

# Robustness to Inflated Subscription in Multicast Congestion Control\*

Sergey Gorinsky

Sugat Jain

Harrick Vin

Yongguang Zhang

Laboratory for Advanced Systems Research  
Department of Computer Sciences, University of Texas at Austin  
gorinsky@cs.utexas.edu    sugat@cs.utexas.edu    vin@cs.utexas.edu

HRL Laboratories, LLC  
Malibu, California  
ygz@hrl.com

## ABSTRACT

Group subscription is a useful mechanism for multicast congestion control: RLM, RLC, FLID-DL, and WEBRC form a promising line of multi-group protocols where receivers provide no feedback to the sender but control congestion via group membership regulation. Unfortunately, the group subscription mechanism also offers receivers an opportunity to elicit self-beneficial bandwidth allocations. In particular, a misbehaving receiver can ignore guidelines for group subscription and choose an unfairly high subscription level in a multi-group multicast session. This poses a serious threat to fairness of bandwidth allocation. In this paper, we present the first solution for the problem of inflated subscription. Our 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.

## Categories and Subject Descriptors

C.2 [Computer-Communication Networks]: Network Architecture and Design

## General Terms

Security, Performance, Design

## Keywords

Congestion Control, Multicast, Fair Bandwidth Allocation, Misbehaving Receivers, Robustness

\*This research was supported in part by grants from the National Science Foundation (NSF ANI-0082294) and Intel.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

SIGCOMM'03, August 25–29, 2003, Karlsruhe, Germany.  
Copyright 2003 ACM 1-58113-735-4/03/0008 ...\$5.00.

## 1. INTRODUCTION

Traditionally, congestion control protocols *trust* receivers and assume their commitment to fair bandwidth sharing. Unfortunately, due to the growth and commercialization of the Internet, this assumption is no longer tenable. Whereas information sources and network providers have an interest in treating their customers fairly, a receiver is primarily interested in maximizing its own throughput. Hence, the receiver may misbehave to acquire unfairly high bandwidth at the expense of competing traffic. Furthermore, open-source operating systems provide receivers with ample opportunities for misbehavior. Thus, *robustness* of congestion control to receiver misbehavior becomes a pressing problem.

Multicast is a service for scalable dissemination of data to a group of receivers. In IP multicast [10, 15], a receiver subscribes to a multicast group by submitting the group address to the local edge router via IGMP [12], and the network organizes its routers in a logical tree that distributes packets from the sender to the subscribed receivers. A single multicast group, however, is often ineffective in accommodating the diverse capabilities of receivers. To satisfy heterogeneous receiving capabilities, multicast sessions often include multiple groups. This allows a receiver to align the received rate with its capability by subscribing to a suitable subset of the groups. In fact, group membership regulation has emerged as a dominant mechanism for multi-group multicast congestion control: RLM [20], RLC [28], FLID-DL [5], and WEBRC [18] form a promising line of multi-group protocols where receivers control congestion primarily through appropriate group subscription.

Unfortunately, the group subscription mechanism also offers to receivers an opportunity to elicit self-beneficial bandwidth allocations. In particular, a misbehaving receiver can ignore subscription guidelines and raise its subscription unfairly. To understand the significance of this misbehavior, consider a setting where receivers F1 and F2 from different FLID-DL sessions share a 1 Mbps bottleneck link with two TCP Reno [2] receivers T1 and T2. We simulate this scenario using NS-2 [23] and a topology described in Section 5. After 100 seconds into the simulation, receiver F1 starts to misbehave and inflates its subscription in violation of the protocol. As Figure 1 illustrates, such a misbehavior boosts the throughput of F1 to 690 Kbps at the expense of well-behaving receivers F2, T1, and T2.

In this paper, we present DELTA and SIGMA, the first solution for the problem of inflated subscription. First, we argue that prevention of inflated subscription requires re-

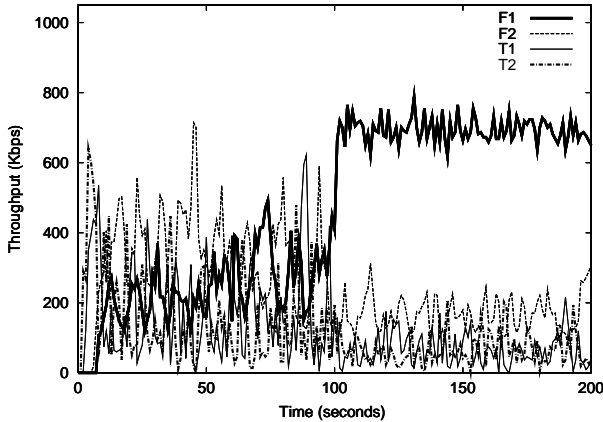


Figure 1: Impact of inflated subscription.

stricted group access. Then, we show that existing architectures for group access control – such as Secure IGMP [3] and Gothic [16] – 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. DELTA and SIGMA use dynamic keys to enforce *congestion-dependent* group access. Our design requires only minimal generic changes in the edge routers, does not alter the core of the network, and introduces no auxiliary servers. Integration with DELTA and SIGMA makes multicast protocols robust to inflated subscription and preserves other congestion control properties. We illustrate this by deriving and evaluating FLID-DS, a robust adaptation of FLID-DL.

The rest of the paper is organized as follows. Section 2 formulates the inflated subscription problem, our assumptions and design requirements. Section 3 describes DELTA and SIGMA. Section 4 discusses properties of our design. Section 5 evaluates the protection offered by DELTA and SIGMA. Section 6 examines the applicability of the proposed approach in end-system multicast. Finally, Section 7 summarizes our contributions.

## 2. PROBLEM FORMULATION

In this section, we first position the problem of inflated subscription within the area of bandwidth attacks. Then, we argue that protection against inflated subscription should rely on network-supported access control where the congestion status of a receiver determines its eligibility to access a multicast group. We describe the requirements for designing such access control and discuss our assumptions.

### 2.1 Threat Model

We examine attacks on congestion control where a multicast receiver abuses the group subscription mechanism to elicit a self-beneficial bandwidth allocation. Although the unfair bandwidth advantage comes at the expense of competing traffic, there are important differences between these *self-beneficial attacks* and denial-of-service attacks.

First, disruption of network services is a sole goal in denial-of-service attacks. 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 net-

work is potentially under attack. On the other hand, a self-beneficial attacker is primarily concerned with increasing its own bandwidth consumption. The damage to other communications 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.

Second, denial-of-service attacks enjoy a 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 – 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 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. We are not aware of any study quantifying the extent and impact of bandwidth cheating. 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. Studies of TCP congestion control show that a misbehaving receiver can substantially increase its throughput at the expense of cross traffic [11, 25]. Thus, even if a small percentage of unicast receivers launches self-beneficial attacks, this misbehavior can severely disrupt network services. In multicast congestion control, self-beneficial receiver attacks are more diverse and pose even more potent threats to the network [13].

In our trust model, a misbehaving receiver only seeks a self-beneficial bandwidth allocation but does not act from pure malice to stage denial-of-service attacks. Local interfaces of edge routers are the only points of access for network users. For instance, a receiver can subscribe to a multicast group only by communicating with a local router. We assume that information sources and network providers (and hence network routers) are trustworthy and always adhere to their protocols. Note that the trust in routers is essential for fair bandwidth allocation because a router has the last word in allocating the bandwidth of its output links.

### 2.2 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 [19]. In this paper, however, we focus only on mechanisms that *prevent* receivers from inflating their subscription.

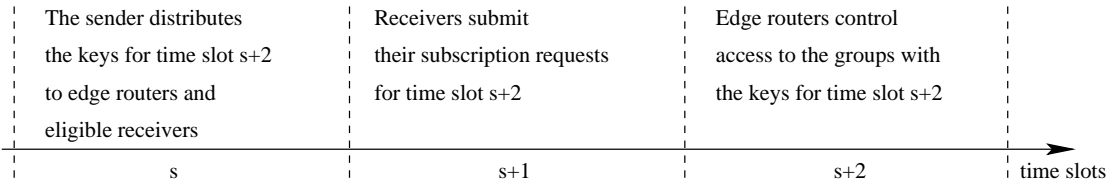


Figure 2: Timeline for distribution and usage of keys.

Since the Internet Group Management Protocol (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 [22] – 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 [16] – 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 receiver congestion status – rather than the receiver 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.*

Observe that the required generality of network support poses a challenge because different multi-group protocols specify different rules for group subscription. For instance, in replicated multicast protocols [8, 9], 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 [5, 14, 17, 18, 20, 27, 28], 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 et al [6] also propose a non-cumulative variation of layered multicast where any combination of the groups in a session constitutes a legitimate subscription level. Furthermore, while some protocols [5, 28] reduce subscription in response to a single packet loss, others [18] monitor a long-term history of losses to determine the fair subscription level. Similarly, while some protocols rely on packet loss as a congestion signal, others employ explicit congestion notification. 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. DESIGN

Our objective is to design group access control based on the congestion status of receivers. 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. However, 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 2 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

	Keys		
	with upgrade authorization	without upgrade authorization	
Maximal group	$\tau$ $\sigma$	$\tau$	top key: $\tau$
Each intermediate group	$\tau$ $\delta$ $\sigma$	$\tau$ $\delta$	decrease key: $\delta$
Minimal group	$\tau$ $\delta$	$\tau$ $\delta$	increase key: $\sigma$

Figure 3: Group keys in the presented DELTA instantiations.

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.

### 3.1 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.

#### 3.1.1 Example of a DELTA Instantiation

FLID-DL [5] and RLC [28] 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,  $\dots$ , 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: ( $\nabla$ ) a congested receiver of  $g + 1$  groups should not obtain an updated key for group  $g + 1$ , and ( $\Delta$ ) 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 ( $\nabla$ ), 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 ( $\Delta$ ). 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:

$$k_g = F(c_{1,1}, \dots, c_{1,n_1}, \dots, c_{g,1}, \dots, c_{g,n_g}) \quad (1)$$

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

	Sender		Receiver	
Input	$N$	number of groups in the session	$g$	current top group
	$S_g$	set of packets for group $g$ where $1 \leq g \leq N$	$R_j$	set of packets received from group $j$
	$l_g$	last packet for group $g$ where $1 \leq g \leq N$		where $1 \leq j \leq g$
Algorithm	<pre>// precomputation of keys and decrease fields for <math>g = 1, \dots, N</math>   <math>C_g \leftarrow</math> nonce;   <math>\tau_1 \leftarrow C_1</math>; for <math>g = 2, \dots, N</math>   <math>\tau_g \leftarrow \tau_{g-1} \oplus C_g</math>; <math>\delta_{g-1} \leftarrow</math> nonce; <math>d_g \leftarrow \delta_{g-1}</math>;   if the protocol authorizes an upgrade to group <math>g</math>     then <math>\sigma_g \leftarrow \tau_{g-1}</math>; // real-time generation of component fields if <math>p \in S_g</math> and <math>p \neq l_g</math> then <math>c_{g,p} \leftarrow</math> nonce; <math>C_g \leftarrow C_g \oplus c_{g,p}</math>; if <math>p = l_g</math> then <math>c_{g,p} \leftarrow C_g</math>.</pre>		<pre>for <math>j = 2, \dots, g</math>   <math>u_{j-1} \leftarrow</math> decrease field from <math>R_j</math>; if the receiver is congested   then if <math>g = 1</math>     then <math>n \leftarrow</math> null;     else <math>n \leftarrow g - 1</math>;   else <math>u_g \leftarrow \bigoplus_{j=1}^g \bigoplus_{r \in R_j} \text{component field from } r</math>;   if the protocol authorizes an upgrade   to group <math>g + 1</math>     then <math>n \leftarrow g + 1</math>; <math>u_{g+1} \leftarrow u_g</math>;     else <math>n \leftarrow g</math>.</pre>	
Output	$c_{g,p}$	component field for packet $p$ in $S_g$ where $1 \leq g \leq N$	$n$	next top group
	$d_g$	decrease field for group $g$ where $2 \leq g \leq N$	$u_j$	updated key for group $j$ where $1 \leq j \leq n$
	$\tau_g$	top key for group $g$ where $1 \leq g \leq N$		
	$\delta_g$	decrease key for group $g$ where $1 \leq g \leq N - 1$		
	$\sigma_g$	increase key for group $g$ where $2 \leq g \leq N$		

Figure 4: DELTA instantiation for layered multicast protocols that define congestion as a single packet loss.

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 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  $1 \leq j \leq g-1$ :

$$k_g = H(i_{1,1}, \dots, i_{1,n_1}, \dots, i_{g-1,1}, \dots, i_{g-1,n_{g-1}}). \quad (2)$$

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 2 when only groups 1 through  $g-2$  lose packets. Hence,  $H$  should also be a one-way function.

To reconcile Equations 1 and 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 Figure 3). The sender communicates these keys to receivers by adding component fields and decrease fields to multicast packets. We define top key  $\tau_g$  for each group  $g$  as:

$$\tau_g = \bigoplus_{j=1}^g \bigoplus_{p \in S_j} c_{j,p} \quad (3)$$

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 to compute key  $\tau_g$ .

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

$$\delta_j = d_{j+1} \quad (4)$$

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  $\delta_1$  through  $\delta_{g-1}$  via Equation 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  $\sigma_m$  for group  $m$  as:

$$\sigma_m = \bigoplus_{j=1}^{m-1} \bigoplus_{p \in S_j} c_{j,p} \quad (5)$$

	Sender		Receiver	
Input	$N$	number of groups in the session	$g$	current group
	$S_g$	set of packets for group $g$ where $1 \leq g \leq N$	$R_g$	set of packets received from group $g$
	$l_g$	last packet for group $g$ where $1 \leq g \leq N$		
Algorithm	<pre>// precomputation of keys and decrease fields for <math>g = 1, \dots, N</math>   <math>C_g \leftarrow</math> nonce; <math>\tau_g \leftarrow C_g</math>; for <math>g = 2, \dots, N</math>   <math>\delta_{g-1} \leftarrow</math> nonce; <math>d_g \leftarrow \delta_{g-1}</math>;   if the protocol authorizes an upgrade to group <math>g</math>     then <math>\sigma_g \leftarrow \tau_{g-1}</math>; // real-time generation of component fields if <math>p \in S_g</math> and <math>p \neq l_g</math> then <math>c_{g,p} \leftarrow</math> nonce; <math>C_g \leftarrow C_g \oplus c_{g,p}</math>; if <math>p = l_g</math> then <math>c_{g,p} \leftarrow C_g</math>.</pre>		<pre>if the receiver is congested   then if <math>g = 1</math>     then <math>n \leftarrow</math> null;     else <math>n \leftarrow g - 1</math>;     <math>u_{g-1} \leftarrow</math> decrease field from <math>R_g</math>;   else <math>u_g \leftarrow \bigoplus_{r \in R_g}</math> component field from <math>r</math>;   if the protocol authorizes an upgrade   to group <math>g + 1</math>     then <math>n \leftarrow g + 1</math>; <math>u_{g+1} \leftarrow u_g</math>;     else <math>n \leftarrow g</math>.</pre>	
	Output	$c_{g,p}$	component field for packet $p$ in $S_g$ where $1 \leq g \leq N$	$n$
$d_g$		decrease field for group $g$ where $2 \leq g \leq N$	$u_j$	updated key for group $n$
$\tau_g$		top key for group $g$ where $1 \leq g \leq N$		
$\delta_g$		decrease key for group $g$ where $1 \leq g \leq N - 1$		
$\sigma_g$		increase key for group $g$ where $2 \leq g \leq N$		

Figure 5: DELTA instantiation for replicated multicast protocols.

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

Figure 4 presents 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.

### 3.1.2 Instantiations for Other Types of Protocols

In Section 3.1.1, we derived a DELTA instantiation that protects FLID-DL, RLC, and similar unreliable protocols for cumulative layered multicast where congestion is defined as a single packet loss. To protect protocols of other types, we extend the presented approach along the following four dimensions: (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 [7]. 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 mark forwarded packets to indicate congestion explicitly. To extend the protection to protocols that exploit such explicit congestion notification (ECN), 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,  $\dots$ , and maximal group  $N$  is the fastest. Let us now consider a replicated multicast protocol that

differs from the protocol in Section 3.1.1 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 this paper deals only with 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 Figure 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:

$$\tau_g = \bigoplus_{p \in S_g} c_{g,p} \quad \text{and} \quad \sigma_g = \bigoplus_{p \in S_{g-1}} c_{g-1,p}. \quad (6)$$

**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 instance, the default threshold for each subscription level in RLM [20] is 25% of the packets transmitted to this level. MLDA [27] and WEBRC [18] are examples of protocols that lower the threshold for each higher subscription level and thereby define a fair subscription level for each loss rate.

For extending the protection to threshold-based protocols, DELTA can use Shamir’s  $(k, n)$  threshold scheme [26] to distribute components of key  $\tau_g$  for subscription level  $g$  among all  $n$  packets transmitted to this level – the sender uses modular arithmetic, picks a polynomial  $q(x)$  of degree  $k - 1$ :

$$q(x) = \tau_g + a_1x + \dots + a_{k-1}x^{k-1}, \quad (7)$$

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

$$c_p = (p, q(p)) \quad (8)$$

where  $p = 1, \dots, n$ . 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:

$$\tau_g = q(0). \quad (9)$$

In layered multicast, subscription levels share groups. Unfortunately, Shamir’s scheme does not enable a reuse of the components from lower subscription levels and, therefore, has high communication overhead. Design of more efficient threshold schemes that reuse components remains an open research problem.

### 3.2 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 2, 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.

#### 3.2.1 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 protocol from Section 3.1.1 does not authorize an upgrade to an intermediate group  $g$ , SIGMA communicates a tuple that links the address of group  $g$  with **top** key  $\tau_g$  and **decrease** key  $\delta_g$ ; if the protocol authorizes the upgrade, the tuple for group  $g$  also contains **increase** key  $\sigma_g$ . 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.

#### 3.2.2 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

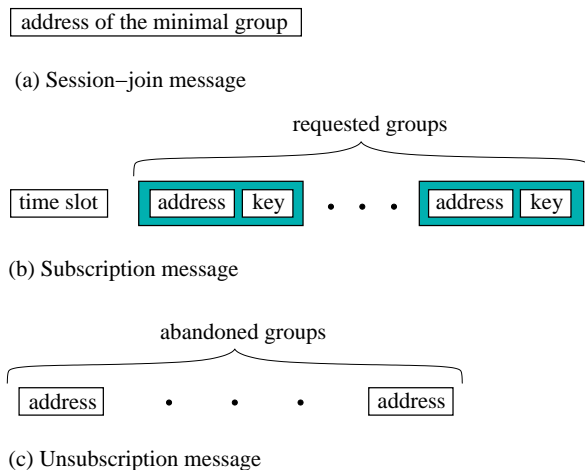


Figure 6: SIGMA messages sent by receivers.

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 2). 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 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 6(b) shows the general format of subscription messages. To ensure reliable subscription, the edge 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 **unsubscribe** message that contains the addresses of the abandoned groups (see Figure 6(c)). When a receiver leaves a group, its unsubscribe 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.

### 3.2.3 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.

## 4. PROPERTIES OF DELTA AND SIGMA

### 4.1 Revisiting the Design Requirements

We start an assessment of our design by revisiting the requirements from Section 2 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 Section 5.3, we experimentally verify that DELTA and SIGMA also preserve other congestion control properties of protected protocols.

### 4.2 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 [24, 29], and (b) hop-by-hop or edge-to-edge encryption to protect against snooping on network links [21].

**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 3.1, keys and components have the same size, and the component guessing gives no advantage over guessing the key directly.

In this paper, we focused on self-beneficial misbehavior of individual receivers. The presented DELTA instantiations are vulnerable to attacks where receivers of a multicast session collude to pass keys from a more capable receiver to a less capable receiver that is unable to reconstruct these keys on its own. One design that is robust to such collusion attacks guards each local interface with a different set of keys. For example, to protect the protocol described in Section 3.1.1, edge routers can make a disseminated key interface-specific by altering randomly the forwarded components of the key. Then, the edge router computes two values: an **upper** key from the original components, and a **lower** key from the modified components. Only if the **upper** key matches a valid key provided by SIGMA, and the interface-specific **lower** key matches a key provided from the interface, the edge router opens access to the group. However, this approach is protocol-specific and sacrifices the generality of SIGMA. Design of efficient and general techniques for protecting against collusion attacks is a topic for future research.

## 5. EVALUATION

This section evaluates DELTA and SIGMA both experimentally and analytically.

### 5.1 Experimental Settings

We implemented DELTA and SIGMA in NS-2 [23] and integrated these implementations with FLID-DL. We refer to the integrated protocol as FLID-DS.

We conduct experiments in a single-bottleneck topology. 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

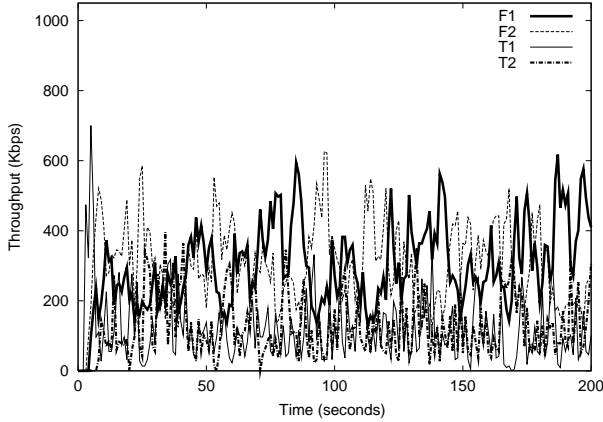


Figure 7: Protection with DELTA and SIGMA.

of 1.5 per group. The time slot duration for FLID-DL is set to its default value of 500 ms [5]. To provide the same congestion control granularity in FLID-DS, we set the time slot duration for FLID-DS to 250 ms (because SIGMA enforces group access with responsiveness of two time slots). All data traffic uses 576-byte packets.

## 5.2 Preventing Inflated Subscription

According to the results presented earlier in Figure 1, a FLID-DL receiver can inflate its subscription to acquire unfairly high throughput at the expense of competing traffic. We now repeat this experiment for FLID-DS: after 100 seconds, F1 tries to inflate its subscription. Figure 7 shows that DELTA and SIGMA preserve the fair bandwidth allocation.

## 5.3 Preserving Congestion Control Properties

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

**Impact on throughput.** In the experiments of this section, 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 8(a) and 8(b) report individual throughput and average throughput (averaged over the number of sessions) for FLID-DL and FLID-DS receivers respectively. Figure 8(c) shows that the receivers achieve similar average throughput in FLID-DL and FLID-DS.

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 8(d) 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 8(e) 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 8(f) 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 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 8(g) and 8(h) indicate, the receivers converge to the same fair subscription both in FLID-DL and FLID-DS.

## 5.4 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 3.1.1 – and SIGMA.

Let group 1 of a FLID-DS session transmit data at rate  $r$ . If the cumulative transmission rate grows multiplicatively with the factor of  $m$  per group, the session transmits data at cumulative rate  $R$ :

$$R = r \cdot m^{N-1} \quad (10)$$

where  $N$  is the number of groups in the session.

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

$$P = \frac{R \cdot t}{s} = \frac{r \cdot t \cdot m^{N-1}}{s} \quad (11)$$

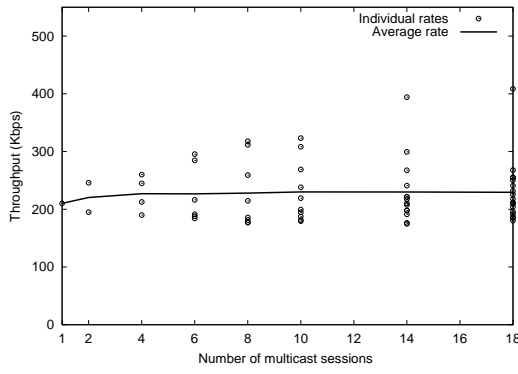
packets during a time slot of duration  $t$ , and

$$p = \frac{r \cdot t}{s} \quad (12)$$

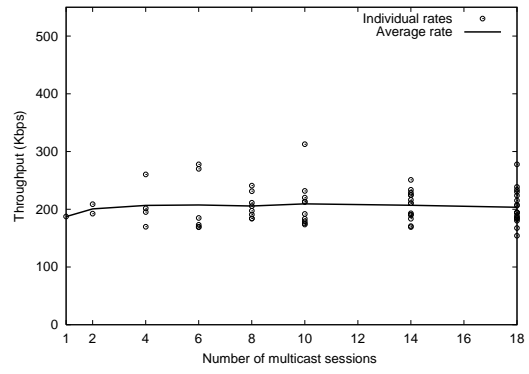
of these packets belong to group 1.

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 group  $g$  such that  $2 \leq g \leq N$ . Therefore, communication overhead  $O_\Delta$  imposed by DELTA equals:

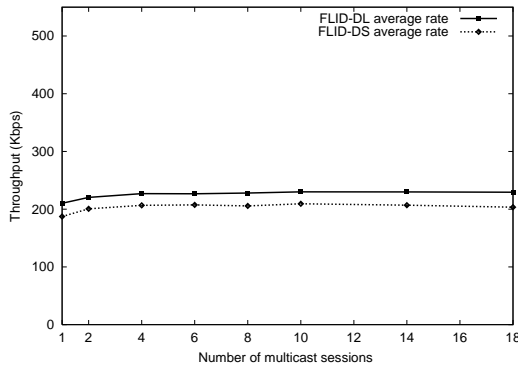
$$\begin{aligned} O_\Delta &= \left\{ \text{ratio of the DELTA bits to the data bits} \right\} \\ &= \frac{(2P - p)b}{R \cdot t} \\ &= \left\{ (10), (11), \text{ and } (12) \right\} \\ &= \frac{(2r \cdot t \cdot m^{N-1} - \frac{r \cdot t}{s})b}{r \cdot t \cdot m^{N-1}} \\ &= \left\{ \text{simplification} \right\} \\ &= \left( 2 - \frac{1}{m^{N-1}} \right) \frac{b}{s}. \end{aligned}$$



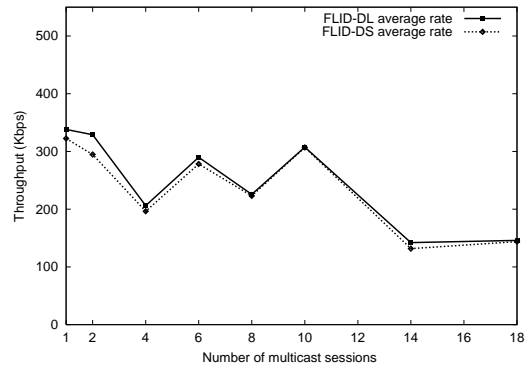
(a) Throughput for FLID-DL without cross traffic



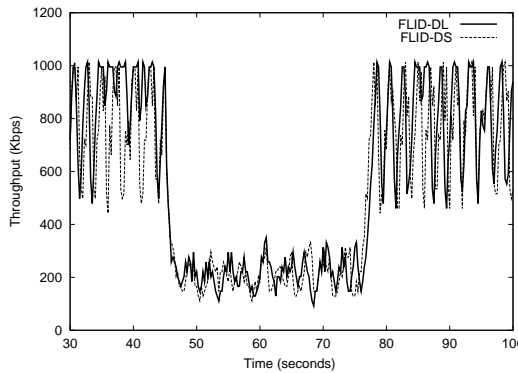
(b) Throughput for FLID-DS without cross traffic



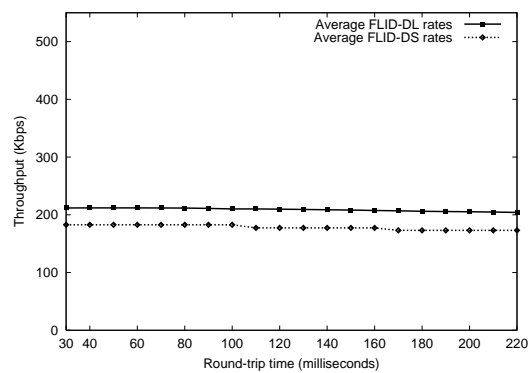
(c) Average throughput without cross traffic



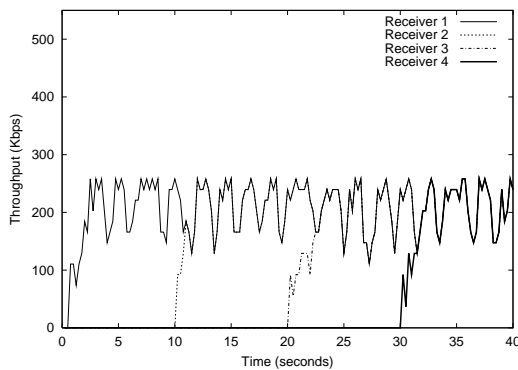
(d) Average throughput with cross traffic



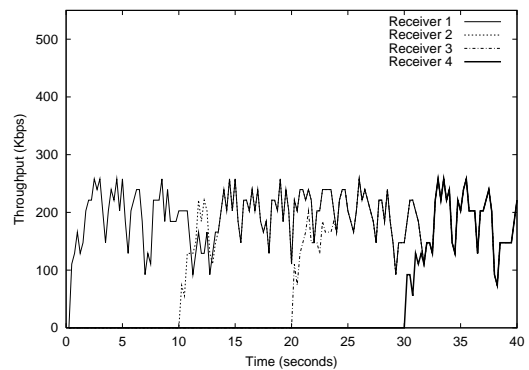
(e) Responsiveness



(f) Heterogeneous round-trip times



(g) Subscription convergence in FLID-DL



(h) Subscription convergence in FLID-DS

**Figure 8: Preservation of congestion control properties.**

SIGMA communicates the keys to edge routers via special packets. For each time slot, these packets deliver an  $l$ -bit slot number and one address-key tuple per group. Every tuple contains a 32-bit multicast address and  $b$ -bit top key. Each of the tuples for groups 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 group  $g$ , the tuple for this group 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  $O_\Sigma$  imposed by SIGMA is equal to:

$$\begin{aligned}
O_\Sigma &= \{ \text{ratio of the SIGMA bits to the data bits} \} \\
&= \frac{\left( l + (32 + b)N + b(N - 1) + b \sum_{g=2}^N f_g \right) z + h}{R \cdot t} \\
&= \{ (10) \text{ and simplification} \} \\
&= \frac{\left( l + 32N + b(2N - 1 + \sum_{g=2}^N f_g) \right) z + h}{r \cdot t \cdot m^{N-1}}.
\end{aligned}$$

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 minimal group 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 (10). 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 groups. Figure 9(a) shows  $O_\Delta$  and  $O_\Sigma$  for  $N$  varying from 2 to 20 when  $t = 250$  ms. Then, we examine the impact of the time slot duration. Figure 9(b) plots  $O_\Delta$  and  $O_\Sigma$  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.

## 6. DISCUSSION

Whereas this paper proposes robust mechanisms for IP multicast congestion control, slow deployment of IP multicast has stirred an interest in *end-system multicast*. 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 forwarding 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 forwarding hierarchies [4]. Then, the path from the sender to a receiver can traverse another receiver that forwards the 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 im-

prove 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 trusted intermediaries, our design is insufficient for providing robust congestion control in peer-to-peer architectures. One promising approach for robust peer-to-peer multicast is to give a receiver some control over choosing the peers on its path. The receiver-guided path formation faces, however, the following challenges:

- A misbehaving receiver can seek a self-beneficial forwarding 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 hierarchy.
- A misbehaving intermediary is difficult to detect. Due to heterogeneous network conditions, a receiver has no easy ways to verify the fairness of the received rate. For instance, the receiver cannot rely on comparing the rates from disjoint paths because the fair rates for such paths can be different.

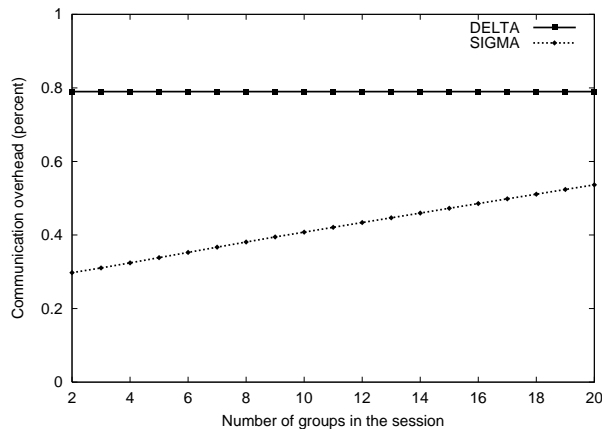
We plan to address the above challenges in our future architecture for robust peer-to-peer multicast.

## 7. CONCLUSION

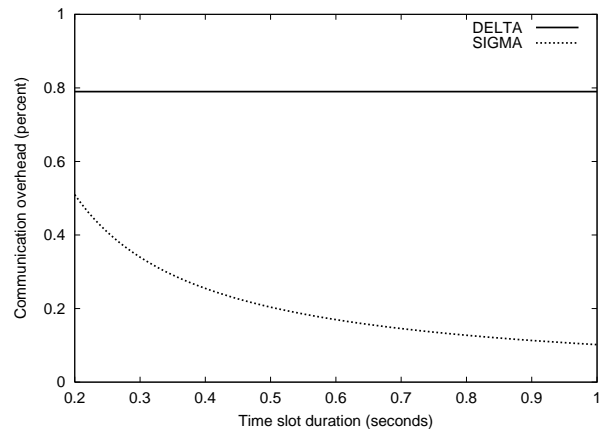
Group subscription is a useful mechanism for multicast congestion control. Unfortunately, this mechanism also provides receivers with opportunities to inflate subscription and thereby acquire unfairly high bandwidth. In this paper, we presented DELTA and SIGMA, the first solution for the problem of inflated subscription. DELTA and SIGMA use dynamic keys to enforce *congestion-dependent* group access. Our design requires only minimal generic changes in the edge routers, does not alter the core of the network, and introduces no auxiliary servers. Integration with DELTA and SIGMA makes multicast protocols robust to inflated subscription and preserves other congestion control properties. We illustrated this by deriving and evaluating FLID-DS, a robust adaptation of FLID-DL.

## 8. REFERENCES

- [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] S. Banerjee, B. Bhattacharjee, and C. Kommareddy. Scalable Application Layer Multicast. In *Proceedings of ACM SIGCOMM 2002*, August 2002.
- [5] 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.
- [6] J. Byers, M. Luby, and M. Mitzenmacher. Fine-Grained Layered Multicast. In *Proceedings IEEE INFOCOM 2001*, April 2001.
- [7] J.W. Byers, M. Luby, M. Mitzenmacher, and A. Rege. A Digital Fountain Approach to Reliable Distribution



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



(b) Dependence on the time slot duration

**Figure 9: Overhead for communicating the group keys via DELTA and SIGMA.**

- of Bulk Data. In *Proceedings ACM SIGCOMM'98*, September 1998.
- [8] 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.
- [9] 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.
- [10] S.E. Deering. *Multicast Routing in a Datagram Internetwork*. PhD thesis, Stanford University, December 1991.
- [11] D. Ely, N. Spring, D. Wetherall, S. Savage, and T. Anderson. Robust Congestion Signaling. In *Proceedings IEEE ICNP 2001*, November 2001.
- [12] W. Fenner. Internet Group Management Protocol, Version 2. RFC 2236, November 1997.
- [13] S. Gorinsky, S. Jain, and H. Vin. Multicast Congestion Control with Distrusted Receivers. In *Proceedings NGC 2002*, October 2002.
- [14] 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.
- [15] 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.
- [16] P. Judge and M. Ammar. GOTHIC: A Group Access Control Architecture for Secure Multicast and Anycast. In *Proceedings IEEE INFOCOM 2002*, June 2002.
- [17] A. Legout and E. W. Biersack. PLM: Fast Convergence for Cumulative Layered Multicast Transmission Schemes. In *Proceedings ACM SIGMETRICS 2000*, June 2000.
- [18] 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.
- [19] R. Mahajan, S. Floyd, and D. Wetherall. Controlling High-Bandwidth Flows at the Congested Router. In *Proceedings IEEE ICNP 2001*, November 2001.
- [20] S. McCanne, V. Jacobson, and M. Vetterli. Receiver-driven Layered Multicast. In *Proceedings ACM SIGCOMM'96*, August 1996.
- [21] S. Mittra. Iolus: A Framework for Scalable Secure Multicasting. In *Proceedings ACM SIGCOMM'97*, September 1997.
- [22] MSTAT Manual Page. <http://man-pages.net/linux/man8/mstat.8.html>, April 2002.
- [23] UCB/LBNL/VINT Network Simulator NS-2. <http://www-mash.cs.berkeley.edu/ns>, May 2002.
- [24] A. Perrig, R. Canetti, D. Song, and D. Tygar. Efficient and Secure Source Authentication for Multicast. In *Proceedings NDSS 2001*, February 2001.
- [25] 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.
- [26] A. Shamir. How to Share a Secret. *Communications of the ACM*, 22(11):612–613, November 1979.
- [27] D. Sisalem and A. Wolisz. MLDA: A TCP-friendly Congestion Control Framework for Heterogenous Multicast Environments. In *Proceedings IWQoS 2000*, June 2000.
- [28] L. Vicisano, L. Rizzo, and J. Crowcroft. TCP-like Congestion Control for Layered Multicast Data Transfer. In *Proceedings IEEE INFOCOM'98*, March 1998.
- [29] 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.