# Enabling Confidentiality of Data Delivery in an Overlay Broadcasting System

Ruben Torres, Xin Sun, AAron Walters, Cristina Nita-Rotaru and Sanjay Rao
Purdue University
{*rtorresg,sun19,sanjay*}*@ecn.purdue.edu, crisn@cs.purdue.edu, awalters@4tphi.net*

***Abstract*— In this paper, we present an extensive study of key dissemination schemes in an overlay multicast context, and the first to involve actual implementation, real traces, and performance in Internet environments. Given that rekey traffic has stronger resilience requirements and is burstier than data traffic, we consider whether data and keys must be distributed using the same overlay or using two separate dissemination structures. Our key findings are: (i) A coupled architecture is effective in achieving resilient key dissemination. Using TCP in each hop of the dissemination structure (an opportunity unique to overlays) is effective in achieving resiliency in end-to-end key delivery. The performance can be further enhanced if convergence properties of overlays are considered; and (ii) A coupled architecture optimized for data delivery has high overheads, while a coupled architecture optimized for key delivery may not honor access bandwidth constraints of nodes. Distributing data and keys using separate overlays achieves low overhead for key dissemination while honoring access bandwidth constraints of nodes.**

**Keywords:** Computer Networks, Computer Network Security, Broadcasting

## I. Introduction

Research in peer-to-peer video streaming [1], [2], [3], [4], [5], [6], [7], [8], [9], [10], [11], [12], [13], [14] has matured to an extent that there are real deployments [15], [16], [17], and more recently, commercial activity [18], [19]. Achieving further usage of these systems in a wider range of applications requires integrating security mechanisms to enable confidentiality and integrity of data delivery. Confidentiality ensures that only authorized nodes have access to the data while integrity ensures that no adversary modified the data between the source and receivers. These security mechanisms can be efficiently provided if all participants in the broadcast share a secret key, referred to as the *group key*. The source restricts the access to the data to only authorized members by encrypting the data with the secret key and changing the key when nodes join or leave the broadcast overlay. The source performs additional encryption operations to ensure that the new key is delivered in a secure way to all members. Protocols used to change and deliver the group key are known as *group key management protocols*.

A vast majority of previous work in group key management for broadcast systems has been conducted in the context of IP Multicast, focusing on improving key delivery [20], [21] and reducing the encryption overhead at the source [20], [21], [22], [23], [24], [25], [26], [27]. Overlay networks pose new challenges and opportunities for group key management. In overlays, there is no native multicast medium (e.g. IP Multicast) that could be used for key distribution. However, keys could be distributed by using the existing overlay data delivery structure or by constructing additional structures specifically designed for key distribution. Moreover, overlays offer new opportunities for designing resilient key dissemination mechanisms.

Recently, researchers have begun to consider issues in key dissemination using overlays [28], [29]. While these works are important first steps, they are "design-centric", and rely on analysis or simulations with synthetic workloads. In contrast, in this paper, we adopt an "evaluation-centric" approach. Our primary focus is on systematically exploring the design space, and obtaining deeper insights into the performance of various design alternatives under realistic deployment scenarios. To this end, we conduct an extensive study of key dissemination schemes in the context of a widely deployed operational broadcasting system [15], using experiments on the Planetlab testbed, and real traces of join/leave dynamics. Evaluations in Internet settings enable us to provide unique insights into the interaction between key management and data traffic, and to study performance of the schemes under realistic loss scenarios. To our knowledge, ours is the first study to involve actual implementation, real traces, and performance in Internet environments.

A primary focus of our evaluations is to explore the merits of decoupling data and key delivery using two separate dissemination structures. Our study is motivated by the fact that rekey traffic differs from data traffic given it is less tolerant to loss, and is more bursty in nature. Reducing the burstiness of rekey traffic is important to minimize the impact of addition of security services on system scalability. Decoupling data and key delivery is unique to the overlay context, and has not been explored before. Our key findings are:

- From the perspective of achieving resilient key dissemination, a coupled architecture suffices, and the benefits of building separate and more resilient structures for key dissemination is marginal. Using reliable protocols (TCP) in each hop of the key dissemination structure is effective in achieving resiliency in end-to-end delivery. The performance can be further enhanced if convergence properties of overlays are considered. While reliable key dissemination has proven challenging in the context of IP Multicast, these results show

Manuscript received March 15, 2007; revised August 7, 2007. This work was supported in part by the National Science Foundation Cybertrust Awards No. 0715833 and No. 0430271. The views expressed in this research are not endorsed by the National Science Foundation.

that it can be considerably simplified with overlays.

• Using separate structures for disseminating data and keys can significantly help reduce peak overheads due to burstiness of rekey traffic, while still achieving good application performance. Our evaluations are inspired by [28], which observed that explicitly optimizing overlays for key distribution is effective in minimizing overheads. However, we show that it is not feasible to use such an overlay for distributing data in bandwidth-demanding video streaming applications, given the need to honor physical access bandwidth constraints of nodes. We show that if key and data dissemination are decoupled, the benefits of reducing the overhead associated with key dissemination still hold, and outweigh the cost of maintaining an additional overlay structure.

Given that overlay broadcasting systems have typically targeted tens of thousands of recipients, it is desirable to minimize the effect of addition of security services on system scalability. As a step towards the goal, and a final contribution of the paper, we show that peak overheads with rekeying may be further reduced by combining knowledge of the duration the nodes stay in the group with the key management algorithms.

The rest of the paper is organized as follows. Section II describes the system settings and assumptions considered in this work. Section III describes key management algorithms. Section IV discusses the design space for dissemination of data and keys. Section V presents our evaluation methodology. Sections VI and VII present our experimental results. Section VIII concludes the paper.

## II. System and Adversary Model

We focus on peer-to-peer networks providing support for single-source high-bandwidth broadcasting applications. The source is assumed to be continually available. This work is conducted in the context of systems that construct trees for data delivery [15], [4], [6], [8], [2], [3], [10], [13]. We discuss implications for multi-tree based [11], [7], and mesh-based approaches [9], [30], [17], [31] in Section VIII. In tree-based approaches, each node receives data from its *parent*, and forwards data to its *children*, with the source being the root of the tree. Each node also maintains a *saturation degree*, the maximum number of concurrent data streams a node is able to support before saturating its physical outgoing access link. The saturation degree may vary across nodes depending on their access bandwidth (e.g. DSL, Ethernet), and it is critical the node bandwidth constraints are honored. Nodes also maintain an additional set of members via a group management protocol, and continually adapt to scenarios such as parent failures.

Our focus is ensuring that only authorized group members have access to *group data* generated by the application and broadcasted by the source. Overlay networks also disseminate control messages generated by the group management and the overlay adaptation protocols. We assume that mechanisms to protect the control traffic are in place. Unless otherwise specified, we consider only outside adversaries who attempt to obtain unauthorized access to the group data. Any node that is no longer part of the group is considered an outsider. The source of the broadcast is trusted to behave correctly and group members do not forward the secret group key or decrypted data to participants who are not part of the group. We assume that there exist mechanisms allowing the source to authenticate a host joining the broadcast group and establish a pair-wise key with it.

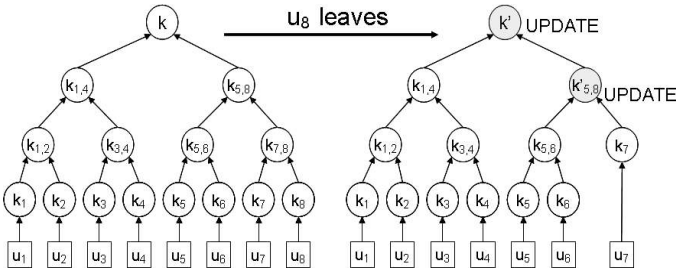



Fig. 1. LKH: Example of updating the tree when user $u_8$ leaves the group.

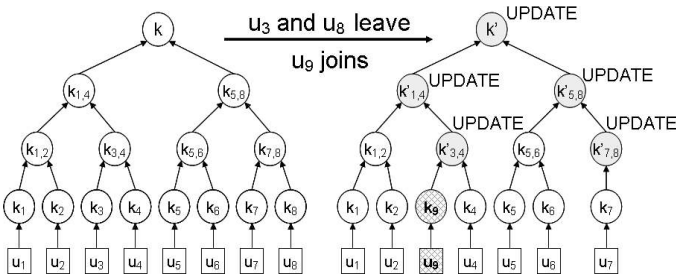



Fig. 2. Marking: Example of updating the tree when the number of joins is smaller than the number of leaves.

## III. Key Management Algorithms

Given our focus on single source broadcasting applications, we consider centralized key management schemes. Such schemes rely on a single entity, referred to as *key server*, to select and distribute the group key. As long as a member has the current group key, it can decrypt and thus have access to the broadcasted data. The access to data is restricted by changing the group key. Any members that are not part of the group yet, or have left the group, are not able to decrypt the data. These properties are known as *forward and backward secrecy*. While the extensive use of centralized key management schemes makes them a natural starting point for our studies, it may be feasible to achieve better scalability by adopting techniques from more hierarchical approaches such as [26] where separate entities are delegated to control subgroups of users. We defer this to future work.

Many centralized key management schemes focus on decreasing the load on the key server resulting from encryptions required when distributing the key. The schemes we consider are:

***Key-Star*:** Introduced in [23], this scheme specifies that the source encrypts the new key with each node's pair-wise key when performing a rekey operation. *Key-Star* requires O(N) encryptions at the source, as well as O(N) messages, where $N$ is the group size.

**LKH:** The computation cost of *Key-Star* was significantly reduced by the LKH [32] scheme. LKH improves over *Key-Star* by using subgroup keys to encrypt the new group key.

TABLE I
NOTATION

| Purpose | Scheme |
|---|---|
| Key Management | ***Key-Star***: Source encrypts new group key with each other pair-wise key.<br>**LKH**: Source encrypts new group key with subgroup keys. Keys are organized in a key tree.<br>***Marking***: LKH plus batch rekeying |
| Resiliency | ***Coupled-DataOpt***: Data and keys disseminated on overlay optimized for data delivery. This includes:<br>• *Tree-UDP*: Keys sent with UDP on each hop of the overlay.<br>• *Tree-TCP*: Keys sent with TCP on each hop of the overlay.<br>• *Tree-Unicast*: Similar to *Tree-TCP* plus overlay convergence considerations.<br>***Gossip***: Keys disseminated using gossip-like mechanisms |
| Overhead Reduction through Dissemination Structure | ***Coupled-KeyOpt***: Data and keys disseminated on overlay optimized for keys delivery.<br>***Decoupled***: Data and keys disseminated over different specialized overlays. |
| Overhead Reduction through Lowering Encryptions | ***StayAware***, ***StayAware-Prefetch***: Consider the time nodes stay in the group, to reduce number of encryptions at rekey event. |

The subgroup keys are not known by the members that left, so the approach has similar security properties as *Key-Star*. The keys are organized in a *key tree* where the root corresponds to the group key, the intermediate keys to subgroup keys and the leaves to the pair-wise keys between the source and each member. LKH achieves logarithmic broadcast size and computational cost with the group size. Figure 1 shows a LKH example, where user $u_8$ leaves. The keys that must be changed because they were known by user $u_8$ are the group key $k$ and the subgroup key $k_{5,8}$. The new group key $k'$ must be received by all members while the new subgroup key $k'_{5,8}$ must be received by users $u_5$, $u_6$ and $u_7$. The subgroup key $k_{1,4}$ is used to disseminate key $k'$ to users $u_1$, $u_2$, $u_3$ and $u_4$.

***Marking*:** The LKH algorithm rekeys the group key every time the group changes. This approach, although provides backward and forward secrecy, does not scale very well, particularly for highly-dynamic groups. This may be remedied using *batch rekeying* [23], an approach where several group changes are accumulated in one key change, decreasing the communication cost needed to change the group key. One important parameter in key management algorithms using batching is the time between consecutive batching operations, known as *rekey period*. A low rekey period results in frequent rekeying, and potentially high overhead. A high rekey period makes a scheme more vulnerable to violations of security properties – in particular, the rekey period is an upper bound of how long a node that has left the group may continue to have access to data it is not authorized to.

A scheme applying batch rekeying for LKH was proposed in [23], [24]. The scheme, which we refer to as *Marking*, uses a similar key tree as LKH, with the difference that several changes of the key tree occur during a rekey operation. Figure 2 presents an example of updating the key tree using *Marking*, when the number of joins is smaller than the number of leaves. The joining member $u_9$ replaces in the tree one of the leaving nodes, $u_3$. We chose the Marking scheme given the benefits of batching in reducing the computation and communication overhead.

## IV. Design Space

We discuss design considerations unique to an overlay context in Section IV-A. This motivates an investigation of several schemes that we evaluate, summarized in Table I. We discuss strategies to achieve resilient key dissemination in Section IV-B, and for reducing overhead of rekey traffic in Section IV-C Finally, Section IV-D presents schemes that reduce the peak overhead for encryption under high group dynamics.

### A. Interplay between Overlays and Key Management

While incorporating techniques for key management, several unique aspects arising with overlays must be considered:

• *Choice of key dissemination structures:* In an IP Multicast architecture, data and keys are both disseminated using IP Multicast. With overlays, there is a rich space for designing structures for key distribution. One approach is to use the existing data delivery tree to also disseminate keys. However, rekey and data traffic differs in two ways. First, while applications can tolerate loss of data packets, losing a key impacts all data encrypted with that key and significantly affects application performance. Second, rekey traffic is bursty, and the overheads involved in disseminating keys in each rekey period are significant. Thus, a question we investigate is whether there are benefits to constructing additional structures specifically designed for key distribution.

• *Resilient key dissemination:* Reliable dissemination of rekey messages is important so hosts may successfully decrypt data. Several works studied reliable dissemination in the context of IP Multicast [33], [34] as well as looked at reliable key dissemination [20], [21]. Overlays however offer new opportunities to simplify the issues involved, that we investigate. A first technique involves using reliable transport protocols (e.g. TCP) in each hop of the overlay key dissemination structure. A second strategy involves disseminating keys through additional overlay structures that are more resilient.

• *Minimizing impact on scalability:* Overlay broadcasting systems have typically targeted several tens of thousands of recipients. While the addition of security services complicates the goal of achieving the same levels of scalability, it is nevertheless desirable to minimize the impact. We take steps towards this end, by investigating the effectiveness of two techniques to minimize the overhead incurred due to rekey traffic. A first technique involves distributing keys using overlay structures optimized to avoid redundant rekey traffic. A second technique involves reducing the number of keys that need to be distributed to the group.

### B. Resilient Key Dissemination Strategies

We implemented and evaluated several schemes for resilient key delivery with overlays. We focus on schemes that leverage opportunities unique to an overlay involving use of per-hop reliable transport protocols, and building additional overlays for key dissemination. Our focus is on minimizing loss of rekey packets rather than perfect reliability – occasional losses can be handled through recovery mechanisms, such as having nodes contact other members. The schemes chosen are:

• *Coupled-DataOpt*: This is the straight-forward approach to disseminate keys using the overlay multicast tree involved in data dissemination. We have considered three variants within this scheme: *Tree-UDP*, *Tree-TCP*, and *Tree-Unicast*. The *Tree-UDP* scheme is used as a base-line for comparison, with UDP being used on each hop of the overlay. This scheme is also normally used on the data delivery path, given video traffic is involved. With the *Tree-TCP*, and *Tree-Unicast* schemes, keys are transmitted using TCP in each hop. Further, in *Tree-Unicast*, the source sends the keys directly using unicast to nodes that have recently joined. This is motivated by the fact that a node that joins the group may take a certain period of time to connect to the tree, and may miss data and keys disseminated along the tree during this period. While the impact of missing data is relatively minor if connection times are small, the impact of missing a key can be more significant.
• *Gossip*: While the existing overlay data delivery structure may be used for delivering keys, additional structures that are more resilient may be constructed for disseminating keys. In particular, we consider a scheme where rekey packets are distributed using gossip-like mechanisms. From the list of current members of the group, the source selects $f$ members at random and distributes the rekey message to these members. When a member receives a rekey packet, it forwards it to $f$ members that it knows (learnt through the group membership component). The parameter $f$ helps trading-off between overhead and rapid dissemination – with higher $f$ values, the overhead is larger, but the time for the message to reach group members is shorter. Our implementation uses a value of $f = 15$, as this setting has been shown to work well in [35]. Each member maintains a list of keys he has already forwarded to avoid repeated transmissions of the same message. Finally, while the gossip technique does not guarantee all members will receive the packet, the probability that members miss rekey packets is low [35].

### C. Key Dissemination Strategies for Overhead Reduction

In addition to requiring resilient delivery, rekey traffic tends to be bursty, and the overhead involved in disseminating keys in each rekey period may be significant. While the *Coupled-DataOpt* scheme has the lowest level of complexity, the distribution of key messages can involve higher overhead. For example, Figure 3.a shows a LKH key tree where $k$ is the group key and $k_{1,2}$, $k_{3,4}$ are subgroup keys. The keys at the leaves of the tree (square boxes) are pair-wise keys of users $u_1$, $u_2$, $u_3$ and $u_4$, with the source. If the group key, $k$, changes, in order to be distributed, it is encrypted with the subgroup keys $k_{1,2}$ and $k_{3,4}$, resulting in messages $\{k\}_{k_{1,2}}$ and $\{k\}_{k_{3,4}}$. $\{k\}_{k_{1,2}}$ is of interest to $u_1$ and $u_2$, while $\{k\}_{k_{3,4}}$ is of interest to users $u_3$ and $u_4$. Figure 3.b shows a possible structure constructed by a *Coupled-DataOpt* scheme. In this case the source has to send messages $\{k\}_{k_{1,2}}$ and $\{k\}_{k_{3,4}}$ to all its children even though they may not need the keys. We consider two approaches to handling this:
• *Coupled-KeyOpt*: This scheme has been inspired by [28]. Here, the same overlay is used to disseminate data and keys, but the overlay is optimized for key distribution. The key

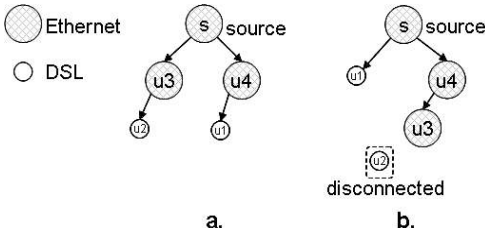



Fig. 4. a) *Coupled-DataOpt* structure is feasible. b) *Coupled-KeyOpt* structure is not feasible.

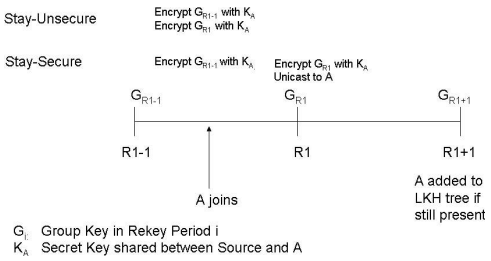



Fig. 5. Illustration of StayAware Heuristics.

dissemination tree matches the logical key tree such that keys are sent just to the nodes that may need them. Figure 3.c shows a structure optimized for key dissemination for the key tree in Figure 3.a. The structure ensures that an intermediate node will receive a key from its parent if and only if the node or at least one of its descendants needs the key. In this example, message $\{k\}_{k_{1,2}}$ is required for users in the subtree rooted at $u_1$ and message $\{k\}_{k_{3,4}}$ is required for users in the subtree rooted at $u_4$. Therefore, the source will send $\{k\}_{k_{1,2}}$ only to user $u_1$ and $\{k\}_{k_{3,4}}$ only to user $u_4$. We refer the reader to [28] for details.

While *Coupled-KeyOpt* may reduce rekeying overhead, it may not be feasible to use such an overlay for distributing data in bandwidth-demanding video streaming applications, given the need to honor the saturation degree of nodes. In the example, assume that $s$, $u_3$ and $u_4$ are behind an Ethernet connection which can support more than 2 children. Also, assume $u_1$ and $u_2$ are behind a DSL connection which cannot support any children. Figure 4 shows the same overlay structures from Figure 3 but highlighting the bandwidth resources of nodes. Figure 4.a is the structure constructed by *Coupled-DataOpt* (same as Figure 3.b) and Figure 4.b is the structure constructed by *Coupled-KeyOpt* (same as Figure 3.c). Clearly, the structure in Figure 4.b does not satisfy the bandwidth constraints of nodes. Honoring the constraints involves using a structure like Figure 4.a for data dissemination, however, this is not optimized for key delivery.
• *Decoupled*: To address the problem, we introduce and explore a third strategy, referred to as *Decoupled*, which uses two dissemination structures one for data and one for keys. Intuitively, such an architecture has the advantage of providing good performance for data delivery and reduction in overhead to diseminate key messages. The drawback in this case is that the source must maintain two structures instead of one, and hence there is additional complexity and overhead to maintain an extra structure.

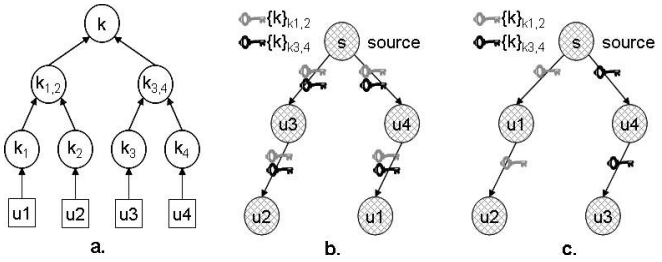



Fig. 3. a) A logical keys tree. b) An overlay structure optimized for data delivery. Intermediate nodes are positioned by their network characteristics. New keys are sent to all nodes. c) An overlay structure optimized for keys delivery. Intermediate nodes are positioned by their ID. New keys are sent only to nodes that need them.

### D. Decreasing Overhead by Reducing Encryptions

While overhead may be reduced by constructing overlays optimized for disseminating rekey packets, we also introduce and evaluate heuristics that minimize the number of encryptions with *Marking* for larger rekey periods, and consequently a lower rekey overhead. The heuristics illustrated in Figure 5 leverage group dynamics characterizations in overlay multicast deployments [15], Mbone measurements [36] and content delivery networks [37]. These studies indicate that an interesting class of group dynamic patterns show a bimodal stay time distribution, with a larger number of nodes that stay for a short while, and a smaller number of nodes that stay for long durations. The implications for algorithms like LKH or *Marking* is that a large number of group changes pertain to short-lived nodes that join the key tree and exit within a short while. Our heuristics seek to detect such nodes that have a disruptive effect on the system and prevent them from being added to the key tree. The schemes are:

• *StayAware*: In this scheme, a node that joins before a rekey period is not incorporated into the key tree until the next rekey period. At rekey period $R_1$, the source performs the following: (i) for a node $A$ that joined in the period $(R_1-1, R_1)$, the group key is encrypted with the pair-wise key shared with node $A$; (ii) a node $B$ that joined in the period $(R_1-2, R_1-1)$ is incorporated into the key tree; and (iii) normal rekey and key dissemination operations are performed. If a significant number of nodes stay for one to two rekey periods, then such nodes will not be incorporated into the key tree. This scheme reduces the number of changes to the tree compared to the original *Marking* algorithm. The cost is that an individual encryption must be performed to nodes that have joined in the last rekey period.

• *StayAware-Prefetch*: This scheme is a refinement of the above scheme which eliminates the need for an encryption using the pair-wise key for all nodes that join in a rekey period. This is achieved by providing a node $A$ joining in $(R_1-1, R_1)$ with the group key at both times $R_1-1$ and $R_1$. The two group keys are encrypted with the pair-wise key that $A$ shares with the source. No encryptions are needed for $A$ during the rekey event at $R_1$, and the source does not need to communicate with $A$ at $R_1$. The scheme has the potential to further reduce the number of encryptions at each rekey period and consequently the burst in communication. We note that as the scheme encrypts two keys each time a node joins, it may not reduce the average rate of encryptions performed per second. The scheme has the limitation that it may increase the vulnerability window. For example, in Figure 5, if A joins in the period from $R_1-1$ to $R_1$, and leaves in the same period, then, the vulnerability window is greater than $R$ and less than $2R$, where $R$ is the rekey period.

## V. Evaluation Goals and Methodology

Our evaluation is driven by several **goals**:

• *Resilient Key Dissemination:* (i) How effective is *Coupled-DataOpt* in ensuring resilient key dissemination under realistic models of node join/leave patterns, and what is the impact on application performance when key management algorithms are introduced? (ii) How does the choice of mechanism for key transport impact performance? How effective are protocols for per-hop reliability for key distribution in enabling end-to-end reliability of key dissemination? (iii) Under what circumstances could constructing additional structures specifically for resilient key dissemination be useful?

• *Reduction of Key Related Overhead:* (i) What is the communication overhead in incorporating key management with *Coupled-DataOpt*? (ii) How significant are the benefits of constructing key dissemination strategies optimized for minimizing overhead under realistic workloads? (iii) What is the impact on application performance with the *Coupled-KeyOpt* approach? (iv) How significant is the reduction in key dissemination overhead with the *Decoupled* approach? Does this reduction outweigh the additional overhead of maintaining two structures? (vi) How effective are *StayAware* and *StayAware-Prefetch* in reducing the number of encryptions for *Marking*?

Our evaluations are conducted in the context of implementations with the ESM broadcasting system [15]. ESM is one of the first operationally deployed systems and has seen significant real-world deployment. ESM constructs a multicast tree for data delivery. Members maintain knowledge about other group members using a gossip-like algorithm. Members monitor their performance in the tree, and adapt by changing parents if the current parent is unsatisfactory or leaves the group. All the cryptographic operations such as key generation, encryption and decryption have been implemented using the Openssl library [38].

### A. Performance Metrics

Our evaluations consider the following metrics:

• **Decryptable Ratio**: We consider the fraction of the raw bandwidth obtained using overlay multicast that can be successfully decrypted by a receiver. For instance, assume that the source transmits 100 data packets, of which the receiver gets 90 packets. Further, assume the receiver is only able to decrypt 80 packets because it did not have necessary keys at the time (due to loss or delay in rekey packets). Then, the Decryptable Ratio is $\frac{80}{90}$. The raw throughput itself is bounded by the source rate and depends on the performance of the underlying overlay multicast system.

• **Communication Overhead:** We consider the total bandwidth of all control messages sent or received by the source arising due to key management. Depending on the context, we also consider overhead due to other control messsages, such as the overhead of the base overlay multicast system itself. Our evaluations only focus on the communication overhead of the source, and do not consider the overhead at internal nodes. Given that overlay broadcasting is a bandwidth constrained application and the bursty nature of batch rekeying, we consider both *average* overhead and *peak* overhead.

• **Computation Overhead:** We consider the number of encryptions per second, as well as the number of encryptions performed by the source every rekey period. We only consider encryptions involved when new keys are created in the LKH tree, and do not consider encryptions associated with data packets. This overhead is incurred only at the source.

### B. Evaluation Methodology

We have conducted a detailed evaluation of key management schemes implemented in the ESM broadcasting system deployed on the Planetlab testbed. We performed experiments on Planetlab by emulating traces from real broadcast events that were conducted using application end-point overlay multicast [15]. The traces capture bandwidth-resource constraints of nodes, as well as information regarding user dynamic patterns. We emulated the traces, by having each client in a trace execute on a Planetlab host. Given that the peak number of clients in the traces we use is much larger than the number of Planetlab nodes, multiple simultaneously participating clients in the trace are mapped onto the same Planetlab node.

Our experiments are conducted with a streaming video rate of 420Kbps – the value used with other ESM deployments [15]. This also represents typical media streaming rates in real settings like [39], [40]. We use the outgoing bandwidth information of clients present in the trace, normalize it to the source rate, and obtain a degree for the client in the corresponding Planetlab instantiation. We assume a maximum degree of a client is six, which corresponds to the settings used in operational deployments reported in [15]. We directly use the same group dynamics pattern as in the trace to drive the experiment.

To ensure that we do not place an undue bandwidth demand on Planetlab nodes, we do not map more than three clients onto a Planetlab node. We also seek to maintain the invariant $\sum_{i=1}^{j} d_j < B/S$, where $j$ is the number of clients in the trace mapped to a Planetlab node $i$, $d_j$ is the degree of the client in the underlying trace, $B$ is the maximum outgoing bandwidth of Planetlab nodes, and $S$ is the source rate.

TABLE II
CHARACTERISTICS OF TRACE SEGMENTS USED

| Event | Deg 0 or 1 | Deg 6 | Peak Grp. Size | Joins | Leaves |
|---|---|---|---|---|---|
| *Conference1* | 33% | 67% | 42 | 8 | 9 |
| *Conference2* | 62% | 38% | 62 | 71 | 63 |
| *Portal* | 65% | 35% | 107 | 184 | 179 |
| *Competition* | 54% | 7% | 116 | 110 | 75 |
| *Rally* | 37% | 12% | 252 | 148 | 149 |

As each of the traces lasts for several hours, it is not feasible to emulate each of them entirely on Planetlab. Consequently, we emulate twenty minute segments of the trace. The clients already present in the trace at the start of the segment join in a burst over the first two minutes, then follow join/leave patterns exactly as in the trace for the next twenty minutes.

An important factor that impacts performance is whether nodes leave gracefully, or leave in an uninformed manner. A node that leaves gracefully informs the children that it is leaving, and continues to forward data for a few seconds. In particular, a value of 5 seconds is used in our evaluations like in [15]. On the other hand, a node could leave abruptly (e.g. due to machine failure), without informing the children. We have conducted sensitivity to the leave model, by performing experiments with both graceful and uninformed node departure models.

### C. Trace Characteristics

Table II summarizes the details of the trace segments used in our evaluations. We used traces from five different broadcasts. *Conference1* and *Conference2* are broadcasts of conferences, *Portal* refers to a broadcast conducted to a web portal, *Competition* is a broadcast of a sports event, and *Rally* refers to a broadcast of an election campaign. The first two columns show the constitution of hosts by presenting the percentage of hosts assigned a low degree (degree 0 or 1), or a high degree (degree 6). For the *Conference1*, *Conference2*, and *Portal* traces, these are the only two categories of nodes, however for the *Competition* and *Rally* traces, there are nodes with intermediate degree as well. The table also presents the peak size seen in the trace segment. The last two columns provide a sense of the group dynamics in the trace segments by presenting the number of joins and leaves that occur during that segment. Our evaluation study uses the *Rally* trace segment as the default, as it has the largest peak size, significant node dynamics, and significant heterogeneity in node constitution. The *Portal* trace segment is interesting in that while it has a smaller peak size, it has the highest churn rate with maximum group changes during a rekey period and median stay time of 3 minutes. The *Conference1* and *Conference2* trace segments have smaller group sizes. While the *Conference2* segment has a significant rate of node dynamics, *Conference1* is much less dynamic.

### D. Choice of Rekey Period

One important parameter in our experiments is the choice of the rekey period (defined in Section III) for the *Marking* algorithm. While a low rekey period results in frequent rekeying and potentially high overhead, a high rekey period requires

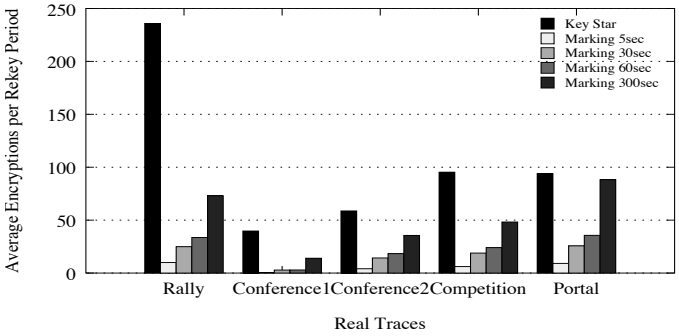



Fig. 6. Avg. encryptions per rekey event with *Marking* for various rekey periods. Each group of bars corresponds to a different trace. The first bar in each group is the average group size.

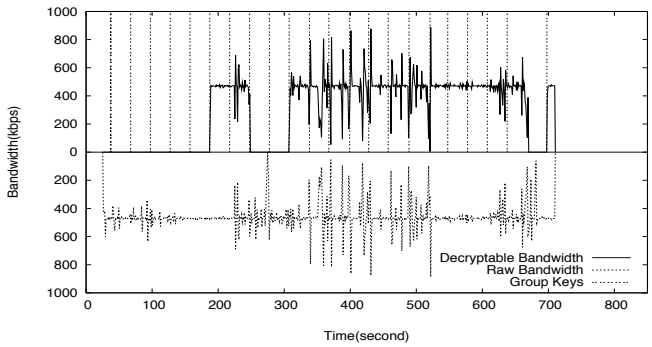



Fig. 7. Impact of loss of rekey packets on application performance.

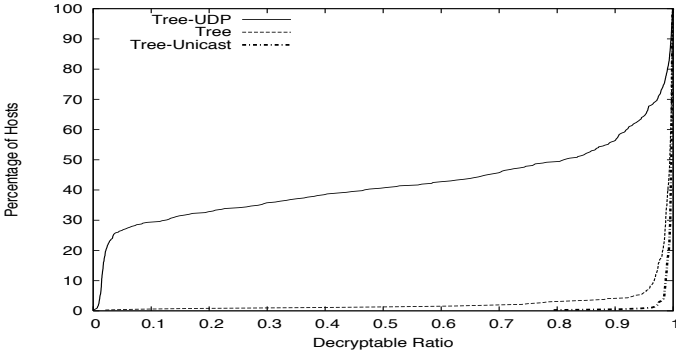



Fig. 8. Impact of key distribution schemes on application performance.

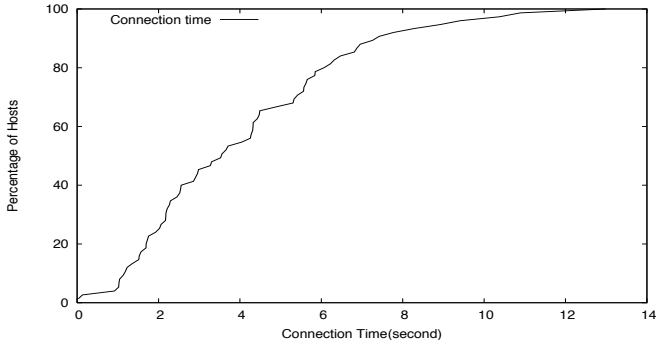



Fig. 9. Connection Time for hosts in the overlay system.

greater encryptions *per rekey event,* and larger peak overheads. Figure 6 compares the performance of *Marking* and *Key-Star* in terms of the number of encryptions per rekey event for various traces and multiple rekey periods. In each group of bars, the first bar represents the number of encryptions per rekey event for *Key-Star*. Note that for *Key-Star*, the number of encryptions is $O(N)$, where $N$ is the group size, independent of the frequency with which rekey events are conducted. The other 4 bars represent the number of encryptions per rekey event for *Marking* for periods of 5, 30, 60 and 300 seconds. The number of encryptions required on a rekey operation for *Marking* depends on the dynamics of the trace, and the length of the rekey period. For a rekey period of 300 seconds, the benefits of using *Marking* over the naive *Key-Star* are small for many traces, and there is almost no benefit for the *Portal* trace. Because of these results, the rest of the experiments with *Marking* focus on rekey periods of 60 seconds.

## VI. Resilient Key Dissemination Strategies

In this section, we consider different strategies for reliable key dissemination. We begin by considering the *Coupled-DataOpt* strategy, and evaluate its three variants *Tree-UDP*, *Tree-TCP*, and *Tree-Unicast*. We then conduct sensitivity studies to investigate how effective *Coupled-DataOpt* is under various regimes. Finally, we discuss potential limitations of *Coupled-DataOpt*, and in what regimes schemes like *Gossip* may be needed. Unless otherwise specified, for each experiment and for each point in every graph, we have conducted 5 runs and plotted the mean and standard deviation.

### A. *Evaluating Variants of* Coupled-DataOpt

To appreciate the impact of the loss of rekey packets on application performance, consider Figure 7 which depicts the performance of a user when *Tree-UDP* is used for key dissemination. The X-Axis represents time, while the Y-Axis depicts the bandwidth the user receives and can decrypt each second. For comparison, the negative Y-Axis shows the raw bandwidth the user receives each second. We note that though the source rate is fixed, the data obtained by receivers can be bursty. Each vertical line in the upper half of the graph, corresponds to the time when a receiver obtains a rekey packet containing a new group key. For the scenario in Figure 7, the node misses the new group and subgroup keys in the first rekey event after it joins. Consequently, it is unable to decrypt keys until 187 seconds later. Interestingly, the impulses show that the node keeps periodically receiving new versions of the group key in the intervening period – however it is unable to decrypt the keys since that requires a subgroup key which the node does not possess. The recovery at 187 seconds occurs because the subgroup keys the node was missing, have changed and have been sent from the source and the node receives those keys successfully. Similarly, the loss of a subgroup key prevents the node from being able to decrypt new group keys and data in the period 247-307.

Figure 8 shows the *Decryptable Ratio* achieved with *Tree-TCP*, and *Tree-UDP* if a rekey period of 60 seconds is used. We make several observations. First, the performance with *Tree-UDP* is poor. Using per-hop UDP results in loss of rekey packets which prevents the node from decrypting raw data it may receive. Second, *Tree-TCP* does much better than *Tree-*

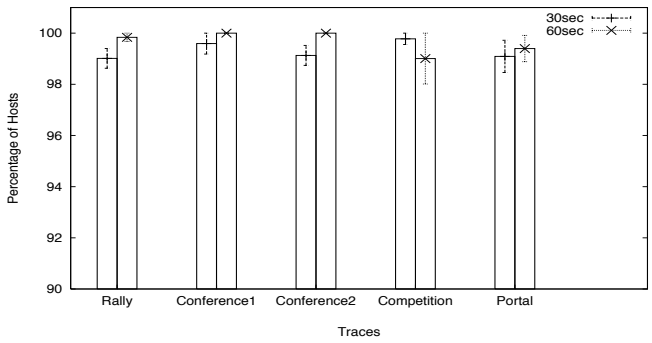



Fig. 10. Performance of *Coupled-DataOpt* with various traces.

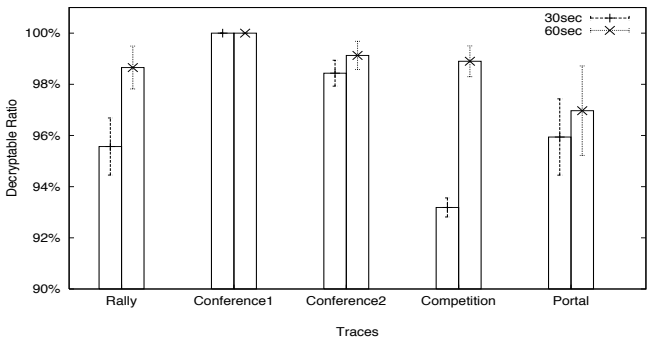



Fig. 11. Performance of *Coupled-DataOpt* with Ungraceful Departures.

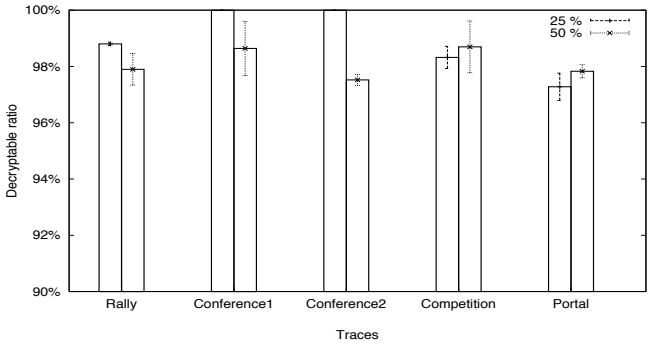



Fig. 12. Performance of *Coupled-DataOpt* with Burst Departures.

*UDP* with *Decryptable Ratio* greater than 0.97 for over 85% of the nodes, indicating that using TCP for key dissemination can have significant benefits. Third, *Tree-TCP* has a tail, and some nodes do not perform as well. Further, *Tree-Unicast* performs better than *Tree-TCP* and helps improve the performance of the tail.

The tail shown by *Tree-TCP* is because a node that joins the group, or is disconnected because its parent left the group, may be disconnected from the overlay tree for a certain period of time. During this time, the node is unable to receive data or keys distributed along the tree. While the impact of missing data is relatively minor if reconnection times are small, the impact of missing a key can be more significant. We refer to the time that a node takes to join an overlay multicast tree, or reconnect when a parent leaves as the *Connection Time*. Figure 9 plots a CDF of the *Connection Time* of nodes in the ESM overlay multicast system. Over 70% of the nodes have a *Connection Time* of less than 5 seconds, though this can be as high as about 10-12 seconds in some cases. *Tree-Unicast* helps improve performance by having the source unicast rekey packets to nodes that joined the group in the last rekey period. *Connection Time* is a concern also for node departures. For node departures, the problem is partially ameliorated because over 40% of nodes that depart do not have children, and thus their leave does not have an impact on other nodes.

### *B. Sensitivity Studies*

Having shown that the *Tree-Unicast* variant of the *Coupled-DataOpt* scheme is the most effective, the rest of our experiments employ this variant, and we use the terms *Coupled-DataOpt* and *Tree-Unicast* interchangeably. We now consider the performance of the scheme under various traces, and group dynamics models.

**Traces:** Figure 10 considers the performance with *Coupled-DataOpt* obtained with the entire set of traces. Each group of bars corresponds to a different trace. For each group, there are 2 bars, one indicating the performance at a rekey period of 30 seconds, and the other indicating performance at rekey period of 60 seconds. Each bar represents the fraction of receivers for which the *Decryptable Ratio* is greater than 90% for a given trace and rekey period. For a rekey period of 30 seconds, over 98% of receivers see a *Decryptable Ratio* greater than 90%. If a rekey period of 60 seconds is used, the performance results are even stronger, with over 99% of receivers seeing a *Decryptable Ratio* greater than 90% for all traces. The performance is better with a higher rekey period because it decreases the probability of a node departure happening shortly before a rekey event.

**Ungraceful Departures:** Our experiments so far assume that nodes depart gracefully, forwarding packets for 5 seconds before they abandon the group. We next consider the case when nodes leave abruptly. In this case, the underlying system takes a detection time of about 5 seconds to identify the departure, and switch the node to a new parent. During this time more rekey packets can be lost for children looking for a new parent. Figure 11 shows the *Decryptable Ratio* for various traces and multiple rekey periods. Each bar represents the fraction of receivers for which the *Decryptable Ratio* is greater than 90% for a given trace and rekey period. For a rekey period of 30 seconds, over 92% of receivers see a *Decryptable Ratio* greater than 90%, and if a rekey period of 60 seconds were used, over 95% of receivers see a *Decryptable Ratio* greater than 90% for all traces.

**Burst Departures:** We conducted experiments with burst departures, by modifying the real traces to make a randomly selected percentage of the nodes leave at the same time. These nodes abandon the group 700 seconds after the beginning of the traces, when the traces have reached their peak sizes. Figure 12 considers the *Decryptable Ratio*. Each group of bars represents a different trace. There are two bars per group, one shows the performance when 25% of the nodes leave at the same time, the other shows the performance when 50% of the nodes leave at the same time. Each bar represents the fraction

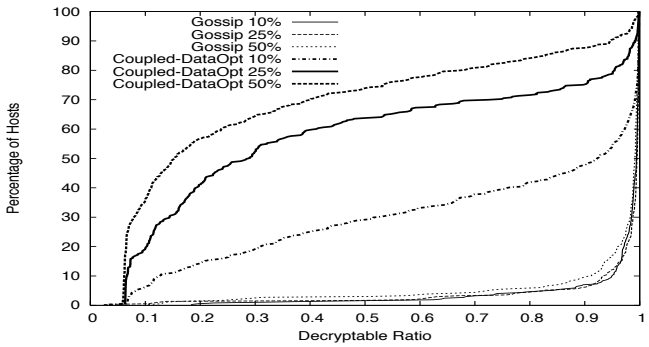



Fig. 13. Impact of adversarial attacks on key distribution. Top 3 curves are *Coupled-DataOpt* and bottom 3 curves are *Gossip*. Each curve corresponds to a different fraction of insider nodes.

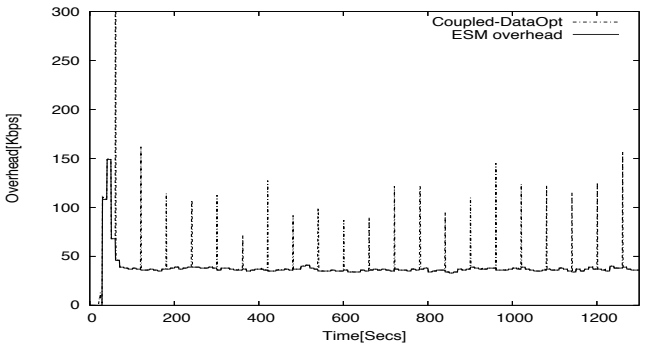



Fig. 14. Overhead per second at the source when mantaining *Coupled-DataOpt* structure, for *Rally* trace and rekey period 60 seconds.

of receivers for which the *Decryptable Ratio* is greater than $90\%$ for a given trace and rekey period $60$ seconds. For a burst departure of $25\%$ and $50\%$ of the nodes, over $96\%$ of the receivers see a *Decryptable Ratio* greater than $90\%$ for all the traces.

### C. Potential Limitations of Coupled-DataOpt

Given the strong performance of *Coupled-DataOpt* in the environments above, we do not present results for the more resilient *Gossip* scheme - overall in these settings *Gossip* has marginal performance benefits, but incurs additional overheads. We next consider **adversarial environments,** where nodes part of the overlay choose deliberately not to forward keys. Such scenarios are a concern in an overlay multicast system because intermediate nodes forward data and keys, and cannot be trusted in insider attack scenarios. This is in contrast to IP Multicast where routers are part of trusted network infrastructure. While in general insiders could choose to perform denial of service by not forwarding data packets, such attacks are easier to identify – existing adaptive mechanisms in overlay multicast could detect a gap in packets being forwarded, and have the node switch to other parents in the overlay. However, a node could potentially cause as much disruption, but prove harder to detect by simply not forwarding the rekey packets, even though it forwards data packets.

Figure 13 illustrates the performance of *Coupled-DataOpt* and *Gossip* in the presence of insiders by showing the CDF of the *Decryptable Ratio* achieved. A certain fraction of receivers are considered insiders, who do not forward key packets, but forward data packets and otherwise co-operate fully with the overlay protocol. The performance of *Coupled-DataOpt* scheme degrades very sharply with the fraction of insiders, even with $10\%$ of the nodes being insiders. In contrast, *Gossip* performs much better with most receivers seeing a high *Decryptable Ratio* even with large fractions of insiders.

While our results show *Gossip* may be preferable to *Coupled-DataOpt* in adversarial environments, we close with several comments. First, *Gossip* does incurs higher overheads, and it may be feasible to achieve better performance with *Coupled-DataOpt* by modifying the adaptation mechanisms of the overlay system to account for decryptable packets. Second, if data itself is delivered using structures other than trees [11], [7], [9], [30], [17], [31], the resiliency of such structures could be exploited for key dissemination instead of using *Gossip*. Finally, a complete solution for key distribution under adversarial regimes must consider various other insiders attacks, and is deferred to future work.

## VII. Strategies for Reducing Overhead

In this section we study the communication overhead incurred when integrating key management schemes with an overlay broadcast system. We first present results that compare the different strategies to disseminate keys, then we evaluate techniques for reducing the number of encryptions, which could in turn decrease the overheads.

### A. Overhead Optimized Key Dissemination Strategies

In this section, we evaluate the benefits of optimizing the overlay for key dissemination and decoupling key and data dissemination structures. We consider the *Coupled-DataOpt*, *Coupled-KeyOpt* and *Decoupled* strategies discussed in Section IV-C. Our implementation of the *Coupled-KeyOpt* scheme follows the recent proposal in [28], as discussued in Section IV-C. For the *Decoupled* scheme, our implementation uses ESM for data delivery and a key-optimized structure augmented with reliable dissemination mechanisms for key delivery. We first look at overheads with *Coupled-DataOpt*. Then, we evaluate the feasibility of using the *Coupled-KeyOpt* strategy for delivering data for bandwidth demanding brodcasting applications using simulations. We show the strategy does not satisfy the saturation degree and physical bandwidth constraints of nodes. The rest of the section focuses on comparing the *Decoupled* and *Coupled-DataOpt* strategies.

*1) Overhead with* Coupled-DataOpt*:* Figure 14 shows the overhead for the *Coupled-DataOpt* scheme as a function of time for the *Rally* trace and a rekey period of 60 seconds. The overhead is sampled every second and considers all control messages at the source, including those due to key management and maintenance of the overlay dissemination structure. The curve sees periodic spikes corresponding to rekey events. The overhead due to the base ESM system is about 50Kbps. When key management is added, the spikes corresponding to rekey events are as large as 150Kbps, but the additional overhead is minimal for the rest of the time. Note that the source is also serving data at 420 Kbps for each

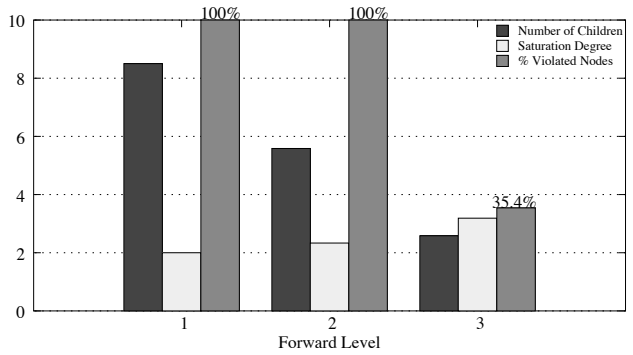



Fig. 15. Avg. number of children imposed by overlay (first bar), saturation degree (second bar) and percentage of nodes with saturation degree violated (third bar - scaled down by a factor of 10), when *Coupled-KeyOpt* is used to deliver keys.

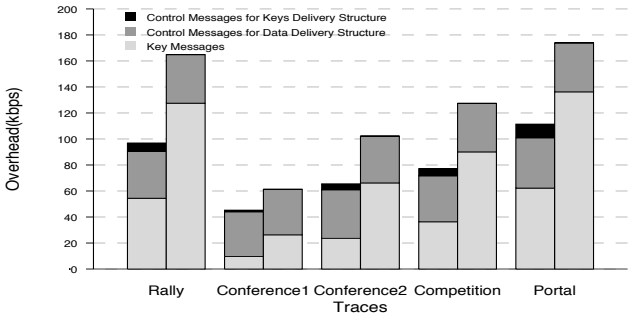



Fig. 16. Peak overhead of *Decoupled* (first bar) and *Coupled-DataOpt* (second bar) with various traces for rekey period 60 seconds.

of its children. These trends are true over multiple traces, and for larger rekey periods the overhead increases.

*2) Feasibility of* Coupled-KeyOpt*:* Figure 15 presents results from a simulation study of the *Coupled-KeyOpt* scheme conducted using the *Rally* trace. Each group of bars correspond to nodes at a particular *forwarding level* in the tree produced by *Coupled-KeyOpt*. The source is at forwarding level 0, its direct children at level 1, and so on. For each forwarding level, three bars are shown corresponding to: (i) the average number of children in the *Coupled-KeyOpt* structure for nodes at that level; (ii) the average saturation degree (maximun degree imposed by node out bandwidth) for nodes at that level; and (iii) the fraction of nodes at that level which have more children than permitted by their saturation degree. As can be seen from the figure (third bar), 100% of the nodes at forward level 1 and 2, and 35.4% of the nodes in level 3 are violating their saturation degree. The average number of children (first bar) exceeds the average saturation degree (second bar) for levels 1 and 2, and exceeds the maximum saturation degree any node has in our experiments for level 1. These results indicate that it is not feasible to use the *Coupled-KeyOpt* strategy for bandwidth-demanding applications. The reason is that the goal of matching the dissemination tree with the logical key tree built by schemes like LKH or *Marking* is at odds with the goal of honoring the heterogeneous access bandwidth constraints of participating nodes.

*3) Benefits of* Decoupled*:* Given the feasibility concerns with *Coupled-KeyOpt*, the rest of the section focuses on comparing the *Decoupled* and *Coupled-DataOpt* strategies. Since both strategies employ the same ESM overlay for data distribution, we expect the application performance to be similar, and our comparisons primarily focus on the overheads involved.

Figure 16 studies the strategies across the complete set of traces. The figure shows the *peak* communication overhead incurred with *Decoupled* and *Coupled-DataOpt* with a rekey period of 60 seconds and for various traces, by sampling overhead at every second after the first rekey period and identifying the peak value. The overhead during the first rekey period is not considered since the system is not in a steady state at that time. Each group of bars corresponds to a different trace. The first bar in each group is the peak overhead incurred with *Decoupled*, and the second bar is the peak overhead with *Coupled-DataOpt*. Each bar consists of various overhead components: key messages, control messages for data-delivery structure, and control messages for keys-delivery structure with *Decoupled*. We make two observations. First, for all traces, the overhead of *key messages* incurred with *Decoupled* is reduced by $50\%$ to $67\%$ of that incurred with *Coupled-DataOpt*. For the *Rally* trace, the overhead of rekey messages is reduced from 120Kbps to 60Kbps. This is expected since *Decoupled* uses a separate optimized keys-delivery structure. Second, for all traces, the *total* overhead incurred with *Decoupled* is reduced by $20\%$ to $50\%$ of that incurred with *Coupled-DataOpt*. For the *Rally* trace, the overhead is reduced from 160Kbps to 100Kbps. For some small-sized and less dynamic traces like *Conference1*, the reduction in total overhead made by *Decoupled* is not so significant as in key messages. The main reason is that for those traces the overhead of maintaining the data-delivery structure is significant, so reducing only key messages can not reduce the total overhead greatly. Another reason is that for *Decoupled*, there is an additional overhead of maintaining the separate keys-delivery structure. However, for larger and more dynamic traces like *Rally* where the overhead of key messages is the major component, the reduction in total overhead is still significant. We also performed experiments with the *Rally* trace for a rekey period of 300 seconds and the reduction in peak overhead of *Decoupled* versus *Coupled-DataOpt* is even more significant.

Overall, these results show that the reduction in peak overheads due to key dissemination with the *Decoupled* approach can outweigh the overhead of maintaining an additional structure. Further, these benefits may be realized while still honoring physical access bandwidth constraints, and achieving good application performance.

### *B. Decreasing Overheads by Reducing Encryptions*

The results above show that using *Decoupled*, and explicitly optimizing the key delivery structure to match the logical key tree can significantly reduce peak overheads. Nevertheless, the peak overheads are still significant. We next present results with the *StayAware*, and *StayAware-Prefetch* schemes, as a means to reducing the large number of encryptions per rekey

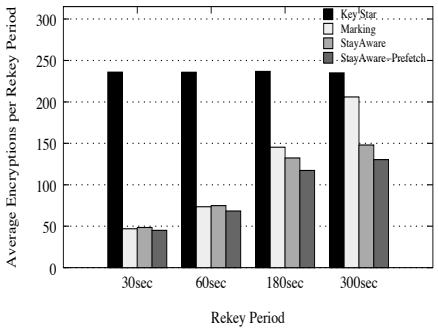



Fig. 17. Rally Trace. Rate of encryptions per second.

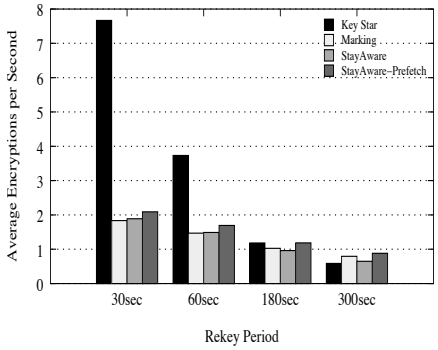



Fig. 18. Rally Trace. Avg. number of encryptions per rekey event.

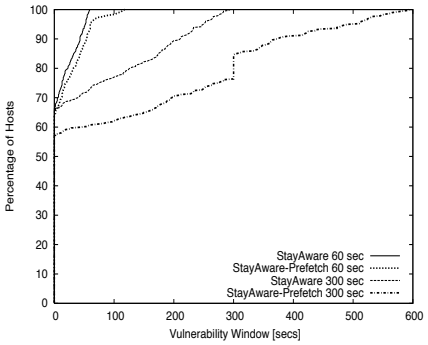



Fig. 19. Rally Trace. Vulnerability window for various rekey periods.

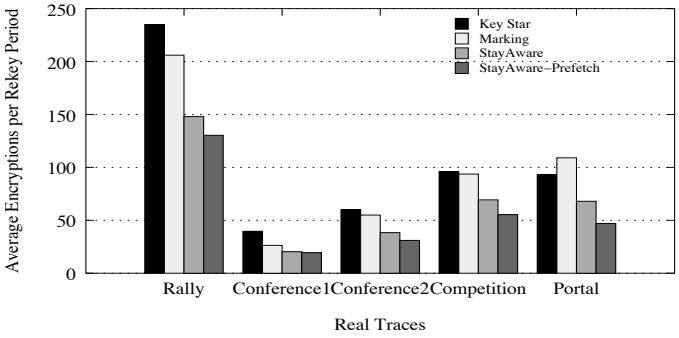



Fig. 20. Sensitivity to traces. Avg. number of encryptions per rekey event.

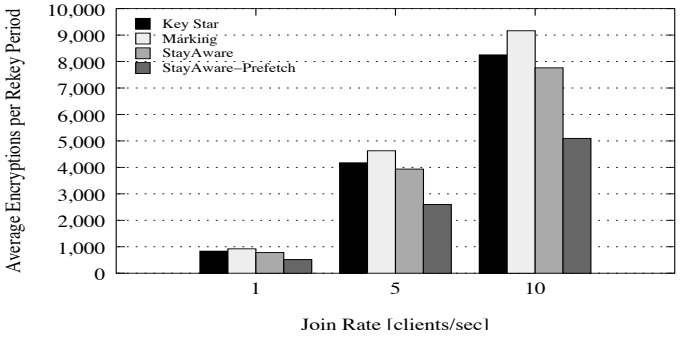



Fig. 21. Synthetic traces. Avg. number of encryptions per rekey event.

event with *Marking*. Such reduction could help further reduce overheads.

Figure 18 plots the average number of encryptions per rekey event for the *Rally* trace. Each group of bars corresponds to a rekey period, and shows the average number of encryptions per rekey event for the *Key-Star*, *Marking*, *StayAware*, and *StayAware-Prefetch* schemes. *StayAware* improves performance over *Marking* for rekey periods of 180 seconds and higher. *StayAware-Prefetch* further reduces the number of encryptions. Figure 17 plots the average rate of encryptions per second with the four schemes. The performance of the schemes we introduce is similar to *Marking*. To summarize, while the schemes we introduce do not alter the average rate of encryptions per second, they can reduce the number of encryptions performed each rekey period, and *StayAware-Prefetch* is more effective in this regard.

A limitation of *StayAware-Prefetch* is that it can result in an increase in the vulnerability window. Figure 19 plots the CDF of the vulnerability window with the *Rally* trace for rekey periods 60 and 300 seconds. For a rekey period of $R$, hosts using *Marking* have a vulnerability window ranging from 0 to $R$, while when using *StayAware-Prefetch*, the vulnerability window may range from 0 to $2R$. The increase only occurs for nodes that join the group and stay for a short while. This might be acceptable in certain scenarios – for example, an application may offer a free preview of content, with no charge for the first few minutes, or may make a minimum charge where the user is charged for a minimum period of time.

**Sensitivity to Trace:** Figure 20 plots the number of encryptions per rekey event for the entire set of traces, keeping the rekey period fixed at 300 seconds. In the three larger traces, *StayAware* performs better than *Marking*, but again benefits are more significant with *StayAware-Prefetch*. For the smaller conference traces, the benefits are less significant, but the peak sizes of these traces are small to begin with.

**Scaling:** Finally, we have considered the impact of scaling. Since large traces were not available to us, we used synthetic traces. We model node arrival using a Poisson process, and node session durations using a Pareto process. These choices are based on group dynamics observed in overlay multicast deployments [15], Mbone measurements [36] and content delivery networks [37]. For the Pareto distribution, we assume an alpha value of 1 and a mean stay time of 1800 seconds - which is in the range of mean stay times seen with the real traces. We vary the join rate of the Poisson process between 1 and 10 nodes per second, leading to group sizes varying between 800 and 8500 nodes. Figure 21 plots the performance of the four schemes. Each group of bars corresponds to a different join rate, with one bar for each trace. The rekey period was 300 seconds. Once again, in all settings, *StayAware* performs better than *Marking*, and benefits are more significant with *StayAware-Prefetch*. Finally, we have also conducted experiments keeping the join rate fixed, but varying the mean stay time. The trends are similar, and we omit the results for space constraints.

## VIII. Summary and Discussion

Most prior work on key management algorithms was conducted in the context of IP Multicast. In this paper, we study the unique opportunities and challenges when incorporating key management schemes in an overlay architecture. We have systematically explored the design space, and conducted the first study that involves implementation, performance evaluation in real Internet environments, and real traces of join/leave dynamics.

We study the potential of a decoupled architecture that uses two dissemination structures, one for data and one for keys, compared to coupled architectures in which the same structure is used for distributing both data and keys. Our results show that:

- The *Coupled-DataOpt* strategy is effective in achieving resilient key dissemination, and constructing additional structures does not significantly improve performance. The key ingredients behind achieving good performance involves using TCP to ensure per-hop reliability (feasible only in the overlay context), and improving performance by considering the convergence properties of overlays. For the *Rally* trace and 60 second rekey periods, with *Coupled-DataOpt* $99\%$ of receivers see a *Decryptable Ratio* greater than $0.99$, and the results hold across multiple traces, and group dynamics models.
- *Coupled-DataOpt* incurs high peak overheads associated with key dissemination, and it is important to explicitly optimize overlays for key distribution to minimize overheads. However, it is not feasible to use *Coupled-KeyOpt* for distributing data in bandwidth-demanding video streaming applications, given the need to honor physical access bandwidth constraints of nodes. With *Decoupled*, physical access bandwidth constraints are honored. Further, the reduction in peak overheads due to key dissemination outweighs the overhead of maintaining an additional structure. For the *Rally* trace and 60 seconds rekey period, the peak overhead of *Decoupled* is 44% less than *Coupled-DataOpt* while the average overheads are comparable.

We introduce heuristics to achieve further reductions in overhead by minimizing the number of encryptions that must be performed in each rekey period by considering the durations nodes stay in the group. For the *Rally* trace, and a rekey period of 300 seconds the *StayAware* scheme reduces the average encryptions per rekey period from 200 to 150. *StayAware-Prefetch* can further reduce this to 120, with a modest increase in the vulnerability window.

While our studies have focused on tree-based data delivery structures, some of our findings also apply to mesh and multi-tree based approaches. These include techniques to lower overheads by decoupling data and key dissemination, and reducing the number of encryptions with *Marking*. While our study on resilient key dissemination does depend on the scheme for data delivery, we expect that the performance of *Coupled-DataOpt* is only enhanced when more resilient structures like mesh and multi-trees are used. That said, a more systematic investigation of implications for mesh and multiple-tree based approaches could be an interesting future direction.

In this paper, we have taken important first steps towards enabling confidentiality of data delivery with peer-to-peer streaming, an area where the emphasis has traditionally been on performance and scalability. While we have looked at mechanisms to reduce overheads due to key management, scaling peer-to-peer systems to hundreds of thousands of receivers is a challenge when security issues are considered. Future work must focus on both exploring fundamental tradeoffs between security and overheads, as well as techniques to achieve better scaling.

## References


[1] Y. Chu, S. G. Rao, and H. Zhang, "A Case for End System Multicast," in *Proceedings of ACM Sigmetrics*, June 2000.

[2] P. Francis, "Yoid: Extending the Internet Multicast Architecture," Apr. 2000.

[3] J. Jannotti, D. Gifford, K. L. Johnson, M. F. Kaashoek, and J. W. O. Jr., "Overcast: Reliable Multicasting with an Overlay Network," in *Proceedings of the Fourth Symposium on Operating System Design and Implementation (OSDI)*, Oct. 2000.

[4] D. Pendarakis, S. Shi, D. Verma, and M. Waldvogel, "ALMI: An Application Level Multicast Infrastructure," in *Proceedings of 3rd Usenix Symposium on Internet Technologies & Systems (USITS)*, March 2001.

[5] K. Sripanidkulchai, A. Ganjam, B. Maggs, and H. Zhang, "The Feasibility of Supporting Large-Scale Live Streaming Applications with Dynamic Application End-Points," in *Proceedings of ACM SIGCOMM*, 2004.

[6] S. Banerjee, B. Bhattacharjee, and C. Kommareddy, "Scalable Application Layer Multicast," in *Proceedings of ACM SIGCOMM*, Aug. 2002.

[7] M. Castro, P. Druschel, A. Kermarrec, A. Nandi, A. Rowstron, and A. Singh, "SplitStream: High-bandwidth Content Distribution in Cooperative Environments," in *Proceedings of SOSP*, 2003.

[8] M. Castro, P. Druschel, A. Kermarrec, and A. Rowstron, "Scribe: A Large-Scale and Decentralized Application-Level Multicast Infrastructure," in *IEEE Journal on Selected Areas in Communications Vol. 20 No. 8*, Oct. 2002.

[9] D. Kostic, A. Rodriguez, J. Albrecht, and A. Vahdat, "Bullet: High Bandwidth Data Dissemination Using an Overlay Mesh," in *Proceedings of SOSP*, 2003.

[10] J. Liebeherr and M. Nahas, "Application-layer Multicast with Delaunay Triangulations," in *Proceedings of IEEE Globecom*, Nov. 2001.

[11] V. Padmanabhan, H. Wang, P. Chou, and K. Sripanidkulchai, "Distributing Streaming Media Content Using Cooperative Networking," in *Proceedings of NOSSDAV*, May 2002.

[12] S. Ratnasamy, M. Handley, R. Karp, and S. Shenker, "Application-level Multicast using Content-Addressable Networks," in *Proceedings of NGC*, 2001.

[13] W. Wang, D. Helder, S. Jamin, and L. Zhang, "Overlay Optimizations for End-host Multicast," in *Proceedings of Fourth International Workshop on Networked Group Communication (NGC)*, Oct. 2002.

[14] S. Q. Zhuang, B. Y. Zhao, J. D. Kubiatowicz, and A. D. Joseph, "Bayeux: An Architecture for Scalable and Fault-Tolerant Wide-Area Data Dissemination," in *Proceedings of NOSSDAV*, Apr. 2001.

[15] Y. Chu, A. Ganjam, T. S. E. Ng, S. G. Rao, K. Sripanidkulchai, J. Zhan, and H. Zhang, "Early Experience with an Internet Broadcast System Based on Overlay Multicast," in *Proceedings of USENIX*, June 2004.

[16] "Tmesh broadcast system," http://warriors.eecs.umich.edu/tmesh/tmeshv.html.

[17] X. Zhang, J. Liu, B. Li, and T.-S. P. Yum, "DONet/CoolStreaming: A Data-driven Overlay Network for Live Media Streaming," in *Proceedings of IEEE INFOCOM*, 2005.

[18] S. Ali, A. Mathur, and H. Zhang, "Measurement of commercial peer-to-peer live video streaming," in *Proc. of Workshop on Recent Advances in P2P Streaming*, 2006.

[19] X. Hei, C. Liang, J. Liang, Y. Liu, and K. Ross, "Insights into pplive: A measurement study of a large-scale p2p iptv system," in *Proc. Workshop on Internet Protocol TV (IPTV) services over World Wide Web in conjunction with WWW2006*, May 2006.

[20] A. Perrig, D. Song, and J. Tygar, "ELK, a new protocol for efficient large-group key distribution," in *Proceedings of the 2001 IEEE Symposium on Security and Privacy*, May 2001, pp. 247–262.

[21] C. K. Wong and S. S. Lam., "Keystone: A group key management service." in *International Conference on Telecommunications, ICT 2000*, 2000.

[22] D. Wallner, E. Harder, and R. Agee, "Key Management for Multicast: Issues and Architectures," RFC 2627, June 1999.

[23] X. S. Li, Y. R. Yang, M. G. Gouda, and S. S. Lam, "Batch rekeying for secure group communications," in *Proceedings of the tenth international conference on World Wide Web*. ACM, 2001, pp. 525–534.

[24] X. Zhang, S. Lam, D. Lee, and Y. Yang, "Protocol design for scalable and reliable group rekeying," *IEEE/ACM Transactions on Networking*, vol. 11, no. 6, Dec. 2003.

[25] S. Setia, S. Koussih, S. Jajodia, and E. Harder, "Kronos: A Scalable Group Re-keying Approach for Secure Multicast," in *2000 IEEE Symposium on Security and Privacy*. IEEE, May 2000, pp. 215–218, oakland, CA.

[26] S. Mittra, "Iolus: A framework for scalable secure multicasting," in *Proceedings of the ACM SIGCOMM '97*, September 1997, pp. 277–288.

[27] S. Rafaeli and D. Hutchison, "A survey of key management for secure group communication," *ACM Computing Surveys*, vol. 35, no. 3, pp. 309 – 329, 2003.

[28] X.B.Zhang, S.S.Lam, and H.Liu, ""Efficient Group Rekeying Using Application Layer Multicast"," in *Proceedings of IEEE ICDCS*, 2005.

[29] C. Abad and I. Gupta, ""Adding confidentiality to application-level multicast by leveraging the muticast overlay"," in *"Proceedings of IEEE 4th International Workshop on Assurance Distributed Systems and Networks (ADSN)"*, 2005.

[30] V. Pai, K. Tamilmani, V. Sambamurthy, K. Kumar, and A. Mohr, "Chainsaw: Eliminating trees from overlay multicast," in *Proc. The 4th International Workshop on Peer-to-Peer Systems (IPTPS)*, 2005.

[31] V. Venkataraman, K. Yoshida, and P. Francis, "Chunkyspread: Heterogeneous unstructured end system multicast," in *Proc. The 14th IEEE International Conference on Network Protocols*, 2006.

[32] C. K. Wong, M. G. Gouda, and S. S. Lam, "Secure group communications using key graphs," *Transactions on Networking*, vol. 8, no. 1, pp. 16–30, 2000.

[33] S. Floyd, V. Jacobson, C. Liu, S. McCanne, and L. Zhang, "A Reliable Multicast Framework for Light-weight Sessions and Application Level Framing," in *IEEE/ACM Transactions on Networking Volume 5, Number 6, pp. 784-803*, Dec. 1997.

[34] S. Paul, K. Sabnani, J. Lin, and S. Bhattacharyya, "Reliable Multicast Transport Protocol (RMTP)," in *IEEE Journal on Selected Areas in Communications Vol. 15 No. 3*, April 1997.

[35] C. Tang, R. Chang, and C. Ward, "Gocast: Gossip-enhanced overlay multicast for fast and dependable group communication," in *Proceedings of IEEE DSN*, 2005.

[36] K. C. Almeroth and M. H. Ammar, "Collecting and Modeling the Join/Leave Behavior of Multicast Group Members in the MBone," in *Proceedings of International Symposium on High Performance Distributed Computing (HPDC)*, Aug. 1996.

[37] K. Sripanidkulchai, B. Maggs, and H. Zhang, "An Analysis of Live Streaming Workloads on the Internet," in *Proceedings of Internet Measurement Conference*, 2004.

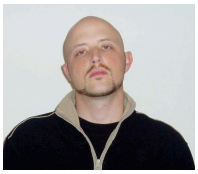
[38] OpenSSL Project team, "Openssl," May 1999, http://www.openssl.org/.

[39] "Akamai Technologies, Inc." http://www.akamai.com.

[40] "Inktomi," http://www.inktomi.com/.


**Ruben Torres** is a Ph.D. student in the School of Electrical and Computer Engineering at Purdue University and a research assistant in the Internet Systems Lab. Ruben received his MS in electrical and computer engineering from Purdue University in 2006 and his BS in electronics and communications engineering from the University of Panama in 2002. His research interests include distributed systems and security, with an emphasis on peer-to-peer networks.

**Xin Sun** received his B.Eng. degree from University of Science and Technology of China in 2005. He is currently a Ph.D. student in the School of Electrical and Computer Engineering at Purdue University, and a research assistant in the Internet Systems Lab. His research interests are in overlay networks with an emphasis on security, and network management.



**AAron Walters** is a founding partner of Volatile Systems. Prior to joining Volatile Systems, he was the Director of Forensics at Komoku, Inc. and the Research Lead of BAE Systems Advanced Detection research group. While a member of CERIAS and the Dependable and Secure Distributed Systems Laboratory, he earned a MS in Computer Science from Purdue in 2006. He received a BS in Computer Engineering from the University of Notre Dame in 2001. His research interests include distributed systems, anomaly detection, digital forensics, and multi-sensor data fusion.

**Cristina Nita-Rotaru** is an Assistant Professor in the Computer Science department of the Purdue University. She is the director of the Dependable and Secure Distributed Systems Laboratory. She received the BS and MS degrees in Computer Science from Politechnica University of Bucharest, Romania, in 1995 and 1996, and a PhD degree in Computer Science from The Johns Hopkins University in 2003. She served on the technical program committee of numerous conferences such as INFOCOM, ICDCS, ICNP, MOBIHOC. She received the National Science Foundation CAREER award in 2006. Her research interests include secure distributed systems, network security protocols in wired and wireless networks. She is a member of the ACM and IEEE Computer Society.

**Sanjay G. Rao** received the Bachelor's degree in Computer Science and Engineering from the Indian Institute of Technology, Madras in 1997 and the Ph.D from the School of Computer Science, Carnegie Mellon University in 2004. He is an Assistant Professor in the School of Electrical and Computer Engineering, Purdue University, West Lafayette, where he leads the Internet Systems Laboratory. He was a visiting researcher in the Network Measurement and Management group at AT&T Research, Florham Park, New Jersey in Summer 2006. He wrote the first paper to show the viability of an overlay approach to multicast, and led the design of an overlay broadcasting system based on a peer-to-peer architecture. His research interests are in Networking, and Distributed Systems. His current projects are in Peer-to-Peer systems, and Network Management. Prof. Rao is a member of the ACM, and serves on the Technical Program Committees of several workshops and conferences including IEEE IWQoS 2007, and IEEE Infocom 2008.