Design of Multicast Protocols Robust Against Inflated Subscription

Sergey Gorinsky, *Member, IEEE*, Sugat Jain, Harrick Vin, *Member, IEEE*, and Yongguang Zhang, *Member, IEEE*

***Abstract*—To disseminate data to a heterogeneous body of receivers efficiently, congestion control protocols for IP multicast compose a session from several multicast groups and prescribe guidelines that enable each receiver to subscribe to an appropriate subset of the groups. However, a misbehaving receiver can ignore the group subscription rules and inflate its subscription to acquire unfairly high throughput. 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. We apply DELTA and SIGMA to derive robust versions of prominent RLM and FLID-DL protocols.**

***Index Terms*—Congestion control, fair bandwidth allocation, misbehaving receivers, multicast, robust communication protocols.**

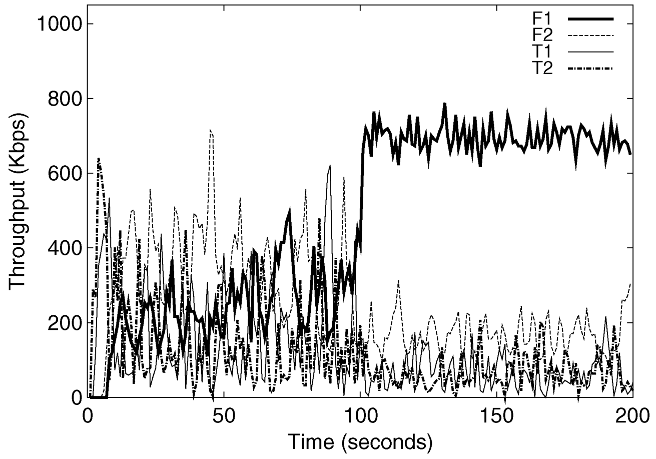


Fig. 1. Impact of inflated subscription.

I. Introduction

Traditional protocols for congestion control assume that each receiver is trustworthy and obeys rules for fair sharing of the network capacity. Unfortunately, because of 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, a 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 [9], [15], a receiver subscribes to a multicast group by submitting the group address to the local edge router via IGMP [11], 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 [29], FLID-DL [4], 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-Mb/s bottleneck link with two TCP Reno [1] receivers `T1` and `T2`. We simulate this scenario using NS-2 [23] and a topology described in Section IV-A. After 100 seconds into the simulation, receiver `F1` starts to misbehave and inflates its subscription in violation of the protocol. As Fig. 1 illustrates, such a misbehavior boosts the throughput of `F1` to 690 Kb/s 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 restricted group access. Then, we show that existing architectures for group access control—such as Secure IGMP [2] and Gothic [16]—do not protect against inflated subscription because they define the eligibility to access a group based on the *identity*, rather than

Manuscript received March 23, 2004; revised August 22, 2004; approved by IEEE/ACM Transactions on Networking Editor D. Wetherall. This work was supported in part by grants from the National Science Foundation (NSF ANI-0082294) and Intel Corporation.

S. Gorinsky is with the with the Applied Research Laboratory, Department of Computer Science and Engineering, Washington University in St. Louis, St. Louis, MO 63130 USA (e-mail: gorinsky@wustl.edu).

S. Jain and H. Vin are with the with the Laboratory for Advanced Systems Research, Department of Computer Sciences, University of Texas at Austin, Austin, TX 78712 USA (e-mail: sugat@cs.utexas.edu; vin@cs.utexas.edu).

Y. Zhang is with HRL Laboratories, LLC, Malibu, CA 90265 USA (e-mail: ygz@hrl.com).

Digital Object Identifier 10.1109/TNET.2006.872573

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 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 robust adaptations of FLID-DL and RLM protocols.

The rest of the paper is organized as follows. Section II formulates the inflated subscription problem, our assumptions and design requirements. Section III describes DELTA and SIGMA. Section IV derives and evaluates robust versions of two prominent protocols FLID-DL and RLM. Finally, Section V summarizes our contributions and examines applicability of the proposed approach in end-system multicast.

II. 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.

A. 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. 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 competing traffic, the attacker has incentives to subdue this traffic to a level that the abused parties can falsely interpret as fair. Stealth of such abuse can make self-beneficial attacks difficult to discern.

Second, denial-of-service attacks enjoy a rich 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. These attacks are purely malicious; the attacker itself does not benefit from the wasted bandwidth. Opportunities for self-beneficial attacks are less ample: for instance, a receiver can acquire useful data from a session at an unfairly high rate by manipulating the congestion control protocol of the session. 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 might be possible to *prevent* self-beneficial attacks.

In contrast to widely publicized denial-of-service incidents, insidious self-beneficial attacks have received significantly less attention from researchers. On the other hand, sneaky self-beneficial misbehavior is far from harmless. In the Internet, the population of bandwidth-greedy users can greatly exceed the number of hackers interested only in disrupting the communications of others. Even inside intra-enterprise networks, selfish misbehavior cannot be discounted. Studies of TCP congestion control show that a misbehaving receiver can substantially increase its throughput at the expense of cross traffic [10], [26]. In comparison to unicast, self-beneficial attacks of multicast receivers are more diverse and pose additional threats to the network [12]. Furthermore, tools that accelerate downloads by renouncing fair congestion control are becoming available. Even if a small percentage of receivers utilize bandwidth-hogging tools and launch self-beneficial attacks, this misbehavior can severely disrupt network services. Quantifying the extent and significance of self-beneficial attacks is an important topic for future research.

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 [13]. We categorize potential self-beneficial misbehavior further into *individual attacks* and *collusion attacks* depending on whether the receiver misbehaves alone or inflates its subscription level in the multicast session with assistance from other receivers.

We assume that local interfaces of edge routers are the only access points for network users. For instance, a receiver can subscribe to a multicast group only by communicating with a local router. We consider information sources and network providers (and therefore network routers) as trustworthy and always adhering 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.

B. 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.

Since IGMP does not restrict the ability of receivers to subscribe to multicast groups, a misbehaving receiver can join any 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., group addresses) from ineligible receivers. Unfortunately, such information hiding is difficult to realize in modern networks: since group addresses are employed for routing, receivers can abuse network monitoring and debugging tools to query routers and obtain the addresses of active groups.

Based on these arguments, we conclude that to restrict group subscription only to eligible receivers, multicast protocols *must* regulate access to groups. Existing architectures for group access control—such as Secure IGMP [2] and Gothic [16]—rely on receiver authentication. Unfortunately, the identity of a receiver does not reveal any information about its congestion status. Thus, conventional group access control mechanisms are inadequate for preventing inflated subscription. Instead,

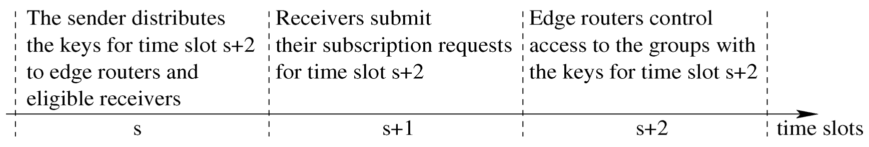

The sender distributes the keys for time slot s+2 to edge routers and eligible receivers
Receivers submit their subscription requests for time slot s+2
Edge routers control access to the groups with the keys for time slot s+2
s
s+1
s+2
time slots

Fig. 2. Timeline for distribution and usage of keys.

multicast 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 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 because different multi-group protocols specify different rules for group subscription [5], [7], [8], [14], [17], [28]. 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 explicit congestion notification (ECN) [25]. 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 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 scalability, fairness, efficiency, responsiveness, and other congestion control properties of multicast protocols.

III. Design

First, we focus on individual attacks of inflated subscription and present DELTA and SIGMA in Section III-A. Then, Section III-B extends the design to make it robust to collusion attacks.

A. 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. However, such schemes violate our Requirement 3 because they make routers aware of the session, its groups, and its congestion control protocol. Instead, we select a 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 is independent of a specific congestion control protocol.

Since network conditions change, congestion-based group access control should also be dynamic. We define a *time slot* as a period during which a group key remains valid. The sender updates all 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. Fig. 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 enough time to reconstruct the keys and submit them to the local edge router for validation before packets from time slot $s+2$ start reaching the router. The duration of time slots is determined by the intended responsiveness of the congestion control protocol. For instance, to support the responsiveness of FLID-DL in its default setting, group keys remain valid for 250 ms.

Since the eligibility to access a group depends on the congestion control protocol, distribution of keys to receivers is also 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.

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, most of the

	Keys that open access to a group	
	with upgrade authorization	without upgrade authorization
Maximal group	$\boxed{\tau}$ or $\boxed{\sigma}$	$\boxed{\tau}$
Each intermediate group	$\boxed{\tau}$ or $\boxed{\delta}$ or $\boxed{\sigma}$	$\boxed{\tau}$ or $\boxed{\delta}$
Minimal group	$\boxed{\tau}$ or $\boxed{\delta}$	$\boxed{\tau}$ or $\boxed{\delta}$

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

Fig. 3. Keys that open access to multicast groups in our DELTA instantiations.

protocols 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*; 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.

A) Example of a DELTA Instantiation: FLID-DL [4] and RLC [29] 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*; 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 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. Note, however, that allowing a receiver to ignore congestion of its top group might affect the overall protocol behavior.

After resolving the contradiction, 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 $g+1$.

To satisfy the above conditions, our DELTA instantiation enhances each packet with a *nonce*, a random number that the sender computes with minimal effort and uses once [21]. As in solutions for robust unicast [10], [26], we apply XOR to blend nonces of multiple packets into a short key (e.g., our experiments below consider nonces and keys that are 8, 16, or 32 bits long).

In the absence of an upgrade authorization for group g, a receiver of g groups should obtain a key for group g only when the receiver gathers all packets from groups 1 through g. To enforce this, the sender attaches a nonce to each packet and defines a *top* key τ_g for every group g as

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

where $\oplus$ is an XOR operation, $c_{j,p}$ is a nonce placed into packet p of group j, and S_j is a set of packets sent to group j. If at least one of the packets fails to reach the receiver, the congested receiver is unable to reconstruct key τ_g from delivered components.

Building keys τ_1 through τ_N from entirely independent sets of nonces might result in high communication overhead because any packet of group g would need to carry a separate component for each key τ_g through τ_N, or $N-g+1$ components. To keep the overhead low, the sender places only one nonce $c_{g,p}$ into each packet p of group g and reuses this component when defining keys τ_g through τ_N.

A congested receiver of g groups should not obtain key τ_g but should preserve access to its lower $g-1$ groups. However, the component sharing makes keys τ_1 through τ_g dependent. In particular, since keys τ_{g-1} and τ_g are related as

$$\tau_g = \tau_{g-1} \oplus \left(\bigoplus_{p \in S_g} c_{g,p} \right), \qquad (2)$$

	Sender		Receiver	
Input	N	number of groups in the session	g	current top group
	S_g	set of packets for group g where $1 \le g \le N$	R_j	set of packets received from group j where $1 \le j \le g$
	l_g	last packet for group g where $1 \le g \le N$		

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 $c_{g,p} \leftarrow C_g$.
				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 $u_g \leftarrow \overset{g}{\underset{j=1}{\oplus}} \underset{r \in R_j}{\oplus}$ 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	$c_{g,p}$	component field for packet p in S_g where $1 \le g \le N$	n	next top group
	d_g	decrease field for group g where $2 \le g \le N$	u_j	updated key for group j where $1 \le j \le n$
	τ_g	top key for group g where $1 \le g \le N$		
	δ_g	decrease key for group g where $1 \le g \le N-1$		
	σ_g	increase key for group g where $2 \le g \le N$		

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

knowledge of τ_{g-1} would reveal τ_g when group g is not congested; therefore, the receiver should not obtain key τ_{g-1} unless groups 1 through $g-1$ lose no packets. To enable a congested receiver to access the lower groups without disclosing their *top* keys to the receiver, our DELTA instantiation introduces alternative *decrease* keys. For each group j from 1 to $N-1$, the sender creates a *decrease* key δ_j as a nonce and attaches it to every packet of group $j+1$. As long as a receiver of g groups gets at least one packet from each group 2 through g, the receiver obtains keys δ_1 through δ_{g-1} and can use them to access its lower groups. Note that *decrease* keys δ_1 through δ_{g-1} reveal no information about *top* key τ_g.

If one of groups 2 through g loses all its packets, a congested receiver of g groups does not obtain a *decrease* key for one of the lower groups and might need to reduce the 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 in the absence of upgrade authorizations.

When the protocol authorizes an upgrade to group g, a receiver of $g-1$ groups should obtain a valid key for group g only if the receiver is not congested. To support this, the sender defines an additional *increase* key σ_g for group g as

$$\sigma_g = \bigoplus_{j=1}^{g-1} \bigoplus_{p \in S_j} c_{j,p}. \tag{3}$$

Therefore, a receiver can access a group with a key from a set of up to three keys (*top* key, *decrease* key, and *increase* key) as shown in Fig. 3. To communicate the keys to eligible receivers, the sender enhances packets with up to two fields: a *component* field contains a nonce serving as a component for *top* and *increase* keys, and a *decrease* field carries the *decrease* key for the group immediately below.

Fig. 4 presents our algorithm for the in-band distribution of the keys. 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.

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

Session structure. In a replicated multicast session, unlike in layered multicast, each subscription level consists of a single group and provides the same content but at a different rate: minimal group 1 transmits at the lowest rate, group 2 has the second lowest rate, ..., and maximal group N provides the content at the highest rate. Let us now consider a replicated multicast protocol that differs from the above layered multicast protocol 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* $g-1$; 3) *when authorized, an uncongested receiver of group g can switch to group* $g+1$.

Note that an authorized uncongested receiver can disobey the above rules by subscribing to group $g+1$ without leaving its current group g. 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.

	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 — Sender:

```
// 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 $c_{g,p} \leftarrow C_g$.
```

Algorithm — Receiver:

```
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 $n \leftarrow g$.
```

	Sender		Receiver	
Output	$c_{g,p}$	component field for packet p in S_g where $1 \leq g \leq N$	n	next group
	d_g	decrease field for group g where $2 \leq g \leq N$	u_j	updated key for group 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 $2 \leq g \leq N$		

Fig. 5. DELTA instantiation for replicated multicast protocols.

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 Fig. 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

$$\tau_g = \bigoplus_{p \in S_g} c_{g,p} \text{ and } \sigma_g = \bigoplus_{p \in S_{g-1}} c_{g-1,p}, \tag{4}$$

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 [27] to distribute components of key τ_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 $k - 1$:

$$q(x) = \tau_g + a_1 x + \ldots + a_{k-1} x^{k-1} \tag{5}$$

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

$$c_p = (p, q(p)) \tag{6}$$

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

$$\tau_g = q(0). \tag{7}$$

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 might incur high communication overhead. Design of threshold schemes that reuse components is an interesting topic for future research.

Reliability. Reliable multicast protocols overcome losses by transmitting additional packets, e.g., packets with retransmitted data or error correction codes [6]. 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 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. The alteration prevents receivers ineligible for a group from reconstructing the group key. Furthermore, it is possible to instantiate DELTA without any changes in routers, e.g., an instantiation can use ECN marks as one-bit components of group keys.

II) 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 II-B, 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.

A) 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 III-A-I 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.

B) 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 Fig. 2). To allow a receiver to maintain 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 forward 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 Fig. 6(a)]. Whereas some multicast protocols always permit unconditional access to the minimal group, the others prescribe that a congested receiver of the minimal group should leave the session. To support both types of protocols, SIGMA enables the sender to instruct edge routers about an allowed duration of unconditional uninterrupted access to the minimal group. If a new receiver does not start submitting valid keys for the minimal group by the end of this interval, the edge router blocks the interface from receiving the group for a duration that the sender specifies also via SIGMA.

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. Fig. 6(b) shows the general format of subscription messages. In particular, if the protected protocol allows a receiver to add multiple groups during one time slot, the receiver can use a single subscription message to submit address-key pairs for all the current and added groups. 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.

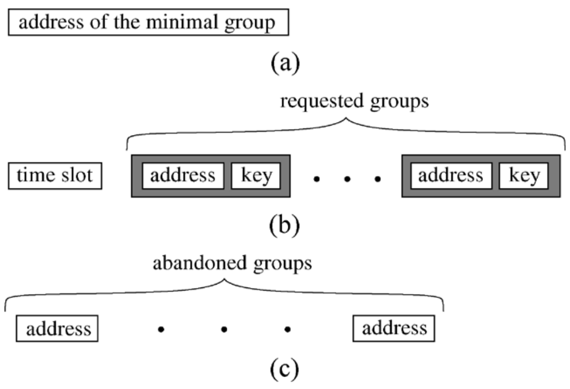


Fig. 6. SIGMA messages sent by receivers: (a) session-join message, (b) subscription message, and (c) unsubscription message.

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 Fig. 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.

C) Deployment of SIGMA: It is possible to deploy SIGMA incrementally. 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, a partial deployment of SIGMA edge routers is beneficial—these routers prevent their local receivers from inflated subscription.

III) Properties of DELTA and SIGMA: We first argue that DELTA and SIGMA meet the design requirements from Section II-B.

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 or additional servers.

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 IV, we experimentally verify that DELTA and SIGMA also preserve other congestion control properties of protected protocols.

We now 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], [30] and b) hop-by-hop or edge-to-edge encryption to protect against snooping on network links [22].

Protection against attacks on SIGMA. As long as a local interface provides an edge router with a valid key for a group, the router forward 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 III-A-I, keys and components have the same size, and the component guessing gives no advantage over guessing the key directly.

B. Protection Against Collusion Attacks

The presented 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 when receivers collude, e.g., when more capable receivers pass keys or their components to less capable receivers. Extending the design to achieve robustness against the collusion attacks requires guarding a group with a different set of keys at each local interface. Edge routers can support interface-specific keys by transforming DELTA fields in forwarded packets as follows:

$$f_{i,p} = G(m_i, f_p) \tag{8}$$

where G is a one-way function, m_i is a modifier at local interface i, and f_p and $f_{i,p}$ are the contents of a DELTA field in packet p before and after the edge router forward the packet to interface i. If delivered interface-specific DELTA fields are insufficient for reconstructing a valid interface-specific key for a group at a receiver, receivers from other interfaces cannot help the congested receiver to access the group because their keys (as well as components of their keys) are invalid at the interface of the receiver. Furthermore, the one-way property of G prevents receivers from determining original field contents f_p or modifiers m_i.

Implementing the above general approach faces a number of challenges. An acceptable implementation should definitely avoid introducing security flaws. For example, edge routers should pick modifiers m_i randomly to negate dictionary attacks trying to uncover predictable modifiers [21]. A more serious obstacle lies in our objective to satisfy all the requirements from Section II-B. An implementation of the interface-specific group access should either a) construct interface-specific keys at the edge routers from generic information provided by the sender or b) disseminate interface-specific keys to the edge routers after constructing the keys at the sender. As we discuss below, these alternatives represent a trade-off between generality of the required network support and scalability of the design.

If the edge routers construct interface-specific keys from generic information provided by the sender, the edge routers have to know the type of the protected protocol along such dimensions as the session structure, congestion notification, and congested state definition. Furthermore, operation of the edge routers becomes protocol-dependent. Unlike the original design where the edge routers run the same simple code to intercept SIGMA packets and extract keys, construction of interface-specific keys at the edge routers is more complicated and involves executing different algorithms for different types of protected protocols. Thus, this extension provides robustness against the collusion attacks by weakening generality of the needed network support.

To preserve generality of the network support, an alternative extension can disseminate interface-specific keys to the edge routers after constructing the keys at the sender. In such extension, the sender has to know the interface-specific modifiers used at the edge routers. Whereas updates of the modifiers can be relatively infrequent without undermining security of the design, the sender must provide the edge routers with new sets of the interface-specific keys for each time slot. This overhead limits scalability of the extension.

Hence, extending DELTA and SIGMA to achieve robustness against the collusion attacks is subject to a fundamental trade-off between degrading scalability of the design and diluting generality of the required network support. Finding an optimal balance between scalability and generality remains an open research problem.

IV. Derivation of Robust Protocols

In Section III, we described DELTA and SIGMA that allow multicast protocols to acquire immunity against inflated subscription. Below, we derive and evaluate robust versions of prominent RLM and FLID-DL protocols.

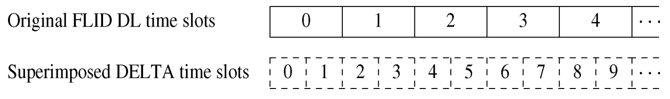


Fig. 7. Integration of FLID-DL with DELTA.

A. Robust Adaption of FLID-DL

FLID-DL is an unreliable layered multicast protocol that defines congestion as a single packet loss [4]. To protect FLID-DL against inflated subscription, we derive its robust version FLID-DS (Fair Layered Increase/Decrease with DELTA and SIGMA) by integrating FLID-DL with the first DELTA instantiation presented in Section III-A-I.

Both FLID-DL and DELTA employ time slots, albeit for different purposes. A time slot in FLID-DL determines how quickly a congested receiver is supposed to unsubscribe from its top group. On the other hand, a time slot in DELTA specifies for how long a group key remains valid. Because SIGMA ensures that a receiver drops its top group within two DELTA time slots after congestion occurs, we set the DELTA time slot duration in FLID-DS to the half of the FLID-DL time slot duration and superimpose two DELTA time slots on each FLID-DL time slot as shown in Fig. 7. Thereby, FLID-DS enforces the intended responsiveness of FLID-DL congestion control.

Even a well-behaving FLID-DS receiver can behave differently than a well-behaving FLID-DL receiver under identical circumstances. For example, if a group loses all its packets, a FLID-DS receiver might need to reduce the subscription by more than one group. Furthermore, our implementation of FLID-DS enables a receiver to perform those bandwidth-hogging actions that a misbehaving receiver could have done anyway in spite of DELTA and SIGMA: for example, the FLID-DS receiver can preserve its top group if this group is its only congested group, and the sender authorizes an upgrade to this group; also, if congestion happens only during the second half of a FLID-DL time slot, the FLID-DS receiver can postpone leaving its top group until the middle of the next FLID-DL time slot (whereas a well-behaving FLID-DL receiver would leave the group at the end of the current FLID-DL time slot). In our evaluation of FLID-DS, we examine how these behavioral differences affect the long-term performance of the protocol.

To evaluate FLID-DS, we conduct experiments using a single-bottleneck dumbbell topology in NS-2 [23]. 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 bottleneck link. The fair bandwidth share for each session is 250 Kb/s. 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 Mb/s. 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 Kb/s. The cumulative transmission rate of the session grows multiplicatively with a factor of 1.5 per group. We set the FLID-DL time slot duration to its default value of 500 ms [4] and therefore use 250 ms as the DELTA time slot duration in FLID-DS. All data packets are 576 bytes long (576 bytes is a common packet size in the Internet).

I) Preventing Inflated Subscription: First, consider a setting where receivers `F1` and `F2` from different FLID-DL sessions share the 1-Mb/s bottleneck link with two TCP Reno [1] receivers `T1` and `T2`. After 100 seconds into the simulation, receiver `F1` starts to misbehave and inflates its subscription in violation of the protocol. Such a misbehavior boosts the throughput of `F1` to 690 Kb/s at the expense of well-behaving receivers `F2`, `T1`, and `T2` [see Fig. 8(a)]. We repeat this experiment when the multicast sessions use FLID-DS instead of FLID-DL. Although `F1` tries to inflate its subscription after 100 s, DELTA and SIGMA preserve—as Fig. 8(b) shows—the fair bandwidth allocation.

II) Preserving Congestion Control Properties: We now 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. 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 an interval of 200 s.

First, we examine the multicast sessions without cross traffic. For FLID-DL and FLID-DS, respectively, Fig. 9(a) and (b) report individual throughputs and their average over the number of sessions. These and subsequent graphs show that integration with DELTA and SIGMA leads to small but often not negligible decrease in the average throughput of the protected protocol. On the other hand, DELTA and SIGMA improve intra-protocol fairness. We attribute the different performance of FLID-DL and FLID-DS to their slight behavioral differences discussed above.

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 s. Fig. 9(c) shows that whereas the bandwidth allocation for multicast traffic depends on the number of sessions, FLID-DL and FLID-DS receivers have similar average throughputs.

Responsiveness. To investigate the impact of DELTA and SIGMA on responsiveness of congestion control, we consider a setting where only a multicast (FLID-DL or FLID-DS) session and an on-off CBR session share the bottleneck link. The CBR session transmits its data at a rate of 800 Kb/s during the time interval between 45 and 75 s. Fig. 9(d) shows that FLID-DS preserves responsiveness of the original FLID-DL protocol.

Heterogeneous round-trip times. In FLID-DL, the average throughput of a receiver is relatively independent from the round-trip time of the receiver. To verify preservation of this property, we conduct experiments where 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. The measured average throughputs of the FLID-DS receivers

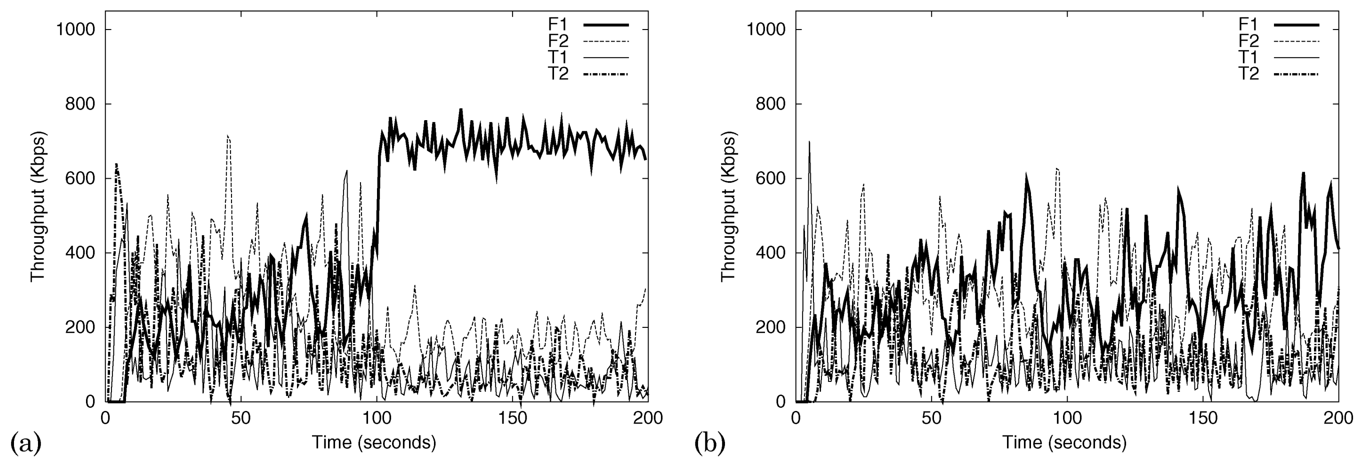


Fig. 8. Protection of FLID-DL with DELTA and SIGMA: (a) vulnerability of FLID-DL to inflated subscription and (b) robustness of FLID-DS against inflated subscription.

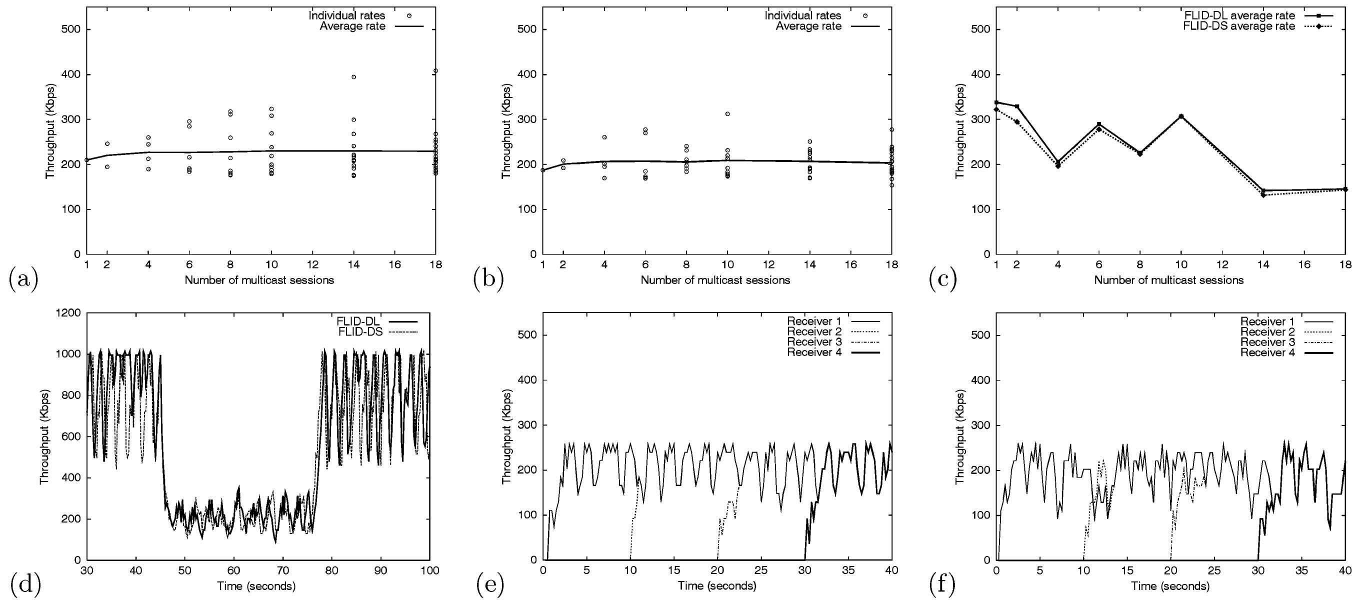


Fig. 9. Preservation of FLID-DL congestion control properties: (a) throughput for FLID-DL without cross traffic, (b) throughput for FLID-DS without cross traffic, (c) average throughput with cross traffic, (d) responsiveness, (e) subscription convergence in FLID-DL, and (f) subscription convergence in FLID-DS.

are almost identical and remain close to the average throughputs 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 four receivers. The receivers join the session at times 0, 10, 20, and 30 s, respectively. As Fig. 9(e) and (f) indicate, the receivers converge to the same fair subscription both in FLID-DL and FLID-DS.

III) Communication Overhead: Consider a FLID-DL session that has N groups, increases the cumulative transmission rate with a factor of m per group, and sends s bits of data in each packet. DELTA adds a b-bit *component* field to each packet and b-bit *decrease* field to every packet of groups 2 through N. Since only $1/m^{N-1}$ of all packets in the session belong to the minimal group, DELTA imposes the following communication overhead:

$$O_\Delta = \left(2 - \frac{1}{m^{N-1}}\right)\frac{b}{s}. \tag{9}$$

For each DELTA time slot, SIGMA sends special packets with an l-bit slot number and one address-key tuple per group. Every tuple contains a 32-bit group address and b-bit *top* key. Besides, tuples for groups 1 through $N-1$ include a b-bit *decrease* key. When the protocol authorizes—with an average frequency of f_g per time slot—an upgrade to group g, the tuple for group g also contains a b-bit *increase* key. Forward error correction boosts the amount of added bits by a factor of z. The headers of the special packets consume a total of

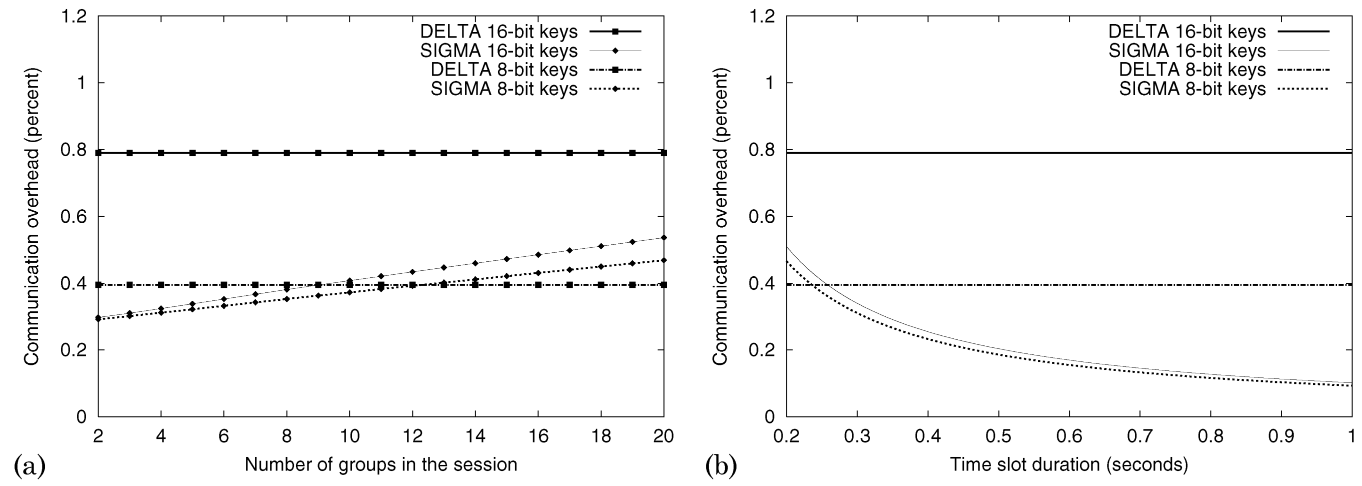


Fig. 10. DELTA and SIGMA communication overhead in FLID-DS: (a) dependence on the number of groups in the session and (b) dependence on the time slot duration.

h bits. Then, the communication overhead for SIGMA is as follows:

$$O_\Sigma = \frac{\left(l + 32N + b\left(2N - 1 + \sum_{g=2}^{N} f_g\right)\right) z + h}{R \cdot t} \tag{10}$$

where R is the cumulative transmission rate of the session, and t is the DELTA time slot duration.

We quantify O_Δ and O_Σ for a session that transmits packets with 500 bytes of data (i.e., $s = 4000$) at cumulative rate R of 4 Mb/s. The minimal group transmits its data at rate r of 100 Kb/s. A slot number consists of 8 bits. We consider two settings where keys are 8 and 16 bits long, respectively. Error correction overcomes 50% packet loss. We set m to $(R/r)^{1/(N-1)}$ and determine values of f_g, z, and h experimentally.

First, we explore the dependence of the overhead on the number of groups. Fig. 10(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. Fig. 10(b) plots O_Δ and O_Σ for t varying from 200 ms to 1 s when $N = 10$. In both cases, the communication overhead remains about 0.8% for DELTA and stays under 0.6% for SIGMA. Increasing the key size from 8 to 16 bits boosts the SIGMA overhead only modestly because most of this overhead is due to SIGMA packet headers. In general, DELTA and SIGMA protect FLID-DL against inflated subscription without imposing a significant overhead.

B. Robust Adaption of RLM

RLM is another unreliable multicast protocol for disseminating layers of hierarchically encoded data [20]. However, RLM considers a receiver to be congested only if the loss rate exceeds a threshold. The default threshold equals 25% of the packets transmitted to the current subscription level.

Unlike with FLID-DL, inflated subscription is not the only type of receiver misbehavior hurting RLM. Receivers in RLM synchronize their subscriptions via a process of *shared learning* [20]. Abusing this process, a misbehaving receiver can keep other receivers below their fair subscription levels [12]. To derive a robust version of RLM, we first replace shared learning with a synchronization mechanism employed in FLID-DL: the sender multicasts explicit signals authorizing uncongested receivers to increase their subscriptions at the end of a time slot. We refer to the adjusted design as RLM-F (RLM with FLID-like subscription synchronization). We choose the RLM-F time slot duration to preserve the average responsiveness of RLM congestion control.

Although immune against abuse of other receivers, RLM-F remains vulnerable to inflated subscription. To complete deriving the robust version of RLM, we integrate RLM-F with the above DELTA instantiation based on Shamir's scheme. We refer to the result as RLM-DS (RLM with DELTA and SIGMA). To enforce the intended responsiveness of RLM-F congestion control, we set the DELTA time slot duration in RLM-DS to the half of the RLM-F time slot duration and superimpose two DELTA time slots on each RLM-F time slot. Since Shamir's scheme does not reuse components, the below evaluation of RLM-DS includes a careful analysis of its DELTA communication overhead.

We evaluate RLM-F and RLM-DS in the same network topology as in Section IV-A. Each multicast session contains six groups. The minimal group transmits data at a rate of 100 Kb/s. Each higher group doubles the cumulative transmission rate of the session. We set the RLM-F time slot duration to 10 s. Hence, the DELTA time slot duration in RLM-DS equals 5 s. The minimal group carries an increase signal every time slot. Each higher group halves the frequency of increase signals. The loss rate threshold for each subscription level is 25% of the packets transmitted to this level during each time slot.

1) Preventing Inflated Subscription: We first consider a setting where four receivers `R1`, `R2`, `R3`, and `R4` from different RLM-F sessions share the 1.1-Mb/s bottleneck link. Up to 100 seconds into the experiment, all the receivers adhere to the protocol and converge toward fair and efficient sharing of the bottleneck bandwidth. After 100 s, receiver `R1` misbehaves by inflating its subscription to four groups. With RLM-F, such the

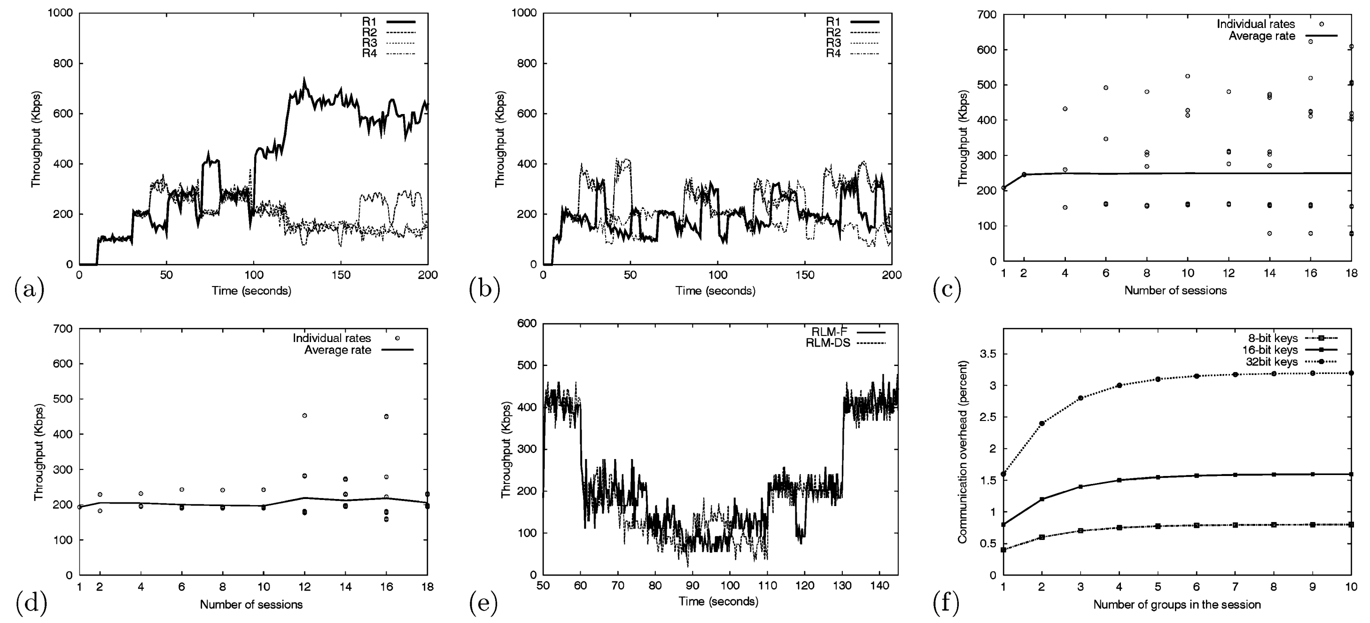


Fig. 11. Protection of RLM with DELTA and SIGMA: (a) vulnerability of RLM-F to inflated subscription, (b) immunity of RLM-DS to inflated subscription, (c) throughput with RLM-F, (d) throughput with RLM-DS, (e) responsiveness, and (f) DELTA communication overhead in RLM-DS.

misbehavior rewards R1 with an unfairly high throughput at the expense of well-behaving receivers R2, R3, and R4 [see Fig. 11(a)]. We repeat this experiment when the multicast sessions use RLM-DS. As Fig. 11(b) shows, DELTA and SIGMA protect the fairness of the bandwidth allocation against the attack.

II) Preserving Congestion Control Properties: To assess the impact of DELTA and SIGMA on throughput, we conduct experiments where the number of multicast (RLM-F or RLM-DS) sessions varies from 1 to 18. There is no other traffic, and the fair bandwidth share for each receiver equals 250 Kb/s. For the only receiver in each session, we measure throughput over a period of 200 s. For RLM-F and RLM-DS, respectively, Fig. 11(c) and (d) report individual throughputs and their average over the number of sessions. The graphs show that while reducing somewhat the average throughput, DELTA and SIGMA decrease dramatically the deviation of the individual throughputs and thereby improve intra-protocol fairness.

We also experiment in a setting where a multicast (RLM-F or RLM-DS) session shares a 500 Kb/s bottleneck link with an on-off CBR session that transmits at a rate of 460 Kb/s during the interval between 60 and 110 s. Fig. 11(e) shows that RLM-DS preserves the responsiveness of RLM-F congestion control.

III) Communication Overhead: Due to the same usage of SIGMA in FLID-DS and RLM-DS, the latter has the same insignificant SIGMA communication overhead as derived in Section IV-A. However, the DELTA overhead is different because keys in Shamir's scheme do not share components. Instead, each packet of group g communicates a separate component (x, y_j) for every group j from g to N by allocating b bits for x as well as for each of the $N-g+1$ integers y_j. DELTA also inserts a b-bit *decrease* field into every packet of groups 2 through N. If p_j refers to the number of data packets transmitted to group j during a time slot, then the DELTA communication overhead in RLM-DS becomes

$$O_R = \frac{\left(p_1(N+1) + \sum\limits_{j=2}^{N}(p_j(N-j+3))\right)b}{\left(\sum\limits_{j=1}^{N} p_j\right)s} \tag{11}$$

and can be expressed as

$$O_R = \frac{3m^{N-1} - 2m^{N-2} - 1}{m^{N-2}(m-1)} \cdot \frac{b}{s}. \tag{12}$$

It is interesting that the derived expression has an upper bound independent from the number of groups:

$$O_R < \left(3 + \frac{1}{m-1}\right)\frac{b}{s}. \tag{13}$$

For example, if each higher group doubles the cumulative transmission rate of the session (i.e., $m=2$), then DELTA adds in average less than four b-bit fields per packet. The reason for the constant upper bound is the multiplicative allotment of group rates: higher groups transmit most of the session packets but insert only few components per packet.

We quantify O_R for a session with six groups $(N = 6)$ where each higher group doubles the cumulative transmission rate $(m = 2)$, each packet carries 500 bytes of data $(s = 4000)$, keys consist of 16 bits $(b = 16)$. In this setting, DELTA adds in average 3.94 fields per packet and imposes 1.6% communication overhead. For keys of size 8, 16, and 32 bits, Fig. 11(f) shows that O_R remains small and almost constant when the number of groups exceeds five. Hence, despite the lack of component reuse in Shamir's scheme, DELTA and SIGMA protect RLM-DS against inflated subscription without imposing significant overhead.

V. 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 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 robust adaptations of FLID-DL and RLM protocols.

Whereas this paper focused on IP multicast, robustness of congestion control protocols is important for *end-system multicast* as well. End-system multicast designs can be classified as *server-based* and *peer-to-peer*. In server-based multicast, trusted servers form forwarding hierarchies. Server-based multicast designs can achieve robustness against inflated subscription by adopting DELTA and SIGMA straightforwardly: as edge routers in IP multicast, edge servers can enforce congestion-dependent access of local receivers to multicast groups.

On the other hand, peer-to-peer multicast designs [3] construct forwarding hierarchies from receivers and thereby open new opportunities for receiver misbehavior:

- A misbehaving receiver that lies on the path from the sender to a receiver can forward 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 intermediary to improve its own reception when downstream and upstream links share a physical medium (e.g., when the misbehaving intermediary is connected to the network by a wireless link).
- 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.

Since our solution assumes trusted intermediaries, DELTA and SIGMA do not neutralize the above attacks. Furthermore, detection of a misbehaving intermediary is a difficult task. In particular, a receiver has no easy ways to determine whether its current path from the sender delivers data at an unfairly low rate. Design of robust protocols for peer-to-peer multicast is a topic for future research.

References

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

[2] A. Ballardie and J. Crowcroft, "Multicast-specific security threats and counter-measures," in *Proc. NDSS 1995*, Feb. 1995.

[3] S. Banerjee, B. Bhattacharjee, and C. Kommareddy, "Scalable application layer multicast," in *Proc. ACM SIGCOMM*, Aug. 2002, pp. 205–217.

[4] J. W. Byers, G. Horn, M. Luby, M. Mitzenmacher, and W. Shaver, "FLID-DL: congestion control for layered multicast," *IEEE J. Sel. Areas Commun.*, vol. 20, no. 8, pp. 1558–1570, Oct. 2002.

[5] J. W. Byers, G. Kwon, M. Luby, and M. Mitzenmacher, "Fine-grained layered multicast with STAIR," *IEEE/ACM Trans. Netw.*, vol. 14, no. 1, pp. 81–93, Feb. 2006.

[6] J. W. Byers, M. Luby, and M. Mitzenmacher, "A digital fountain approach to asynchronous reliable multicast," *IEEE J. Sel. Areas Commun.*, vol. 20, no. 8, pp. 1528–1540, Oct. 2002.

[7] S. Y. Cheung and M. H. Ammar, "Using destination set grouping to improve the performance of window-controlled multipoint connections," *Comput. Commun. J.*, vol. 19, pp. 723–736, 1996.

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

[9] S. E. Deering, "Multicast routing in a datagram Internetwork," Ph.D. dissertation, Stanford Univ., Stanford, CA, Dec. 1991.

[10] D. Ely, N. Spring, D. Wetherall, S. Savage, and T. Anderson, "Robust congestion signaling," in *Proc. IEEE ICNP*, Nov. 2001, pp. 332–341.

[11] W. Fenner, Internet Group Management Protocol, Version 2 , RFC 2236, Nov. 1997.

[12] S. Gorinsky, S. Jain, and H. Vin, "Multicast congestion control with distrusted receivers," in *Proc. Networked Group Communication (NGC)*, Oct. 2002, pp. 19–26.

[13] S. Gorinsky, S. Jain, H. Vin, and Y. Zhang, "Robustness to inflated subscription in multicast congestion control," in *Proc. ACM SIGCOMM*, Aug. 2003, pp. 87–98.

[14] S. Gorinsky, K. K. Ramakrishnan, and H. Vin, Addressing heterogeneity and scalability in layered multicast congestion control Dept. Comput. Sci., Univ. Texas at Austin, Tech. Rep. TR2000-31, Nov. 2000.

[15] H. W. Holbrook and D. R. Cheriton, "IP multicast channels: EXPRESS support for large-scale single-source applications," in *Proc. ACM SIGCOMM*, Sep. 1999, pp. 65–78.

[16] P. Judge and M. Ammar, "Gothic: a group access control architecture for secure multicast and anycast," in *Proc. IEEE INFOCOM*, Jun. 2002, pp. 1547–1556.

[17] A. Legout and E. W. Biersack, "PLM: fast convergence for cumulative layered multicast transmission schemes," in *Proc. ACM SIGMETRICS*, Jun. 2000, pp. 13–22.

[18] M. Luby, V. K. Goyal, S. Skaria, and G. B. Horn, "Wave and equation based rate control using multicast round trip time," in *Proc. ACM SIGCOMM*, Aug. 2002, pp. 191–214.

[19] R. Mahajan, S. Floyd, and D. Wetherall, "Controlling high-bandwidth flows at the congested router," in *Proc. IEEE ICNP*, Nov. 2001, pp. 192–201.

[20] S. McCanne, V. Jacobson, and M. Vetterli, "Receiver-driven layered multicast," in *Proc. ACM SIGCOMM*, Aug. 1996, pp. 117–130.

[21] A. J. Menezes, P. C. Van Oorschot, and S. A. Vanstone, *Handbook of Applied Cryptography*. Boca Raton, FL: CRC Press, 2001.

[22] S. Mittra, "Iolus: a framework for scalable secure multicasting," in *Proc. ACM SIGCOMM*, Sep. 1997, pp. 277–288.

[23] UCB/LBNL/VINT Network Simulator ns-2. May 2004 [Online]. Available: http://www-mash.cs.berkeley.edu/ns

[24] A. Perrig, R. Canetti, D. Song, and D. Tygar, "Efficient and secure source authentication for multicast," in *Proc. Network and Distributed System Security Symp. (NDSS)*, Feb. 2001, pp. 35–46.

[25] K. K. Ramakrishnan and S. Floyd, A proposal to add Explicit Congestion Notification (ECN) to IP , RFC 2481, Jan. 1999.

[26] S. Savage, N. Cardwell, D. Wetherall, and T. Anderson, "TCP congestion control with a misbehaving receiver," *ACM Comput. Commun. Rev.*, vol. 29, no. 5, pp. 71–78, Oct. 1999.

[27] A. Shamir, "How to share a secret," *Commun. ACM*, vol. 22, no. 11, pp. 612–613, Nov. 1979.

[28] D. Sisalem and A. Wolisz, "MLDA: a TCP-friendly congestion control framework for heterogeneous multicast environments," in *Proc. IWQoS*, Jun. 2000, pp. 65–74.

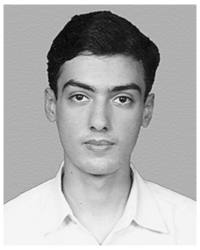
[29] L. Vicisano, L. Rizzo, and J. Crowcroft, "TCP-like congestion control for layered multicast data transfer," in *Proc. IEEE INFOCOM*, Mar. 1998, pp. 996–1003.

[30] M. Waldvogel, G. Caronni, D. Sun, N. Weiler, and B. Plattner, "The versakey framework: versatile group key management," *IEEE J. Sel. Areas Commun.*, vol. 17, no. 9, pp. 1614–1631, Sep. 1999.

Sergey Gorinsky (M'05) is a native of Skhodnya, Russia. He received the Engineer degree from Moscow Institute of Electronic Technology, Zelenograd, Russia, and the M.S. and Ph.D. degrees from the University of Texas at Austin.

He is currently with the Applied Research Laboratory, Department of Computer Science and Engineering, Washington University in St. Louis, where he works as an Assistant Professor. His primary research interests are in computer networking and distributed systems.

Dr. Gorinsky has been a member of the ACM since 2004.



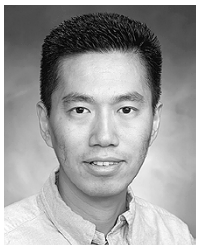
Sugat Jain received the B.Tech. degree in computer science and engineering from the Indian Institute of Technology, Delhi, and the M.S. degree in computer science from the University of Texas at Austin, where he is currently pursuing the Ph.D. degree in the Department of Computer Sciences.

Prior to joining UT Austin, he worked for a year at AT&T Research Labs, Florham Park, NJ.

Harrick Vin (M'99) received the Ph.D. degree from the University of California at San Diego.

He is a Professor in the Department of Computer Sciences at the University of Texas at Austin, founding Director of the Distributed Multimedia Computing Laboratory, and co-Director of the Laboratory for Advanced Systems Research at the University of Texas at Austin. His research interests are in the areas of networks, operating systems, distributed systems, and multimedia systems.

Dr. Vin has been a member of the ACM since 1991.



Yongguang Zhang (M'94) received the Ph.D. degree from Purdue University, West Lafayette, IN.

He is a Senior Research Scientist at HRL Laboratories, LLC, Malibu, CA, where he does research in mobile wireless networks and systems. He edited a book on internetworking and computing over satellite networks and taught computer science courses at the University of Texas at Austin.