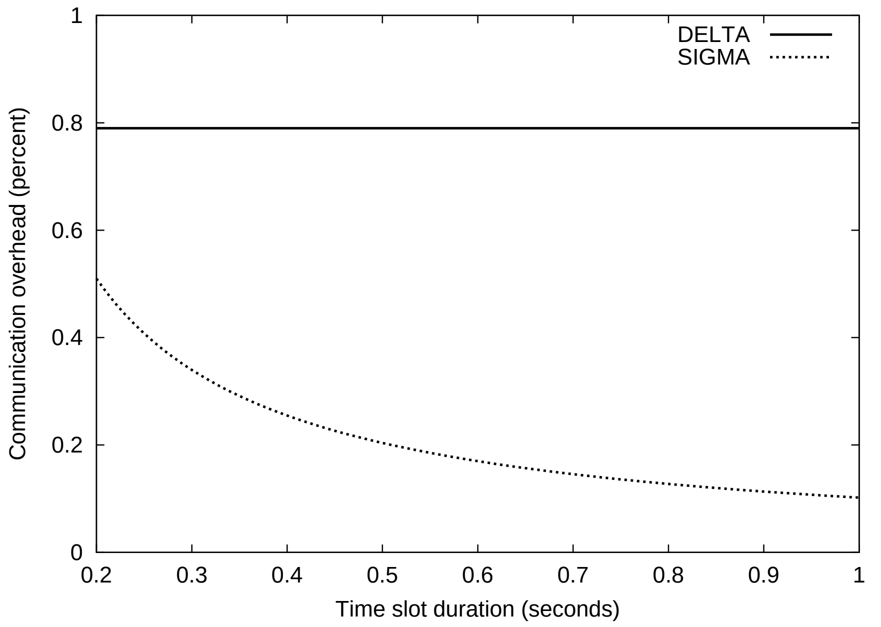
First, we explore the dependence of the overhead on the number of layers. Figure 4.9(a) shows O_Δ and O_Σ for N varying from 2 to 20 when $t = 250$ ms. Then, we examine the impact of the time slot duration. Figure 4.9(b) plots O_Δ and O_Σ for t varying from 200 ms to 1 second when $N = 10$. In both cases, the communication overhead remains about 0.8% for DELTA and stays under 0.6% for SIGMA. Thus, DELTA and SIGMA protect against inflated subscription without imposing a significant overhead.

4.2 Robust Adaption of RLM

In Section 4.1, we designed and evaluated FLID-DS, an adaptation of FLID-DL that is robust to inflated subscription of misbehaving receivers. Whereas FLID-DL treats a single packet loss as congestion, other existing protocols define a congested state differently. Let us now focus on Receiver-driven Layered Multicast (RLM) [30]



(a) Dependence on the number of layers in the session



(b) Dependence on the DELTA time slot duration

Figure 4.9: Communication overhead of DELTA and SIGMA in FLID-DS

as a representative of threshold-based congestion control protocols.

4.2.1 Protocol Description

RLM is an unreliable multicast protocol that started it all for feedback-free congestion control. The sender assigns data layers to multicast groups statically. In the default setting, a receiver is considered congested if it receives less than 75% of packets transmitted to its current subscription level. The design of RLM demonstrated that receivers can control congestion solely via the mechanism of group subscription at local routers, without communicating any information to the sender. On the other hand, RLM suffers from a number of drawbacks eliminated in its successors. In particular, subscription synchronization in RLM relies on the receiver-driven mechanism of shared learning.

4.2.2 Adaptation

As we showed in Section 2.2.6, a misbehaving receiver can abuse the mechanism of shared learning to prevent other receivers from reaching their fair subscription levels. Hence, designing a robust version of RLM involves incorporating techniques that protect not only against inflating own subscription but also against subduing legitimate subscriptions of other receivers.

First, we protect RLM receivers from being manipulated by other receivers in the session. To do so, we substitute shared learning with a sender-driven mechanism for subscription synchronization, as implemented in FLID-DL. We refer to the adjusted design as RLM-F (Receiver-driven Layered Multicast with FLID-like subscription synchronization). Instead of relying on personal join timers, uncongested

RLM-F receivers increase their subscription levels in a layered session in response to explicit increase signals from the sender. Similarly to FLID-DL, RLM-F operates in terms of time slots: each increase signal is associated with a time slot and authorizes uncongested receivers to add a layer at the end of the corresponding time slot. On the other hand, RLM-F inherits from RLM the static assignment to data layers to multicast groups (and thus is affected by large IGMP leave latencies).

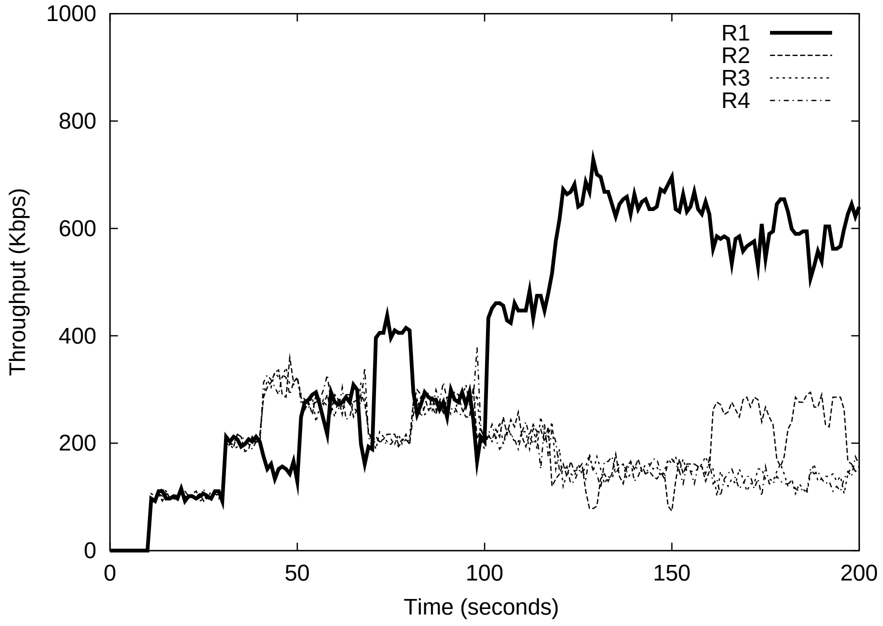
With the sender-driven synchronization of subscriptions, RLM-F is immune to prevention of other receivers from legitimate subscriptions. However, RLM-F remains vulnerable to inflated subscription attacks. To complete our transformation of RLM into a multicast congestion control protocol that is robust to receiver misbehavior, we integrate RLM-F with a DELTA instantiation for threshold-based protocols, as described in Section 3.2.1. We refer to the resulting protocol as RLM-DS (Receiver-driven Layered Multicast with DELTA and SIGMA). To provide RLM-DS with the same temporal granularity of congestion control as in RLM-F, we set the time slot duration for DELTA in RLM-DS to a half of the time slot duration for RLM-F and superimpose two DELTA time slots on each RLM-F time slot. For each DELTA time slot, the sender uses Shamir's scheme to generate and distribute components of group keys among the packets transmitted during the time slot. The components of the `top` key for the top group of a subscription level are distributed among all the packets transmitted to this level. Only if the loss rate does not exceed the threshold prescribed by the original RLM protocol (in the default setting, the threshold equals 25% of the transmitted packets), the receiver collects enough components for reconstructing the key. Keys for different groups do not share components. Consequently, each packet of group j carries $N - j + 1$ components, one component for each group j through N where N is the number of groups in the session.

4.2.3 Evaluation

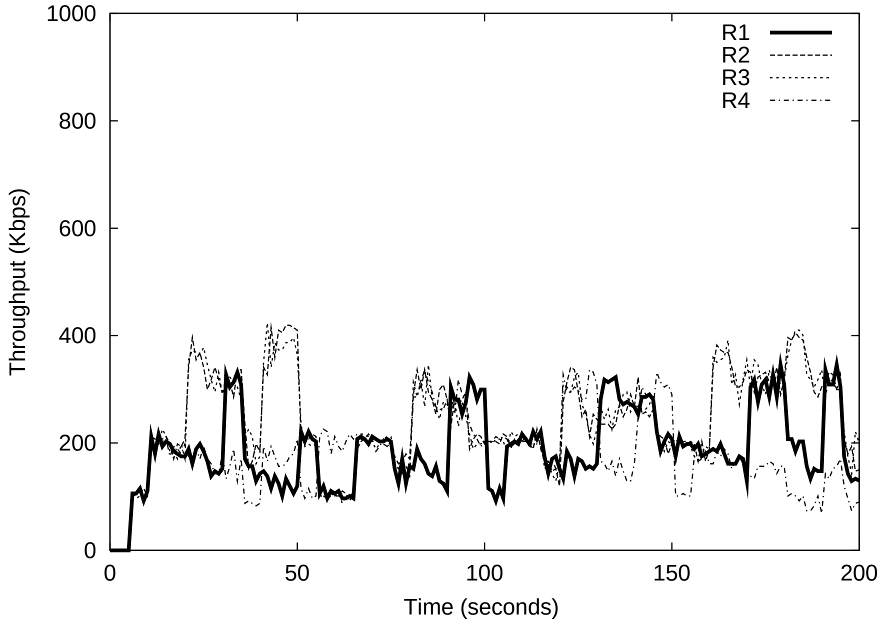
Above, we have enhanced RLM-F with DELTA to derive RLM-DS, a protocol robust to inflated subscription in networks with SIGMA group access control. Below, we verify experimentally that DELTA and SIGMA protect RLM-DS against the receiver misbehavior. Then, we examine the impact of DELTA and SIGMA on throughput and responsiveness of RLM-F. Finally, we explore the communication overhead introduced by DELTA and SIGMA. In our evaluation of RLM-DS, we employ the same network topology as in the above experiments with FLID-DL. Each multicast session (RLM-F or RLM-DS) consists of 6 groups with each group carrying a statically assigned data layer. The sender transmits the base layer at a rate of 100 Kbps. Each enhancement layer doubles the cumulative transmission rate of the session. The time slot duration for RLM-F is set to 10 seconds. Hence, the time slot duration for DELTA in RLM-DS is 5 seconds. The base layer carries an increase signal every time slot. Each enhancement layer halves the frequency of increase signals. The loss threshold for each subscription level is set to 25% of the packets transmitted to this level during the time slot. All data traffic uses 576-byte packets.

Addressing the Threat of Inflated Subscription

We first consider a setting where four receivers R1, R2, R3, and R4 from different RLM-F sessions share the 1.1 Mbps bottleneck link. Up to 100 seconds into the experiment, all the receivers adhere to the protocol and converge towards fair and



(a) Vulnerability of RLM-F to inflated subscription



(b) Immunity of RLM-DS to inflated subscription

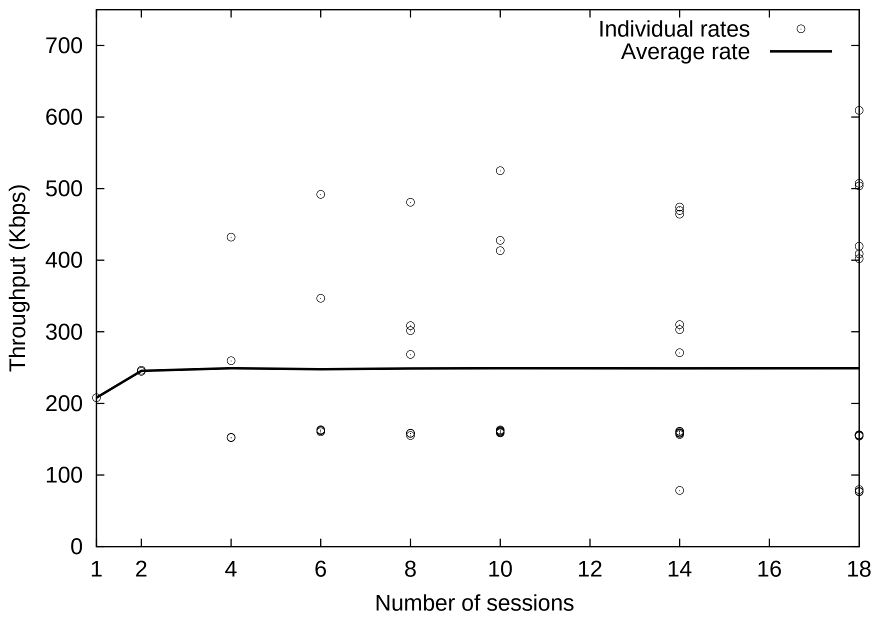
Figure 4.10: Protection of RLM-F with DELTA and SIGMA

efficient sharing of the bottleneck bandwidth. After 100 seconds, receiver R1 misbehaves by inflating its subscription to 4 layers. As Figure 4.10(a) illustrates, such the misbehavior rewards R1 with an unfairly high throughput at the expense of well-behaving receivers R2, R3, and R4. We repeat this experiment when the multicast sessions use RLM-DS instead of RLM-F. Although R1 tries to inflate its subscription after 100 seconds, DELTA and SIGMA protect – as Figure 4.10(b) shows – the fairness of the bandwidth allocation.

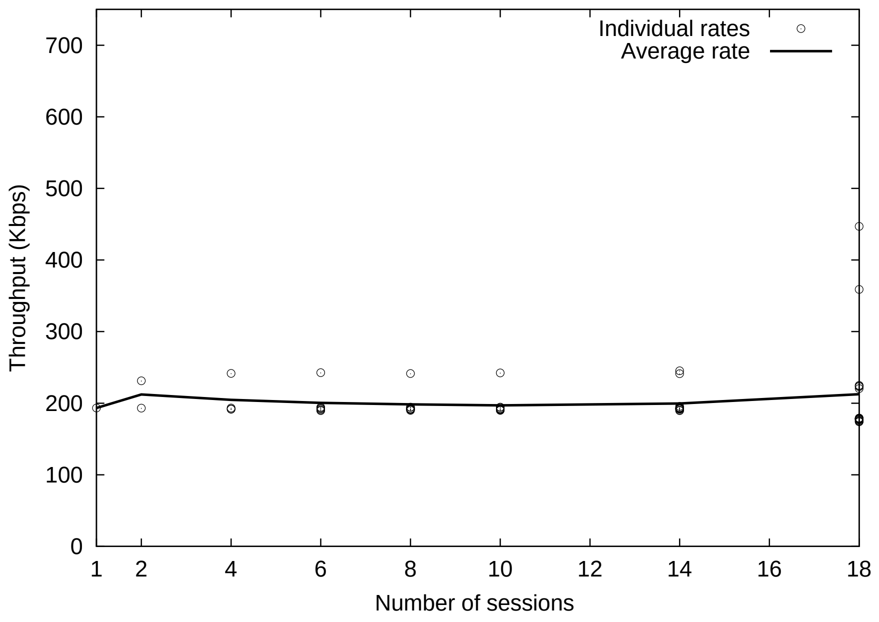
Impact on Other Congestion Control Properties

Throughput. We compare RLM-F and RLM-DS with respect to the average throughput achieved by a multicast receiver. Each experiment lasts 200 seconds. We vary the number of multicast (RLM-F or RLM-DS) sessions from 1 to 18. There is no cross traffic in these experiments. For the only receiver in each multicast session, we measure its throughput over the experiment duration. The fair bandwidth share is equal to 250 Kbps per receiver. Figures 4.11(a) and 4.11(b) report individual throughput and average throughput (averaged over the number of sessions) for RLM-F and RLM-DS receivers respectively. Figure 4.12 compares the average throughput in RLM-F and RLM-DS. The graphs show that while supporting a somewhat smaller average throughput, DELTA and SIGMA reduce the deviation for individual throughputs in RLM-F and thereby improve the intra-protocol fairness.

Responsiveness. To assess the impact of DELTA and SIGMA on the responsiveness of RLM-F congestion control, we consider a setting where the bottleneck link with a capacity of 500 Kbps is shared only by a multicast (RLM-F or RLM-DS) session and an on-off CBR session. The CBR session transmits its data



(a) Throughput with RLM-F



(b) Throughput with RLM-DS

Figure 4.11: Impact of DELTA and SIGMA on RLM-F throughput

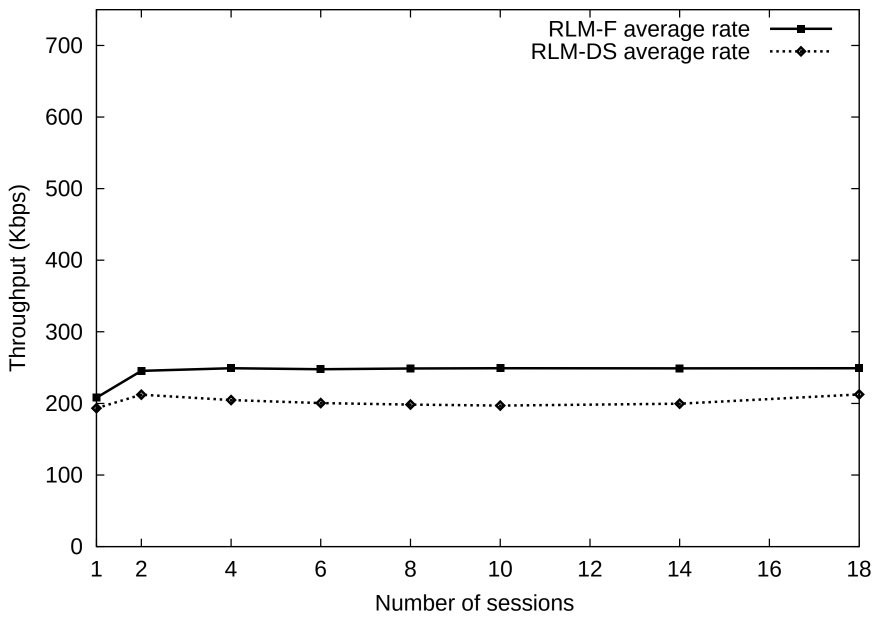


Figure 4.12: Average throughput in RLM-F and RLM-DS

at a rate of 460 Kbps during the time interval between 60 seconds and 110 seconds. Figure 4.13 shows that RLM-DS preserves the responsiveness of RLM-F congestion control.

Overhead

In Section 4.1.3, we derived an analytical expression for the communication overhead imposed by SIGMA in the context of FLID-DS. This expression holds valid for RLM-DS because both protocols follow the same pattern to distribute group keys from the sender to edge routers: for each time slot, the sender communicates a top key for each group, a decrease key for each lower group of the session, and an increase key for each upper group authorized to be added. Consequently, since SIGMA imposes no significant communication overhead in FLID-DS, the same is

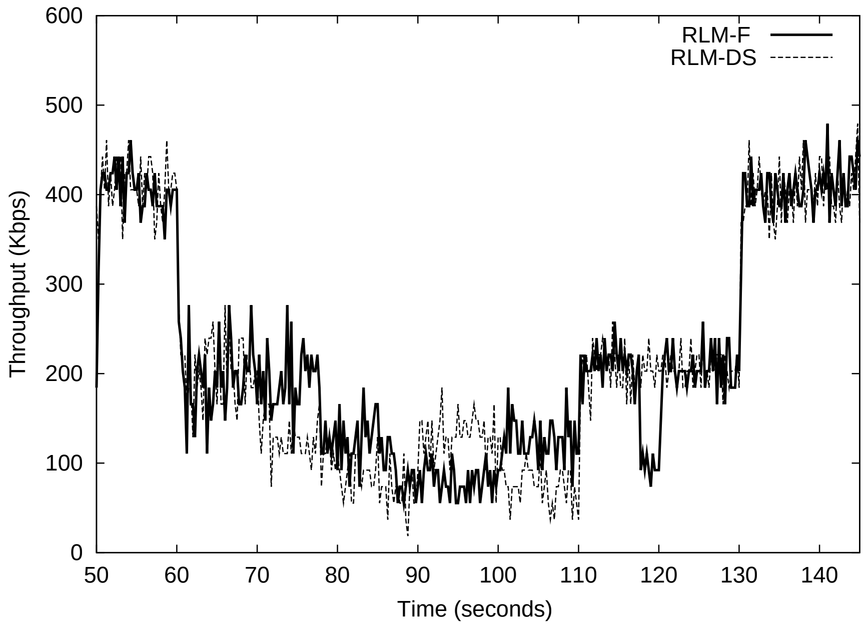


Figure 4.13: Preservation of RLM-F responsiveness

true in RLM-DS.

The instantiation of DELTA for RLM-DS, however, is different in two respects. First of all, a component of a top key in Shamir's scheme is not a single value but a tuple $(z, q(z))$ where z is an integer, and q is a polynomial. Second, since different keys require different polynomials, Shamir's scheme does not enable the keys for upper groups to reuse $q_j(z)$ from the component of the key for a lower group j. Thus, each packet of group j carries one b-bit value z and $N - j + 1$ b-bit values $q_g(z)$ where N is the number of groups in the session, and $j \leq g \leq N$. In addition to communicating the components of top keys, DELTA inserts a b-bit decrease key into every packet of group j such that $2 \leq j \leq N$.

Let us now derive communication overhead O_R introduced by DELTA for transforming RLM-F to RLM-DS. If each packet of a RLM-DS session carries s bits of data, and group 1 transmits at rate r, then group 1 transmits in average

$$p_1 = \frac{r \cdot t}{s} \tag{4.7}$$

packets during a DELTA time slot of duration t. If the cumulative transmission rate grows multiplicatively with the factor of m per group, group j (where $2 \leq j \leq N$) transmits in average

$$p_j = (m^{j-1} - m^{j-2})\frac{r \cdot t}{s} = m^{j-2}(m-1)\frac{r \cdot t}{s} \tag{4.8}$$

packets during the time slot. Then, communication overhead for DELTA equals:

$$\begin{aligned}
& O_R \\
= \quad & \{ \text{ ratio of the DELTA bits to the data bits } \} \\
& \frac{\Big(p_1(N+1) + \sum_{j=2}^{N} (p_j(N-j+3))\Big)b}{\Big(\sum_{j=1}^{N} p_j\Big)s} \\
= \quad & \{ \text{ Equations 4.7 and 4.8 } \} \\
& \frac{\Big(\frac{r \cdot t}{s}(N+1) + \sum_{j=2}^{N} (m^{j-2}(m-1)\frac{r \cdot t}{s}(N-j+3))\Big)b}{\Big(\frac{r \cdot t}{s} + \sum_{j=2}^{N} (m^{j-2}(m-1)\frac{r \cdot t}{s})\Big)s} \\
= \quad & \{ \text{ simplification } \} \\
& \frac{\Big(N+1+\sum_{j=2}^{N} (m^{j-2}(m-1)(N-j+3))\Big)}{m^{N-1}} \cdot \frac{b}{s} \\
= \quad & \{ \text{ simplification } \} \\
& \frac{2m^{N-1} + \sum_{j=2}^{N} m^{j-1}}{m^{N-1}} \cdot \frac{b}{s} \\
= \quad & \{ \text{ simplification } \} \\
& \frac{3m^{N-1} - 2m^{N-2} - 1}{m^{N-2}(m-1)} \cdot \frac{b}{s}.
\end{aligned}$$

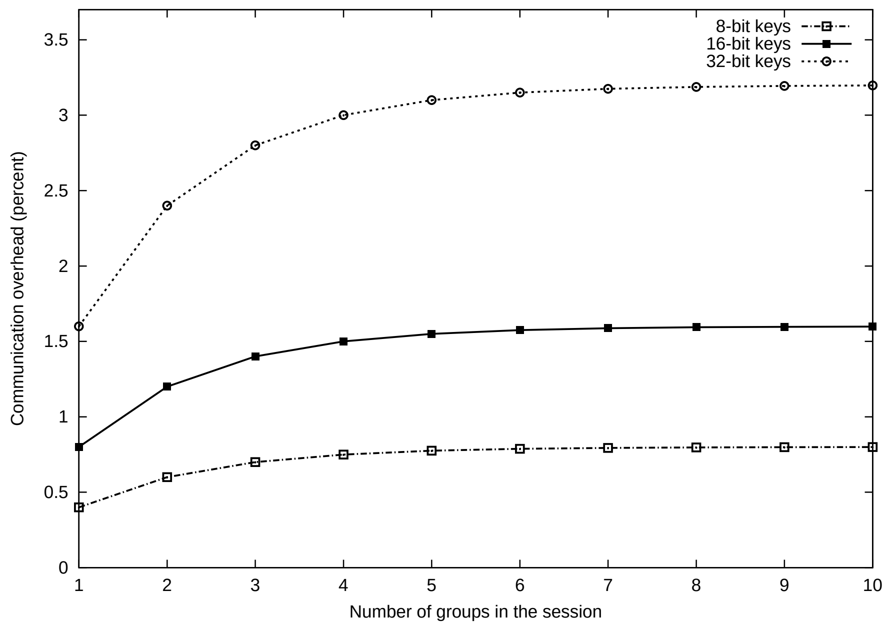


Figure 4.14: Communication overhead of DELTA in RLM-DS

Note that the derived expression is bounded from above by a value that is independent from the number of groups in the session:

$$O_R < \Big(3 + \frac{1}{m-1}\Big)\frac{b}{s}. \tag{4.9}$$

For example, if $m = 2$ (i.e., each enhancement layer doubles the cumulative transmission rate), then DELTA inserts in average less than 4 b-bit fields per multicast packet.

To quantify the communication overhead even further, let us consider a RLM-DS session that consists of 6 groups (i.e., $N = 6$), doubles its cumulative transmission rate with each enhancement layer (i.e., $m = 2$), transmits 500-byte data packets (i.e., $s = 4000$), and uses 16-bit keys (i.e., $b = 16$). In this setting, DELTA adds in average 3.94 fields (16 bits long each) per multicast packet and thus introduces 1.6% communication overhead. Figure 4.14 plots the overhead for keys of size 8 bits, 16 bits, and 32 bits. Even though Shamir's scheme does not enable reuse of components, the communication overhead grows very slowly after the number of groups in the session exceeds 5. This result (as well as the upper bound from Inequality 4.9) is due to the multiplicative growth of the group transmission rates: most of the packets belong to upper groups where each packet carries only few components. Thus, DELTA and SIGMA protect RLM-DS against inflated subscription without imposing a significant overhead.

Chapter 5

Conclusion

5.1 Dissertation Summary

In this dissertation, we investigated the impact of trust assumptions on fairness of congestion control protocols for IP multicast. Traditional multicast congestion control relies on cooperation: each party is assumed to follow guidelines for fair sharing of the network bandwidth. We argued that with the growth and commercialization of the Internet, the assumption of universal trust is no longer tenable. The dissertation studied a relaxed model where receivers are distrusted and can misbehave to acquire an unfairly high bandwidth at the expense of competing traffic. We identified potential vulnerabilities of multicast congestion control mechanisms to receiver misbehavior. Our experiments with existing multicast protocols showed that each of the evaluated protocols is susceptible to at least one of the identified threats.

To take the first step towards robust multicast designs for distrusted environments, we then focused on the class of feedback-free protocols where receivers provide no feedback to the sender and control congestion by regulating their subscription levels in the multi-group session. Unfortunately, the mechanism of group subscription offers a misbehaving receiver an opportunity to inflate its subscription level. We demonstrated that inflated subscription attacks pose a major threat to fairness of bandwidth allocation.

This dissertation is the first to solve the problem of inflated subscription. The presented designs exploited an insight that the ability of a receiver to access a multicast group should be tied with the congestion status of the receiver. First, we addressed individual attacks where a receiver inflates its subscription with no assistance from other receivers. Our solution guards access to multicast groups with dynamic keys and consists of two independent components: DELTA (Distribution of ELigibility To Access) – a novel method for in-band distribution of group keys to receivers that are eligible to access the groups according to the congestion control protocol, and SIGMA (Secure Internet Group Management Architecture) – a generic architecture for key-based group access at edge routers. DELTA and SIGMA require only minimal generic changes in the edge routers, do not alter the core of the network, and introduce no auxiliary servers. Then, we extended the design to protect multicast congestion control against inflated subscription of colluding receivers. To illustrate that integration with DELTA and SIGMA makes multicast protocols robust to inflated subscription and preserves other congestion control properties, we derived and evaluated robust adaptations of RLM and FLID-DL protocols.

5.2 Future Work

Whereas this dissertation proposed robust mechanisms for IP multicast congestion control, slow deployment of IP multicast has stirred an interest in *end-system mul-*

ticast. Emerged designs for end-system multicast can be classified into *server-based* and *peer-to-peer* architectures.

In server-based multicast, trusted managed servers – e.g., edge servers from Akamai [1] – form multicast routing hierarchies. Server-based architectures can adopt DELTA and SIGMA straightforwardly to acquire robustness against inflated subscription: edge servers can enforce (in the same fashion as edge routers do it for IP multicast) appropriate congestion-dependent access of local receivers to multicast groups.

In peer-to-peer architectures, receivers themselves form multicast routing hierarchies [5]. Thus, receivers participate not only in congestion control but also in routing. The greater degree of involvement creates new opportunities for receiver misbehavior:

1. A misbehaving receiver can seek a self-beneficial routing hierarchy at the expense of others. For example, the slowest receiver can attempt to obtain a direct connection to the sender by displacing faster receivers to lower levels of the multicast routing hierarchy.

2. If the path from the sender to a receiver passes through another receiver, the intermediary receiver can misbehave by forwarding data at a lower than fair rate. Denial-of-service is not the only rationale for such an attack. The slow forwarding can enable the misbehaving intermediary to improve its own reception by acquiring the released bandwidth (e.g., when the bandwidth bottleneck is the wireless connection of the intermediary to the network).

Since DELTA and SIGMA assume that routing is performed by trusted parties, our design is insufficient for ensuring fairness of bandwidth allocation in peer-to-peer architectures. Protection against the new threats of routing misbehavior might require principally different solutions. In particular, it seems difficult to address the attack of slow forwarding with traditional methods based on verification: due to heterogeneous network conditions, an abused receiver has no easy ways to verify the fairness of the allocated rate. For instance, the receiver cannot rely on comparing the rates from disjoint paths because the fair bandwidth shares for such paths can be different. Furthermore, we are not aware of any prior studies that define a fair bandwidth allocation for networks where receivers participate in routing. We plan to address the above issues in our future work on robust peer-to-peer multicast.

Bibliography

[1] Akamai. http://www.akamai.com, April 2003.

[2] M. Allman, V. Paxson, and W. Stevens. TCP Congestion Control. RFC 2581, April 1999.

[3] A. Ballardie and J. Crowcroft. Multicast-Specific Security Threats and Counter-Measures. In *Proceedings Symposium on Network and Distributed System Security*, February 1995.

[4] A. Ballardie, P. Francis, and J. Crowcroft. Core Based Trees (CBT): An Architecture for Scalable Inter-Domain Multicast Routing. In *Proceedings of ACM SIGCOMM '93*, August 1993.

[5] S. Banerjee, B. Bhattacharjee, and C. Kommareddy. Scalable Application Layer Multicast. In *Proceedings of ACM SIGCOMM 2002*, August 2002.

[6] S. Bhattacharyya, D. Towsley, and J. Kurose. The Loss Path Multiplicity Problem for Multicast Congestion Control. In *Proceedings IEEE INFOCOM'99*, March 1999.

[7] J. Byers, M. Frumin, G. Horn, M. Luby, M. Mitzenmacher, A. Roetter, and

W. Shaver. FLID-DL: Congestion Control for Layered Multicast. In *Proceedings NGC 2000*, November 2000.

[8] J. Byers, M. Luby, and M. Mitzenmacher. Fine-Grained Layered Multicast. In *Proceedings IEEE INFOCOM 2001*, April 2001.

[9] J.W. Byers, M. Luby, M. Mitzenmacher, and A. Rege. A Digital Fountain Approach to Reliable Distribution of Bulk Data. In *Proceedings ACM SIGCOMM'98*, September 1998.

[10] S. Y. Cheung and M. H. Ammar. Using Destination Set Grouping to Improve the Performance of Window-controlled Multipoint Connections. *Computer Communications Journal*, 19:723–736, 1996.

[11] S. Y. Cheung, M. H. Ammar, and X. Li. On the Use of Destination Set Grouping to Improve Fairness in Multicast Video Distribution. In *Proceedings IEEE INFOCOM'96*, March 1996.

[12] S. Deering, D. Estrin, D. Farinacci, V. Jacobson, C.-G. Lue, and L. Wei. An Architecture for Wide-Area Multicast Routing. In *Proceedings ACM SIGCOMM '94*, September 1994.

[13] S.E. Deering. *Multicast Routing in a Datagram Internetwork.* PhD thesis, Stanford University, December 1991.

[14] N.G. Duffield, M. Grossglauser, and K.K. Ramakrishnan. Distrust and Privacy: Axioms for Multicast Congestion Control. In *Proceedings NOSSDAV'99*, June 1999.

[15] D. Ely, N. Spring, D. Wetherall, S. Savage, and T. Anderson. Robust Congestion Signaling. In *Proceedings IEEE ICNP 2001*, November 2001.

[16] W. Fenner. Internet Group Management Protocol, Version 2. RFC 2236, November 1997.

[17] S. Floyd, V. Jacobson, C. Liu, S. McCanne, and L. Zhang. A Reliable Multicast Framework for Light-weight Sessions and Application Level Framing. *IEEE/ACM Transactions on Networking*, 5(6):784–803, December 1997.

[18] S. Gorinsky, S. Jain, and H. Vin. Multicast Congestion Control with Distrusted Receivers. In *Proceedings NGC 2002*, October 2002.

[19] S. Gorinsky, S. Jain, H. Vin, and Y. Zhang. Robustness to Inflated Subscription in Multicast Congestion Control. In *Proceedings ACM SIGMETRICS 2003*, June 2003.

[20] S. Gorinsky, S. Jain, H. Vin, and Y. Zhang. Robustness to Inflated Subscription in Multicast Congestion Control. In *Proceedings ACM SIGCOMM 2003*, August 2003.

[21] S. Gorinsky, K.K. Ramakrishnan, and H. Vin. Addressing Heterogeneity and Scalability in Layered Multicast Congestion Control. Technical Report TR2000-31, Department of Computer Sciences, The University of Texas at Austin, November 2000.

[22] S. Gorinsky and H. Vin. The Utility of Feedback in Layered Multicast Congestion Control. In *Proceedings NOSSDAV 2001*, June 2001.

[23] H.W. Holbrook and D.R. Cheriton. IP Multicast Channels: EXPRESS Support for Large-Scale Single-Source Applications. In *Proceedings ACM SIGCOMM'99*, September 1999.

[24] V. Jacobson. Congestion Avoidance and Control. In *Proceedings ACM SIGCOMM'88*, August 1988.

[25] P. Judge and M. Ammar. GOTHIC: A Group Access Control Architecture for Secure Multicast and Anycast. In *Proceedings IEEE INFOCOM 2002*, June 2002.

[26] A. Legout and E. W. Biersack. PLM: Fast Convergence for Cumulative Layered Multicast Transmission Schemes. In *Proceedings ACM SIGMETRICS 2000*, June 2000.

[27] L.H. Lehman, S.J. Garland, and D.L. Tennenhouse. Active Reliable Multicast. In *Proceedings IEEE INFOCOM'98*, March 1998.

[28] M. Luby, V.K. Goyal, S. Skaria, and G.B. Horn. Wave and Equation Based Rate Control Using Multicast Round Trip Time. In *Proceedings ACM SIGCOMM 2002*, August 2002.

[29] R. Mahajan, S. Floyd, and D. Wetherall. Controlling High-Bandwidth Flows at the Congested Router. In *Proceedings IEEE ICNP 2001*, November 2001.

[30] S. McCanne, V. Jacobson, and M. Vetterli. Receiver-driven Layered Multicast. In *Proceedings ACM SIGCOMM'96*, August 1996.

[31] S. Mittra. Iolus: A Framework for Scalable Secure Multicasting. In *Proceedings ACM SIGCOMM'97*, September 1997.

[32] J. Moy. Multicast Extensions to OSPF. *RFC 1584*, March 1994.

[33] MSTAT Manual Page. http://man-pages.net/linux/man8/mstat.8.html, April 2002.

[34] J. Nonnenmacher, E.W. Biersack, and D. Towsley. Parity-Based Loss Recovery for Reliable Multicast Transmission. *IEEE/ACM Transactions on Networking*, 6(4):349–361, August 1998.

[35] J. Nonnenmacher, M. Lacher, M. Jung, E.W. Biersack, and G. Carle. How Bad is Reliable Multicast without Local Recovery? In *Proceedings IEEE INFOCOM'98*, March 1998.

[36] UCB/LBNL/VINT Network Simulator NS-2. http://www-mash.cs.berkeley.edu/ns, May 2002.

[37] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose. Modeling TCP Throughput: A Simple Model and its Empirical Validation. In *Proceedings ACM SIGCOMM'98*, September 1998.

[38] C. Papadopoulos, G. Parulkar, and G. Varghese. An Error Control Scheme for Large-Scale Multicast Applications. In *Proceedings IEEE INFOCOM'98*, March 1998.

[39] S. Paul, K. Sabnani, J.C. Lin, and S. Bhattacharyya. Reliable Multicast Transport Protocol (RMTP). *IEEE Journal on Selected Areas in Communications*, 15(3), April 1997.

[40] R. Perlman, C.-Y. Lee, J. Crowcroft, Z. Wang, C. Diot, J. Thoo, and M. Green.

Simple Multicast: A Design for Simple, Low-Overhead Multicast. Internet Draft, February 1999.

[41] A. Perrig, R. Canetti, D. Song, and D. Tygar. Efficient and Secure Source Authentication for Multicast. In *Proceedings NDSS 2001*, February 2001.

[42] K.K. Ramakrishnan and S. Floyd. A Proposal to Add Explicit Congestion Notification (ECN) to IP. RFC 2481, January 1999.

[43] L. Rizzo. pgmcc: A TCP-friendly Single-Rate Multicast Congestion Control Scheme. In *Proceedings ACM SIGCOMM 2000*, August 2000.

[44] S. Savage, N. Cardwell, D. Wetherall, and T. Anderson. TCP Congestion Control with a Misbehaving Receiver. *ACM Computer Communications Review*, 29(5):71–78, October 1999.

[45] N. Shacham. Multipoint Communication by Hierarchically Encoded Data. In *Proceedings IEEE INFOCOM'92*, May 1992.

[46] A. Shamir. How to Share a Secret. *Communications of the ACM*, 22(11):612–613, November 1979.

[47] D. Sisalem and A. Wolisz. MLDA: A TCP-friendly Congestion Control Framework for Heterogenous Multicast Environments. In *Proceedings IWQoS 2000*, June 2000.

[48] T. Speakman, J. Crowcroft, J. Gemmell, D. Farinacci, S. Lin, D. Leshchiner, M. Luby, T. Montgomery, L. Rizzo, A. Tweedly, N. Bhaskar, R. Edmonstone, R. Sumanasekera, and L. Vicisano. PGM Reliable Transport Protocol Specification. RFC 3208, http://www.ietf.org/rfc/rfc3208.txt, December 2001.

[49] L. Vicisano, L. Rizzo, and J. Crowcroft. TCP-like Congestion Control for Layered Multicast Data Transfer. In *Proceedings IEEE INFOCOM'98*, March 1998.

[50] B. Vickers, C. Albuquerque, and T. Suda. Source-Adaptive Multi-Layered Multicast Algorithms for Real-Time Video Distribution. *IEEE/ACM Transactions on Networking*, December 2000.

[51] D. Waitzman, C. Partridge, and S. Deering. Distance Vector Multicast Routing Protocol. *RFC 1075*, November 1988.

[52] M. Waldvogel, G. Caronni, D. Sun, N. Weiler, and B. Plattner. The VersaKey Framework: Versatile Group Key Management. *IEEE Journal on Selected Areas in Communications*, 17(9):1614–1631, September 1999.

[53] J. Widmer and M. Handley. Extending Equation-Based Congestion Control to Multicast Applications. In *Proceedings ACM SIGCOMM 2001*, August 2001.

Vita

Sergey Gorinsky, or Sergey Vladimirovich Gorinsky with the patronymic included according to the Russian tradition, was born in the Zelenograd hospital of Moscow, Russia on June 30, 1971. Official documents move his birthplace to the nearby town of Skhodnya where Sergey lived with his parents Vladimir Sergeevich Gorinsky and Galina Ivanovna Gorinsky (nee Ivanova) for the next 23 years. In 1988, Sergey Gorinsky graduated from School #2 of his hometown with a Silver Medal and was accepted to Moscow Institute of Electronic Technology (MIET) in Zelenograd. After completing five and a half years of studies at MIET with Highest Honors in 1994, he received the degree of Engineer in computer engineering and left for the US to attend New Jersey Institute of Technology (NJIT). In 1997, Sergey migrated to Texas to continue his doctoral studies in computer science at the University of Texas at Austin. Twice though New Jersey lured him back for summer internship at Lucent Bell Labs (1998) and AT&T Labs Research (1999). On his way to the Ph.D. degree, Sergey earned the Master of Science degree in 1999. He advanced to the Ph.D. candidacy after defending his dissertation proposal in 2000 with the late Edsger Dijkstra in memorable attendance. On February 7, 2002, Sergey married his Hungarian friend Andrea Nemeth on the Hawaiian island of Maui. In 2003, Sergey Gorinsky accepted an offer to join Washington University in St. Louis as an Assistant Professor.

Permanent Address: 36 Frunze Street, Apartment 23
Skhodnya, Moscow Oblast 141420
Russia

This dissertation was typeset with $\LaTeX\,2_\varepsilon$ by the author.