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*— Most prior work on the use of key management algorithms to enable confidentiality of video delivery has been conducted in the context of IP Multicast. In this paper, we consider the unique challenges and opportunities of integrating key management algorithms in an overlay multicast system. We conduct a systematic and extensive performance evaluation of strategies for key dissemination in the context of an operational overlay broadcasting system on the Planetlab testbed using real traces of join/leave dynamics. We show that leveraging TCP in each hop of the overlay dissemination structure can significantly simplify reliable key dissemination. The performance can be further enhanced if convergence properties of overlays are considered. We show that using two specialized dissemination structures, one for data and one for keys, potentially achieves low overhead for key dissemination without sacrificing application performance. To our knowledge, this is the first paper to study key management schemes in an overlay context using real implementation and Internet experiments and the first to consider issues in resilient key dissemination with overlays.**

I. Introduction

In recent years, overlay multicast has emerged as an alternative architecture to IP Multicast for enabling group communication applications over the Internet. In an overlay architecture, end systems participating in a multicast group self-organize into efficient structures for delivering data without requiring any support from the existing network infrastructure. Significant effort focused on validation of the architecture [1], [2], [3], [4], [5], and design of protocols [1], [2], [3], [6], [7], [8], [9], [10], [11], [12], [13], [14]. The initial success has led to validations through real deployments and extensive usage of these systems in real applications [15], [16], [17].

Achieving further usage of these systems in a wider range of applications requires integrating security mechanisms to enable confidentiality and integrity of data delivery. Such security mechanisms can be efficiently provided by using symmetric-key based cryptographic algorithms, which in turn require all participants to share a secret key. This key is referred to as the *group key* and protocols providing its establishment and management are referred 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 and focused on reducing the encryption overhead at the source. In contrast to these works, we investigate challenges and opportunities for group key distribution and management in the context of overlay networks. 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 group key distribution.

In this paper, we focus on single-source broadcasting applications. Our work is conducted in the context of an operational overlay broadcasting system, the End System Multicast (ESM) and its data dissemination algorithms [15]. We selected ESM because it is one of the first operationally deployed systems and has seen significant real-world deployment. We have implemented the LKH [18] key management algorithm and its batching variant [19], [20] in ESM. We chose the LKH protocol and its variant due to their wide use in the research community.

Our contribution:

• We conduct a systematic performance evaluation of strategies for key dissemination in the context of an operational overlay broadcasting system on the Planetlab testbed using real traces of join/leave dynamics. While a few recent works have considered issues with key dissemination using overlays [21], [22], these works rely on analysis or simulations with synthetic workloads and do not consider issues such as resilient key delivery. To our knowledge, this paper presents the most extensive study of key dissemination schemes in an overlay context, and the first to involve actual implementation, real-world performance, and real traces. Real Internet deployment in turn allows us to provide unique insights into the interaction between key management and data traffic, as well as investigate the sensitivity of such protocols to losses in realistic Internet data and traffic conditions.

• We conduct the first study of issues in resilient key dissemination in an overlay context. While reliable key dissemination is a challenging and well-studied problem in the context of IP Multicast, we show that it can be considerably simplified with overlays. In particular, overlays provide the unique opportunity to employ protocols for per-hop reliability in the key dissemination structure. We show that leveraging TCP in each hop of the overlay dissemination structure can significantly simplify reliable key dissemination, and could help achieve resiliency in end-to-end delivery. The performance can be further enhanced if convergence properties of overlays are considered.

• We study the design space for dissemination of data and keys. We first consider *coupled architectures* in which the same dissemination structure is used for both data and keys. In particular, we consider (i) a *coupled architecture optimized for data* which intuitively has the lowest level of complexity;

and (ii) a *coupled architecture optimized for keys* which was shown in recent work [21] to result in savings in overhead associated with key dissemination. Our results show that while (i) incurs a high overhead for key dissemination, (ii) violates the physical access bandwidth constraints of nodes for bandwidth-demanding applications. We then consider a *decoupled architecture* using two specialized dissemination structures, one for data and one for keys. The architecture honors access bandwidth constraints at nodes, and our results show the benefits of reducing overheads associated with key dissemination outweigh the cost of maintaining an additional overlay structure.

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

II. System and Adversary Model

A. System Model

We focus on overlay networks providing support for single-source broadcasting applications, that are high-bandwidth (hundreds of kilobits per second), and real-time but not interactive. Such applications can tolerate modest delays of a few seconds through buffering. The system consists of a set of nodes and a data source node communicating via unicast links. All nodes but the source have similar functionality. The nodes are not only receivers of data, but also contribute to the routing process. The source is assumed to be continually available. Direct communication may exist between the source and every receiver (or every pair of nodes).

The overlay construction is completely self-organized and distributed. Each node maintains a set of neighbors referred to as *peers*, a routing table and the upstream node forwarding the data, referred to as the node's *parent*. The neighbor set is bootstrapped at join time by contacting the source and is continually updated via a group management protocol to reflect a set of nodes that are currently reachable in the overlay. The routing table represents a set of nodes that the node is responsible for routing data to, referred to as *children*. The size of this set is limited by the *saturation degree* of the node. This represents the maximum number of concurrent data streams a node is able to support before saturating its physical out-going access link. The saturation degree may vary across nodes representing nodes with heterogeneous access bandwidth (e.g. DSL, Ethernet), and it is critical the node bandwidth constraints are honored. Each node runs an overlay optimization protocol to adapt to abnormal scenarios such as parent failures. In this paper, we assume nodes in the group form a *tree* structure, where the source of the broadcasting application is the root of the tree. We believe the results in the paper may be easily extended to richer structures used for delivering data.

B. Adversary Model

Our focus is ensuring that only authorized group members will have access to *group data*, traffic generated by the application and broadcasted by the source. Overlay networks also disseminate control traffic such as messages generated by the group management and the overlay optimization protocols. The protection of the control traffic is an important problem, but it is out of the scope of this paper. 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. 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 get access to the data. These properties are known as forward and backward secrecy. We assume that the source of the broadcast is trusted to behave correctly and so are the group members. The source and group members are trusted not to 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 interested in the broadcast. In addition there exist means that allow the source and each host part of the group to share a pair-wise key.

III. Design Space

The primary focus of the paper is to systematically study strategies for key dissemination when incorporating confidentiality in an overlay broadcasting system. Our studies employ well-known algorithms for key management which we discuss in Section III-A. The first part of our study considers schemes where key management algorithms are incorporated with minimal changes to the overlay system, by simply disseminating keys using the existent overlay data delivery structure. Even with such a minimalist approach, it is critical to ensure resilient key dissemination – losing a key impacts all data encrypted with that key and significantly affects application performance. We present strategies to achieve resilient key dissemination in Section III-B. While the existing overlay data delivery structure could be used for key dissemination, there are potential benefits in constructing additional structures specifically designed for key distribution. We discuss possible schemes in this space in Section III-C.

A. Key Management Algorithms

Key management algorithms can be classified as centralized and contributory. Centralized key management schemes rely on a single entity, referred to as *key server*, to select and distribute the group key. In contrast, contributory schemes compute the group key based on individual contributions from each protocol participant. Given our focus on single source broadcasting applications, we consider centralized key management schemes.

One of the main factors to consider for centralized key management schemes is the load on the key server resulting

from encryptions required when distributing the key. An additional factor is the key refresh mechanism needed in order to preserve security properties such as forward and backward secrecy. Two strategies have been proposed in the literature: refresh the key every time the group changes or refresh it periodically. In the latter approach, known as *batch rekeying* [19] several group changes are accumulated in one key change. As a result, batch rekeying decreases the number of messages and communication rounds 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 on how long a node that has left the group may continue to have access to information it is not authorized to.

We consider the following key management algorithms:

• *Key-Star*: This is a protocol in which 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. The terminology is adopted from [19].

• *Marking*: This scheme is a batching variant of a well-known protocol, LKH [23]. LKH improves over *Key-Star* by using not only pair-wise shared keys with each member, but also subgroup keys when performing a rekey operation. By using sub-group keys to encrypt the new group key, the encryption cost at the source is significantly reduced. The sub-group keys are not known by the members that left, so the approach has similar security properties as in the case when group keys were encrypted using pair-wise keys. 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.

The protocol in [19] and subsequently refined in [20], which we refer to as *Marking*, applies batch rekeying for the LKH algorithm. As several group changes may have occurred during a rekey period, the algorithm specifies how these changes will be applied to modify the key tree. We chose this scheme over LKH, given the benefits of batching in reducing the computation and communication overhead.

B. Resilient Key Dissemination Strategies

The straight-forward approach to integrate key management algorithms in an overlay system is to use the existent overlay data delivery structure to disseminate rekey messages. However, while applications can tolerate losses in data packets, losses in rekey packets can be more severe. Thus, it is necessary to employ explicit mechanisms to enhance resiliency of key delivery. 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.

While reliable key dissemination is a challenging and well-studied problem in the context of IP Multicast, the problem can be considerably simplified with overlays by using reliable transport protocols (e.g. TCP) in each hop of the overlay key dissemination structure. However, per-hop reliability may not suffice to achieving end-to-end resiliency of key delivery, as losses may occur when the overlay is in a transient state.

With this view, we implemented and evaluated several schemes for distributing rekey messages:

• *NaiveUnicast*: The new key is distributed by the source to each receiver individually using a TCP connection. Note that only the keys that the particular receiver needs are included. This algorithm is used as a base-line for comparison.

• *Tree-TCP*, *Tree-UDP*: The overlay multicast tree involved in data dissemination is used for key dissemination. Keys are transmitted using TCP in each hop for the *Tree-TCP* scheme, and using UDP for the *Tree-UDP* scheme.

• *Tree-Unicast*: We introduce this scheme to handle convergence issues with overlays. We provide the motivation and details in Section V-B.1.

C. Key and Data Dissemination Coupling Strategies

Using the existing overlay data delivery structure for key dissemination has the lowest level of complexity. We refer to such a scheme as *Coupled-DataOptimized*. However, with this strategy, the distribution of key messages can be suboptimal and involve higher overhead. For example, Figure 1.a shows an LKH key-tree. K is the group key, K_0 and K_1 are subgroup keys. The keys at the leaves of the tree (square boxes) are pair-wise keys of users $U_{0,0}$, $U_{0,1}$, $U_{1,0}$ and $U_{1,1}$, with the source. If group key K changes, in order to be distributed, it is encrypted with the subgroup keys K_0 and K_1, resulting in messages $\{K\}_{K_0}$ and $\{K\}_{K_1}$. Note that $\{K\}_{K_0}$ is of interest to $U_{0,0}$ and $U_{0,1}$, while $\{K\}_{K_1}$ is of interest to users $U_{1,0}$ and $U_{1,1}$. Figure 1.b shows a possible structure constructed by a *Coupled-DataOptimized* scheme. In this case the source has to send messages $\{K\}_{K_0}$ (grey key) and $\{K\}_{K_1}$ (black key) to all its children even though they may not need the keys. This is because the source does not have complete knowledge of where nodes that require a key are located in the overlay structure.

An alternative approach is to use the same overlay to disseminate data and keys, but to optimize the overlay for key distribution, as proposed in recent work [21]. We refer to such a scheme as *Coupled-KeyOptimized*. In this approach, the key dissemination tree formed by the group members matches the logical key tree built by LKH in order to send keys just to the nodes that may need them. For example, Figure 1.c shows a structure optimized for key dissemination for the key tree in Figure 1.a. The structure is carefully optimized such 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 $\{K\}_{K_0}$ (grey key) is required for users in the subtree rooted at $U_{0,0}$ and message $\{K\}_{K_1}$ (black key) is required for users in the subtree rooted at $U_{1,1}$. Therefore, the source will send $\{K\}_{K_0}$ only to user $U_{0,0}$ and

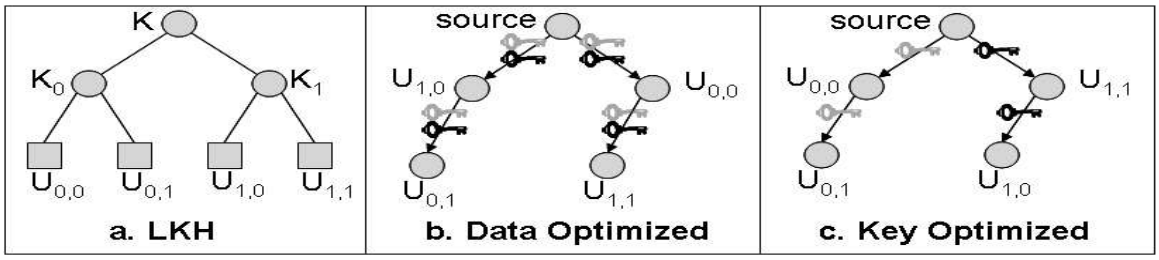

Fig. 1. a) An LKH keys tree. b) An overlay structure optimized for data delivery. Intermediate nodes are positioned by their network characteristics. New keys are sent to all nodes. c) An overlay structure optimized for keys delivery. Intermediate nodes are positioned by their ID. New keys are sent only to nodes that need them.

$\{K\}_{K_1}$ only to user $U_{1,1}$. We refer the reader to [21] for implementation details of the scheme.

While *Coupled-KeyOptimized* may reduce rekeying overhead, we hypothesize that it can significantly impact application performance, and more critically can violate the saturation degree of nodes when bandwidth demanding broadcasting applications are considered. In heterogeneous bandwidth environments that may involve hosts behind DSL and Ethernet, the saturation degree of a node in a data dissemination tree is constrained by the physical outgoing access bandwidth of the node. A scheme like *Coupled-KeyOptimized* does not take into account the physical outgoing access bandwidth limitations. Nodes closer to the source will tend to have a large number of children, irrespective of physical access bandwidth constraints.

To address the problem, we introduce and explore a third strategy, referred to as *Decoupled*, which uses two specialized 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.

IV. Evaluation Goals and Methodology

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

• *Reliable Key Dissemination:* (i) What is the impact of key management algorithms on application performance given that a loss or delay of keys can prevent a host from being able to decrypt data? (ii) How does the choice of mechanism for key transport impact performance? (iii) How effective are protocols for per-hop reliability for key distribution in enabling end-to-end reliability of key dissemination? (iv) What are the additional overheads incurred in terms of computation (encryptions) and communication when mechanisms for reliable key dissemination are introduced?

• *Key and Data Coupling:* (i) What is the impact on application performance with the *Coupled-KeyOptimized* approach? (ii) How significant is the reduction in key dissemination overhead with the *Decoupled* approach? Does this reduction outweigh the additional overheads of maintaining two structures? (iii) How sensitive are the results to the traces and rekey period, and are the benefits significant under real work-loads of interest?

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. The raw throughput, or the throughput a receiver sees in the absence of key management, 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 particular 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 conducted every rekey period.

B. Evaluation Methodology

To answer the questions listed above, we have conducted a detailed evaluation of various key management schemes implemented in an operational 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. Further, 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 the deployment of an operational system based on overlay multicast [15]. This also represents typical media streaming rates in real settings like [24], [25]. We use the outgoing bandwidth information of clients present in the trace, normalize it to the source rate, and

TABLE I
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

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.

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.

Our evaluations have considered a range of rekey periods and studied performance sensitivity to this parameter.

C. Trace Characteristics

Table I summarizes the details of the trace segments used in our evaluations. We used segments of 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 in the period. 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.

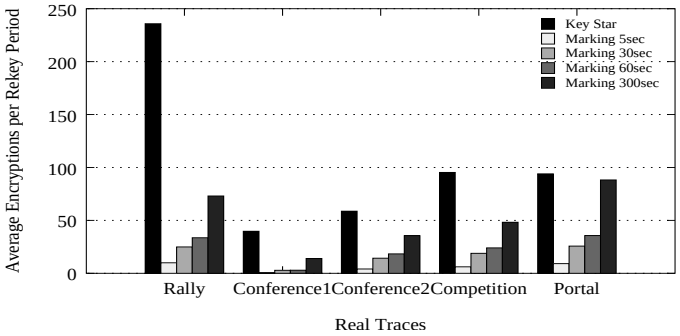

Fig. 2. Avg. encryptions per rekey event with *Marking* for various rekey periods. Each group of bars corresponds to a different trace. The first bar in each group is the average group size.

V. RESULTS

We present results from our evaluations of an overlay multicast system on Planetlab, using the key management algorithms and dissemination structures discussed in Section III. We first discuss the choice of rekey period in Section V-A. Next, we evaluate mechanisms for reliable key dissemination in Section V-B. Finally, in Section V-C, we present results for several strategies for coupling data and key dissemination. 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. Choice of Rekey Period

One important parameter in our experiments is the choice of the rekey period (defined in Section III-A) for the *Marking* algorithm. While a low rekey period results in frequent rekeying and potentially high overhead, the advantages of *Marking* diminish compared to *Key-Star* at higher rekey periods. With the *Marking* scheme with higher rekey periods, the number of encryptions *per rekey event* can be as high as $s * LKH_{deg}$, where s is the number of keys changed during a rekey and LKH_{deg} is the degree of the LKH tree. The number of encryptions required on a rekey operation for *Marking* depends on the dynamics of the trace, the length of the rekey period, and which users leave. In contrast, 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.

Based on the above insight, Figure 2 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*, which is independent of the rekey period and simply the average group size of that trace. The other 4 bars represent the number of encryptions per rekey event for *Marking* for periods of 5, 30, 60 and 300 seconds. For all traces, the number of encryptions per rekey event with *Marking* increases with higher rekey periods. 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.

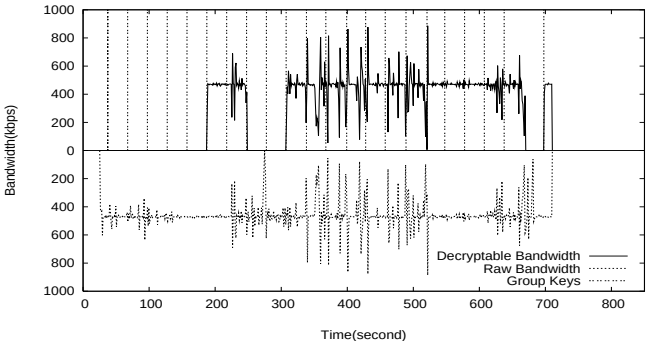

Fig. 3. Impact of loss of rekey packets on application performance

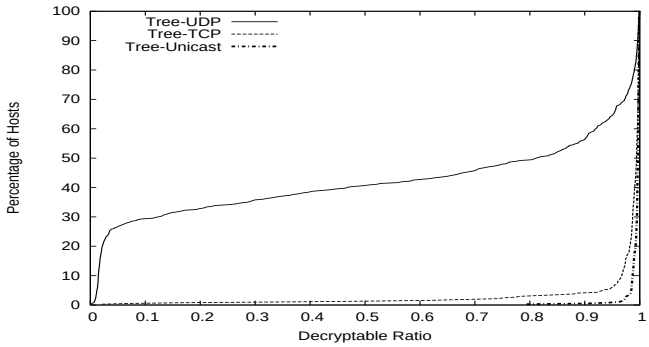

Fig. 4. Impact of key distribution schemes on application performance

Because of these results, the rest of the experiments with *Marking* focus on rekey periods of 60 seconds and smaller.

B. Resilient Key Dissemination Strategies

In this section, we consider different strategies for reliable key dissemination. We first show the impact of key loss on application performance. Next, we show application performance achieved when per-hop unreliable and reliable protocols are used for key dissemination. Then, we consider convergence properties of overlays as a way to further improve end-to-end resiliency of key delivery. Finally, we present the overhead incurred by incorporating resilient key dissemination in an existent overlay broadcasting system.

To appreciate the impact of the loss of rekey packets on application performance, consider Figure 3 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 3, 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

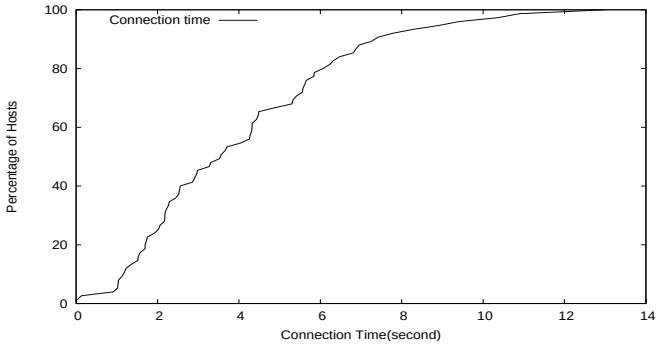

Fig. 5. Connection Times for hosts in the overlay system

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 4 shows the *Decryptable Ratio* achieved with *Tree-TCP*, and *Tree-UDP* if a rekey period of 60 seconds is used. We make two 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-UDP* with *Decryptable Ratio* greater than 0.97 for over 85% of the nodes, indicating that use of TCP for key dissemination can have significant benefits. Figure 4 also shows the *Tree-Unicast* scheme which we discuss next.

1) Overlay Convergence: Figure 4 shows that there is a tail, and some nodes do not perform well. Our analysis reveals the primary reason for the tail is that when a node joins the group, or is disconnected because its parent left the group, it 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 5 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. *Connection Time* is a concern with both node joins and 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.

To address the problem for node joins, we introduce a heuristic called *Tree-Unicast*. The approach is similar with the *Tree-TCP* scheme, where the LKH keys that have been modified are disseminated using the overlay tree. In addition, for nodes that have recently joined, the source sends them the keys directly by using unicast. The time for which the source disseminates keys via unicast depends on the *Connection*

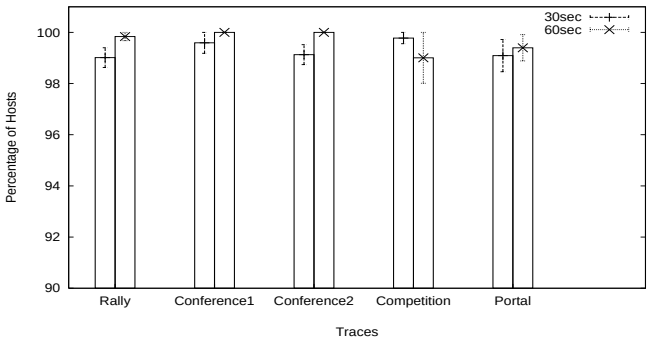

Fig. 6. Performance of *Tree-Unicast* with various traces.

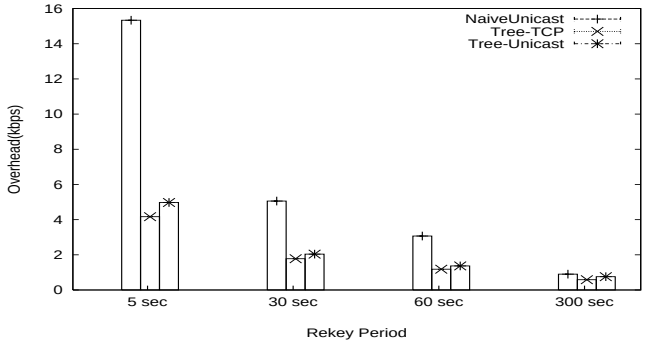

Fig. 7. Avg. overhead due to key management with various schemes and rekey periods

Time of the underlying overlay protocol. Given that typical *Connection Time* values are of the order of $4-10$ seconds, and rekey periods we consider are longer, we simply have the source directly unicast packets to nodes that joined during the last rekey period. Figure 4 shows the *Decryptable Ratio* with *Tree-Unicast*. We note this does achieve better performance than *Tree-TCP* and helps improve the performance of the tail.

Figure 6 considers the performance with *Tree-Unicast* 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.

2) Communication Overhead: While *Tree-Unicast* improves the performance of *Tree-TCP*, it places additional overhead on the source because of keys unicasted to the nodes that join since the last rekey. Figure 7 shows the additional average overhead incurred with key management algorithms by measuring all control traffic related to key management sent or received by the source, averaged across the session duration, for the *Rally* trace. Each group of bars denotes a different rekey period, while the three bars denote *NaiveUnicast, Tree-TCP*, and *Tree-Unicast*.

There are several points to note from Figure 7. First, for all schemes, the overhead due to key management decreases when the rekey period increases. This is expected since rekey operations are conducted less frequently. Second, the benefits of *Tree-TCP* are more visible for lower rekey periods. For example, for a rekey period of 5 seconds, *Tree-TCP* lowers the overhead due to key management from 16 Kbps with *NaiveUnicast* to about 4 Kbps. For rekey periods of 300 seconds, both *NaiveUnicast* and *Tree-TCP* incur overheads of less than 1Kbps. Finally, the additional overhead incurred with *Tree-Unicast* is small and the scheme still has lower overhead than *NaiveUnicast*.

We have repeated the overhead measurements for all traces. Across all traces, while *Tree-Unicast* incurs higher overhead as compared to *Tree-TCP*, the overall overhead due to key management is acceptable and ranges from $2-3$ Kbps for rekey periods of 30 and 60 seconds. We omit the results due to lack of space.

While these results focus on the average overheads, it is important to also consider the peak overheads, and this forms the focus of the next section.

C. Key and Data Dissemination Coupling Strategies

Our evaluations so far have focused on using the existing data delivery structure for constructing overlays. 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-DataOptimized*, *Coupled-KeyOptimized*, and *Decoupled* strategies discussed in Section III-C. Our *Coupled-DataOptimized* scheme simply refers to the *Tree-Unicast* scheme introduced in the previous section – the new term is used for notational convenience. Our implementation of the *Coupled-KeyOptimized* scheme follows the recent proposal in [21], as discussed in Section III-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 begin by evaluating the feasibility of using the *Coupled-KeyOptimized* strategy for delivering data for bandwidth demanding brodcasting applications using simulations. We show the strategy violates the saturation degree and physical bandwidth constraints of nodes. The rest of the section then focuses on comparisons between the *Decoupled* and *Coupled-DataOptimized* strategies.

1) Feasibility of Coupled-KeyOptimized*:* Figure 8 presents results from a simulation study of the *Coupled-KeyOptimized* scheme conducted using the *Rally* trace. Each group of bars correspond to nodes at a particular *forwarding level* in the tree produced by *Coupled-KeyOptimized*. 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-KeyOptimized* structure for nodes at that level; (ii) the average saturation

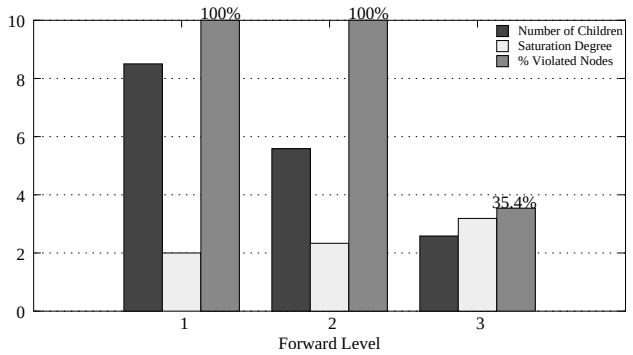

Fig. 8. Number of children imposed by overlay (first bar), saturation degree (second bar) and percentage of nodes with saturation degree violated (third bar - scaled down by a factor of 10), when *Coupled-KeyOptimized* is used to deliver keys.

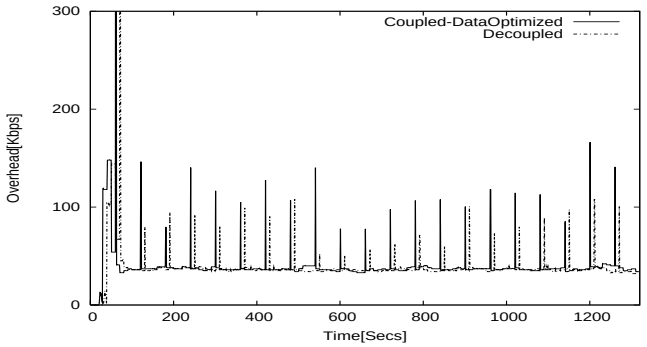

Fig. 9. Overhead per second at the source when mantaining *Coupled-DataOptimized* and *Decoupled* structures (An offset included in X axis to clearly read the graph), for *Rally* trace and rekey period 60 seconds.

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-KeyOptimized* strategy for bandwidth-demanding applications. The reason is that the goal of matching the dissemination tree with the logical key tree built by LKH is at odds with the goal of honoring the heterogeneous access bandwidth constraints of participating nodes.

2) Benefits of Decoupled*:* Given the feasibility concerns with *Coupled-KeyOptimized*, the rest of the section focuses on *Decoupled* and *Coupled-DataOptimized* 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 9 shows the overhead for the two schemes 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

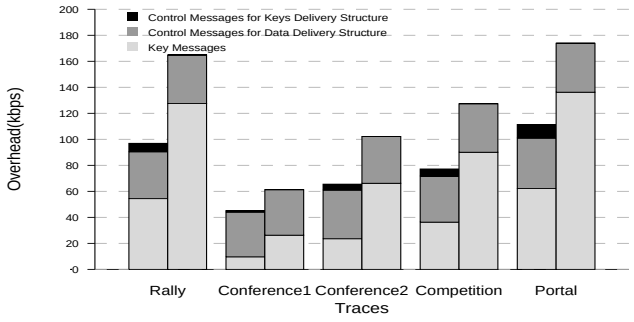

Fig. 10. Peak overhead of *Decoupled* (first bar) and *Coupled-DataOptimized* (second bar) with various traces for rekey period 60 seconds.

structures. We have added an offset on the X-Axis to *Decoupled* curve for clarity of reading this figure. Both curves see periodic spikes corresponding to rekey events. Both schemes have similar overheads in the time ranges between rekey events, but the overhead at the rekey event is reduced for *Decoupled*.

Figure 10 studies the schemes across the complete set of traces. The figure shows the *peak* communication overhead incurred with *Decoupled* and *Coupled-DataOptimized* 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 the peak overhead with *Coupled-DataOptimized*. 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* incured with *Decoupled* is reduced by between 50% to 67% of that incurred with *Coupled-DataOptimized*. 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 between 20% and 50% of that incurred with *Coupled-DataOptimized*. Here 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 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-DataOptimized* 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.

VI. Summary and Conclusions

While key management algorithms have been widely explored in the past, most prior work 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. Specifically:

• We present the first study of key dissemination schemes with overlays that involves implementation, performance evaluation in real Internet environments, and which uses real traces of join/leave dynamics. We show results from key management schemes (*Key-Star* and *Marking*) and key distribution strategies (*NaiveUnicast*, *Tree-TCP*, *Tree-UDP*, and *Tree-Unicast*), deployed on the Planetlab testbed and evaluated with real join/leave dynamics from previous operational deployments (Table I). Prior work in this space [21], [22], has relied on analysis or simulations with synthetic workloads.

• We conduct the first study of resilient key dissemination using overlays. While reliable key dissemination has proven challenging in the context of IP Multicast, we show that it can be significantly simplified in the overlay context through use of TCP to ensure per-hop reliability. For the *Rally* trace and a rekey period of 60 seconds, 90% of receivers see a *Decryptable Ratio* greater than 0.99 when using per-hop TCP. Per-hop reliability is by itself insufficient to ensure end-to-end reliability due to transient conditions that may occur in the overlay. We show that it is feasible to improve performance if the convergence properties of overlays are considered and propose *Tree-Unicast*. We observe that for the *Rally* trace and 60 second rekey periods, with *Tree-Unicast* 99% of receivers see a *Decryptable Ratio* greater than 0.99 as compared to 90% of receivers for the *Tree-TCP* scheme. A detailed study of sensitivity to several traces and rekey periods support these results.

• We study the potential of a decoupled architecture that uses two specialized dissemination structures, one for data and one for keys, compared to coupled architectures in which the same structure is used for disseminating both data and keys. We show that *Coupled-DataOptimized* incurs high peak overheads associated with key dissemination, and *Coupled-KeyOptimized* (recently proposed in [21]), violates the physical access bandwidth constraints of nodes for bandwidth-demanding broadcasting applications. 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, *Decoupled* reduces the peak overhead 44% in comparison to *Coupled-DataOptimized* while the average overheads are comparable.

Our future work involves extending our results to deployments with real users. In addition, we will explore ways to minimize peak overheads, and reduce the costs of incorporating confidentiality in extremely large-scale overlay multicast groups.

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

[19] 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.

[20] 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.

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

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

[23] 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.

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

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