Spatial Clustering for IP Multicast: Algorithms and an Application

Suman Banerjee, Samrat Bhattacharjee
Systems and Networking Research Group
Department of Computer Science
University of Maryland, College Park, MD 20742
{suman,bobby}@cs.umd.edu

Abstract

We introduce spatial clustering of multicast group members as a mechanism to scale wide-area multicast-based applications. To motivate the use of such structures, we present a clustering based re-keying scheme for secure multicast. Using amortized analysis, we show that the communication, processing, and storage costs for this scheme to distribute keys upon membership changes is of constant order. This improves upon the previously known best-case logarithmic bounds under the same assumptions.

Next, we develop a clustering algorithm and a cluster formation protocol that can be built atop IP multicast to create clusters with properties required to implement the secure multicast scheme. To show the viability of such a clustering scheme on the Internet, we present results from implementing this clustering technique on a Internet map of over 280,000 IP routers. We describe how a small set (~ 64) of multicast addresses can be used to efficiently implement intra-cluster communication in large groups (> 64K members). Finally, we present results from packet-level simulations of the clustering protocol to demonstrate protocol robustness under varying membership dynamics.

1 Introduction

The use of multicast has enabled large-scale wide-area applications on the Internet. In large multicast groups senders cannot scalably maintain state for every receiver since this leads to the well-known "feedback implosion" problem [15]. Additional receiver co-ordination, aggregation, and feedback control mechanisms are required to maintain large groups. The prototypical example where such state maintenance is necessary is reliable multicast. Reliable multicast solutions that handle per-receiver state (e.g. retransmission request for lost packets) at senders do not scale; hence, distributing this state maintenance to other entities, receivers [10, 21, 4] or "repair servers" [15] is important in creating a scalable framework.

Clustering techniques have traditionally been used to provide scalable solutions in many systems: adaptive clustering solutions are being currently defined to provide scalable routing solutions for both static networks [23, 8], and for multi-hop mobile networks [20, 25]. A clustering-based architecture has been proposed to generate dynamic distance maps of the Internet [27] and to provide aggregation of receivers with similar interests and capabilities [29].

In this paper, we introduce topology-based member clustering as a mechanism to impose a distributed structure on multicast groups. Our clustering scheme performs a *spatial* partitioning of the multicast topology to create local aggregation of state at distributed points in the network. To motivate such structures, we will apply spatial clustering to the problem of distributing keys for secure multicast and derive a *constant* overhead re-keying mechanism.

1.1 Secure Multicast Key Distribution

Secure group communication systems provide mechanisms for multi-party applications to communicate securely over an untrusted network. In order to send secure messages over an insecure channel, the message has to be encrypted. When such a message is sent to a group, all members of the group must be able to decrypt the message. The most efficient form of such secure group communication requires all members to possess the same "group key"; otherwise, en-route re-keying is required for every packet. However, for each group membership change, this group key needs to be replaced by a new group key. The centralized solution of individually exchanging the new key with each member does not scale to large groups. We will present an efficient clustering-based scheme for group re-keying. Each re-key in our scheme has constant amortized processing, message, and space overheads and this is better the previously known-schemes [28, 6] all of which impose logarithmic overhead. To the best of our knowledge, the clustering-based algorithm described here is the first constant time re-keying scheme when the group has to be re-keyed after every membership change.

1.2 Labeling and Clustering

We decompose the problem of cluster formation into three-layers. Instead of exposing the vagaries of the underlying topology to the clustering protocol, we layer the clustering protocol atop a *labeling* protocol. Given any arbitrary multicast distribution scheme (such as IP-multicast), the labeling protocol is responsible for defining an ancestral relationship among the multicast group members (See Figure 1). Thus, the labeling protocol is used to define an overlay tree containing only the multicast group members over which the clustering protocol will operate.

Note that the ancestral relationship on the multicast tree can be made application-specific. For example, for a reliable multicast application, a member A can be an ancestor of another member B iff A is on the path from the source towards B, and has a higher bandwidth available from the source than B. In this case, the nearest ancestor would be a good candidate to be a re-transmission server [21].

The clustering protocol operates on the overlay tree defined by the labeling protocol. Each cluster is a connected component of the multicast tree and contains a distinguished node — the cluster-leader. Usually the cluster-leader will be responsible for cluster-maintainance. The clustering protocol provides applications the ability to send messages to members in specific clusters and notifies the application of changes in cluster membership. The complete three-layer decomposition is shown in Figure 1.

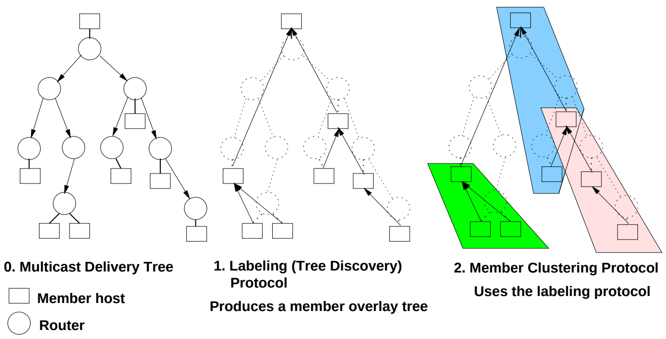


Figure 1: Three-layer decomposition of clustering

The main contributions of this work are:

- A protocol for creating configurable clusters using IP-multicast;

- Derivation of a constant cost algorithm for secure key-distribution using clustering;
- Analysis of the topological properties of clusters on Internet-like topologies to show the viability of spatial clustering;
- Simulation of the clustering protocol to demonstrate its convergence properties.

1.3 Roadmap

In the next section, we present our clustering-based multicast key distribution scheme and derive the constant overhead bound. The scheme (and the description) will assume the existence of a clustering protocol with specific properties. In Section 3, we describe how to implement the underlying labeling protocol within IP multicast. We present the basic ideas of the spatial clustering algorithm in Section 4. Next we present results from simulating this algorithm on Internet-like topologies in order to validate the feasibility of such clustering on the Internet. The final clustering protocol is presented in Section 6. In Section 7, we present a set of packet-level simulation results that demonstrate the robustness and convergence properties of the clustering protocol. We discuss related work in Section 8 and conclude in Section 9.

2 Secure Multicast using Clustering

In this section, we present a secure key distribution scheme for multicast applications that has a *constant* amortized cost overhead, to re-key over the entire group. The scheme assumes the existence of a receiver clustering protocol, which given a set of members distributed in a multicast delivery tree, maintains a set of clusters with the following properties:

- Each member belongs to exactly one cluster.
- Each cluster is a connected sub-graph of the multicast delivery tree.
- Each cluster has k members, for some fixed k, [1]
- Each cluster has a member chosen to be the cluster leader.

Using this underlying clustering protocol, the key distribution mechanism creates a hierarchy of clusters, as shown in Figure 2. A "layer" comprises of a set of members of the

[1] The size of a cluster does not need to be exactly k, but must be within some constant factor of k.

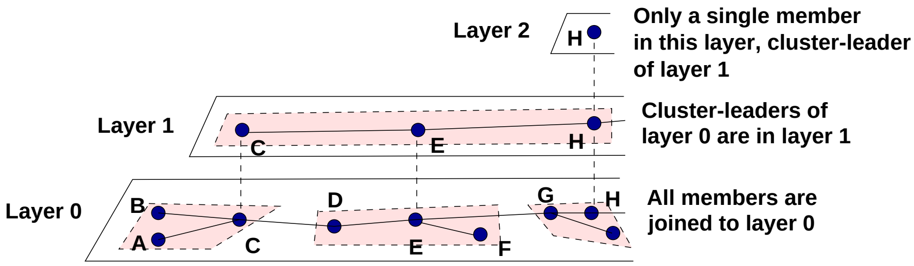


Figure 2: A three layer hierarchy on the member overlay tree for key distribution in secure multicast

secure multicast group in the same level of the hierarchy. Layers are numbered sequentially with the lowest layer of the hierarchy being layer zero (L_0). An instance of the clustering protocol is executed at each layer of the hierarchy to create a set of clusters. All members of the secure multicast group are part of the lowest layer (L_0). Only the cluster leaders of the clusters in layer L_i are part of layer L_{i+1}.

Let us assume that the number of members (N) in the multicast group equals k^R for some integer R. The following properties hold:

- There are exactly $R+1$ layers, where $R = \log_k(N)$. For all j in $[0, R]$, L_j has $N/(k^j)$ members. Specifically, L_0 contains all the members, and the highest layer, L_R, has only a
- If a member u, is present in L_j, it is present in all lower layers $L_0 \ldots L_{j-1}$.
- If a member u is not present in layer L_j, it is not present in any of the higher layers, $L_{j+1} \ldots L_R$.

2.1 Layer Keys and Cluster Keys

A secret *layer-key* is associated with each layer of the hierarchy. A group member possesses a layer key for a specific layer if and only if it is a member of a cluster in that layer. Similarly, a secret *cluster-key* is also associated with each cluster. Once again, a group member possesses a cluster key for a specific cluster if and only if it is a member of that cluster.

2.2 Communication on the Secure Multicast Group

Since all members belong to L_0, the key associated with L_0 is used to communicate within the secure multicast group. The encrypting and decrypting cost for sending a message to the entire group is always $O(1)$; there is no re-keying required at any node for data messages. As described next, the hierarchy of layers and the different clusters are used to efficiently re-key the group upon member joins or leaves.

2.3 Key Distribution Protocol

The key distribution protocol ensures that *at any instant, the layer key of each layer is only available to members joined to that layer.* The membership in layer, L_j, can change if :

- **A member joins layer L_j :** A member joins layer $L_j, j > 0$ iff it has been chosen as the leader of some cluster in layer L_{j-1} to which it is joined. A member would join layer L_0 upon joining the secure multicast group.

- **An existing member leaves layer L_j :** A member leaves the layer $L_j, j > 0$ if it ceases to be a cluster leader of any cluster in layer L_{j-1}, and it recursively leaves from all higher layers. When a member leaves the secure multicast group entirely, it would leave all the layers to which it was joined.

Whenever a member leaves (joins) a layer, L_j, a new layer key is required for that layer. The leader of the cluster of layer L_j, which is affected by this leave (join), sets up a new cluster key for the cluster. Assume, for simplicity, that a single designated member, S, generates new layer keys when needed The new layer key for the layer L_j, encrypted by the layer key of L_{j+1}, is multicast by S to all members in layer L_{j+1}. (Recall that the members of layer L_{j+} are the cluster leaders for the clusters in layer L_j.) Subsequently, this new layer key of layer, L_j —encrypted by the latest cluster key of each cluster— is multicast within the clusters by each of the cluster leaders to all the cluster members. These last set of multicast messages *traverse disjoint parts of the delivery tree.*

The key distribution scheme can be described using the following notation.

- Members and Member Sets
  - u, v : Members of the secure multicast group
  - S : The global key server for all the layers.

- L_i : Layer i in the hierarchy
- $C(u,j)$: Cluster of layer L_j, to which u belongs.
- $Ldr(u,j)$: Leader of the cluster in layer L_j to which u belongs.

- Keys and Messages
  - $\chi_G(t)$: The secret key of G at time t, where where G is a set of members. If G is a cluster, then this is the cluster key, if G is a layer, then this is the layer key. If G is a pair of members, then this is a key shared only by these two members.
  - $\{m\}_e$: Message m is encrypted by the key e.
  - $\langle Unicast :: u \rightarrow v : x\rangle$: u sends a unicast message x to v.
  - $\langle Multicast :: u \rightarrow G : x\rangle$: u multicasts message x to set of members G. Here, G can be either a cluster or a layer. Hence a message of the form $\langle Multcast :: u \rightarrow L_1 : x\rangle$ means u multicasts message x to all members in layer L_1.

When a member u joins or leaves layer L_j, the following operations are performed distributedly by the key distribution protocol :

1. Cluster re-key : $Ldr(u,j)$ generates a new cluster key $\chi_{C(u,j)}(t+1)$ and unicasts it to each current member of the cluster $C(u,j)$ encrypted separately by the pair-wise key of the leader with each member.

$$\forall v \in C(u,j), \langle Unicast :: Ldr(u,j) \rightarrow v : \{\chi_{C(u,j)}(t+1)\}_{\chi_{\{Ldr(u,j),v\}}}\rangle$$

   Obviously, in case u is leaving this cluster, this message is not sent to u. The total communication overhead of this cluster re-key is $O(k)$ per link.

2. Layer re-key : The global key server S generates a new layer key for layer L_j, and multicasts it to all the cluster-leaders of the clusters of layer L_j — these are exactly the members of layer L_{j+1}. Each cluster leader of layer L_j then performs a cluster multicast to all the members of its cluster in layer L_j. The multicast messages are encrypted by the appropriate keys.

$$\langle Multicast :: S \rightarrow L_{j+1} : \{\chi_{L_j}(t+1)\}\chi_{L_{j+1}}(t)\rangle$$

$$\forall v \in L_{j+1}, \langle Multicast :: v \rightarrow C(v,j) : \{\chi_{L_j}(t+1)\}_{\chi_{C(v,j)}(t+1)}\rangle$$

The multicast message from S to layer L_{j+1} traverses each link only once. The cluster multicasts of the cluster-leaders of layer L_j traverse spatially disjoint parts of the multicast delivery tree. Hence, only one such multicast message traverses a link in the tree. Thus, the combined communication cost of these multicasts is $O(1)$ per link.

2.4 Analysis

The efficiency of the key distribution protocol is analyzed using three different metrics :

- Communication cost per link
- Processing requirements at each member for encryption and decryption
- Key storage

Communication cost : Upon most member joins or leaves, only the lowest layer, L_0 will be affected, leaving all other layers unperturbed. However, in the worst case (when the single member at layer $L_{\log_k N}$ leaves) all the layer keys would need to be changed. In this case, one cluster at each layer needs to be re-keyed along with all the layer keys, making the worst case cost of a join or a leave $O(k \log N)$.

Under the assumption that each member of the group is equally likely to join or leave, the amortized cost for joins and leaves is now shown to be of constant order. For any layer, the number of members in that layer is N/k^j. The cost of a single change (join or leave) at any layer is $O(k+2)$. Without loss of generality, only the communication cost of members leaving the group is considered. When a member which occurs in layer L_j leaves, it leaves from each of the layers $L_0 \ldots L_j$. The communication cost per link for this change is $O((k+2)(j+1))$. Hence, the amortized cost of for a member leaving the multicast group is given by :

$$\frac{1}{N}\sum_{j=0}^{R}\frac{N}{k^j}(2+k)(j+1) \;=\; O(k+\frac{\log_k N}{N}) \text{ where } \; R=\log_k N$$

However, k is a constant and the $\frac{\log N}{N} \rightarrow 0$ asymptotically with increasing N. Hence, the amortized cost of a member leaving (and also for joining) is $O(k)$.

Processing Costs and Key Storage : Similar analysis can be used to show that each member change requires only constant processing overhead (key encryption and decryption) at each of the members. The amortized number of keys stored at a member is also of a constant order.

3 Multicast Tree Labeling Protocol

The labeling protocol defines an parent-child relationship among the different members joined to the multicast delivery tree. This relationship need not be purely based on the point of attachment of the host members on the multicast router tree, but can depend on any application-specific metric.

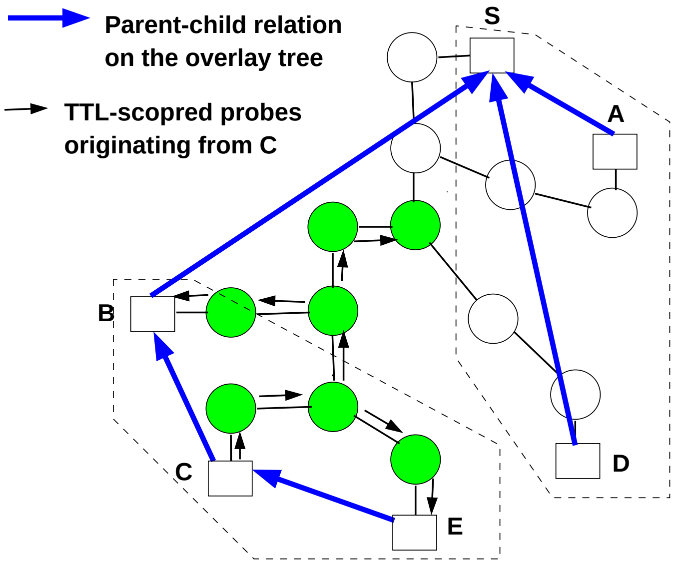


Figure 3: Labeling scheme for secure multicast application

Secure Multicast : For our re-keying scheme in secure multicast, we define this parent-child relationship as follows :
Let $d(u, v)$ denote the distance between the members u and v along the multicast delivery tree in router hops and let member S be the source. A member y is considered to be a parent of member x, iff :

C1: $d(S, y) \leq d(S, x)$. This condition ensures that the parent is farther from the source than the child.

C2 $\forall$ nodes z that satisfy **C1**, $d(y, x) \leq d(z, x)$. This condition chooses a closest member that satisfies condition **C1**.

Since the members are almost always attached to leaf routers, the the path from the source to the parent of a member is not an exact, but an approximate prefix of the path from the source to th child.

The labeling protocol can be implemented on the existing IP multicast infrastructure, by a TTL-scoped expanding ring search mechanism. For example, in Figure 3, member C has currently sent a TTL-scoped multicast probe (with a TTL of five). This probe would reach all the shaded routers in the figure. As a consequence, group members B and E will receive this probe packet. The probe packet from C carries the distance from the source to C. The probe packet at B and E serves two different purposes:

- Node B realizes that it is a possibly parent of C (**C1**). Hence, it would announce its presence to C. This would allow C to conclude that B is its parent, since only B satisfies **C2**.

- Node E also realizes that C is possibly its parent, as it satisfies all the conditions required at this time. Of course, E needs to search for better parent candidates, but now it can limit the Expanding Ring Search to two hops, since it is already aware of C.

Thus, this labeling protocol provides a reduced functionality (commensurate with the needs of the clustering protocol) than other router-intensive multicast tree discovery protocols, like Tracer [19] and STORM [26], that uses `mtrace` to discover all routers on the paths. Obviously, the other protocols can also be used instead of the protocol we have described.

The labeling protocol can be adapted for other applications by suitably customizing the conditions that define that parent-child relationship. To create a preference clustering among receivers of a multicast group [29], we can define a member y to be a parent of x iff y has the closest match in preference to x within a hop-limited domain. Simple lexicographic orderings can be imposed to ensure that the relationship is asymmetric. On the overlay tree so created, a clustering technique, like the one described in this work, can be used to generate all the receiver clusters based on preference proximity.

4 Clustering Algorithm: Basic Idea

In this section, we will describe a simple clustering algorithm that operates on the member overlay tree generated by the labeling protocol to create clusters of members. The algorithm takes a single parameter k — the size of a cluster. However, algorithms that try to create clusters of size exactly k are likely to be unstable, since "good" clusters (of size k) will be disbanded when even one member joins or leaves. Instead, the algorithm we present will

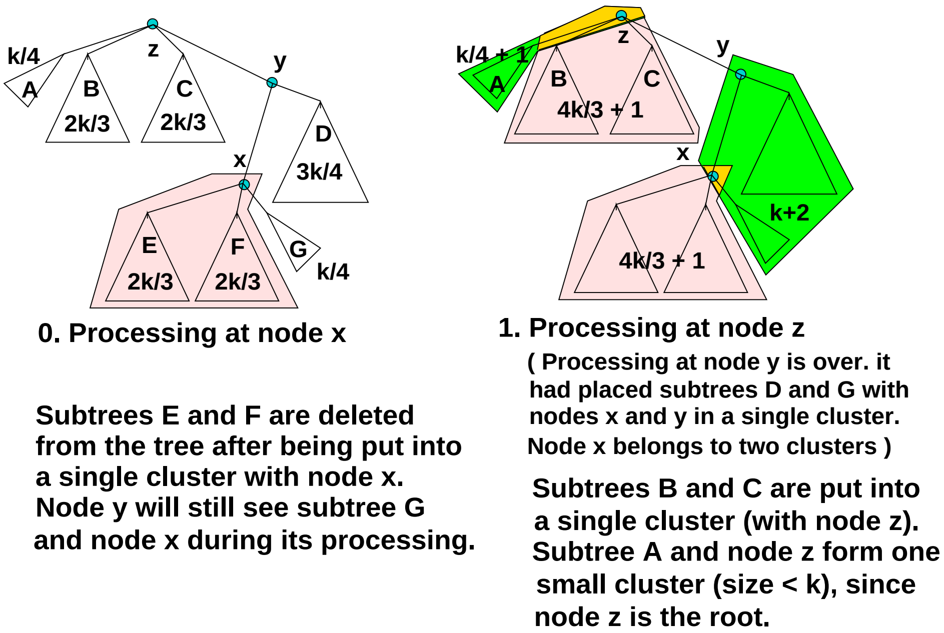


Figure 4: The simple clustering algorithm on a tree

create clusters of size between k and $2k$. For the hierarchical secure multicast scheme, the amoritzed cost of re-keying would still be $O(k)$.

The simple algorithm, though efficient, is centralized, and uses cluster size as the only metric to configure the clusters. However, it will help articulate the final clustering protocol that takes both size and depth of clusters into account.

The clustering algorithm takes an arbitrary tree, T, as an input and outputs a set of clusters, which essentially are tree segments. Let $T(u)$ denote the subtree rooted at node u. The tree is traversed from the leaves, upwards to the root, essentially in post-order. Let u be the first node in the post-order traversal, such that, $|T(u)| \geq k$ and for each child v of u, $|T(v) < k$. Clusters are formed at node u by incrementally grouping the subtrees $T(v)$ till the size bound of k is met. The node u is placed in each of these clusters (to connect the subtrees). There may be one remaining subtree, rooted at a child, x, of u, which could not be put into any cluster ($|T(x)|$ must be $< k$). In this case u and $T(x)$ is retained in the tree, while all the other nodes that have been put into clusters are deleted from the tree. At the end of a single pass of the entire tree, the set of desired clusters is generated.

Consider an example in Figure 4. When node x is being processed, one cluster is created (x, E, F), and the small subtree, G, could not be placed in any cluster is left in the tree, and

Procedure 1 : ClusterTree(T, k)

Returns $ClusterSet$ as the set of clusters

$ClusterSet \leftarrow \perp$
for u in post-order traversal of T **do**
  if $|T(u)| \geq k$ **then**
    $TempClusters \leftarrow$ MergeSubtrees(u, k)
    for $v \in Children(u)$ **do**
      if $\exists C \in TempClusters$ s.t. $v \in C$ **then**
        Delete $T(v)$ from T
      end if
    end for
    if All children of u are deleted **then**
      Delete u from T
    end if
    $Clusters \leftarrow Clusters \cup TempClusters$
  end if
end for

Procedure 2 : MergeSubtrees(u, k)

Returns $TempClusters$ as clusters created

$TempClusters \leftarrow \perp$; $PartialCluster \leftarrow \perp$
for $v \in Children(u)$ **do**
  $PartialCluster \leftarrow PartialCluster \cup T(v)$
  if $|PartialCluster| \geq k - 1$ **then**
    Add $\{u\}$ to $PartialCluster$
    Add $\{PartialCluster\}$ to $TempClusters$
    $PartialCluster \leftarrow \perp$
  end if
end for

is formed into a cluster when processing at node y is done. The entire subtree rooted at y is deleted since all these nodes have been placed into some cluster. Processing at the root, z leaves one small cluster, (z, A), that does not meet the size bound. The exact algorithm is specified in Procedure ClusterTree.

5 Analysis on an Internet Map

In this section, we present the clustering performance of the simple algorithm on an Internet map obtained by the SCAN[2] project. This map contains about 280,000 IP routers discovered using traces on the Internet. The map has been created using the Mercator [11] tool. It uses the source-routing capability of some routers to detect many of the router adjacencies that cannot be detected otherwise from a single point of observation.

The members of a multicast application will usually be located on hosts that are on the "edge" of the Internet. Therefore, for our experiments, we randomly attached member hosts to only leaf routers (i.e. routers that have unit degree on the Internet map). About 50% of all the routers in the map were edge routers. We attached between 10,000 and 500,000 hosts onto these edge routers uniformly at random, for the experiments.

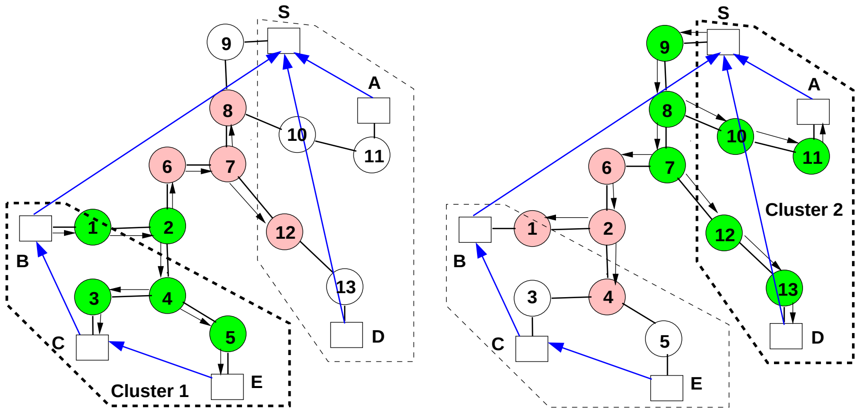
5.1 Member Overlay Tree

To observe the structure of the member overlay tree created using the labeling protocol, we ran a set of experiments. We varied the number of group members between 20,000 and 140,000 in steps of 20,000 and collected results for each size using 100 randomly generated multicast trees. In general, the greater the number of high-degree non-leaf members in the tree, the higher would be the overlap between clusters. As we discuss in our scalable re-keying mechanism for secure multicast, this would be undesirable. We observed that on average, for all these different member sets, about 80% of the non-leaf members in the overlay tree had 6 or less children. As a consequence, we expect to achieve a good clustering of the members, as is demonstrated in the next set of results.

5.2 Assigning Multicast Addresses to Clusters

In each cluster, we assume that only the cluster-leader sends some local multicast traffic to all other group members (as is true in our secure-multicast re-keying scheme). Ideally, we

[2] `http://www.isi.edu/scan`



Spread for Cluster 1 : Routers 1 to 8 and 12. Routers 6, 7, 8, 12 do not lead to members.

Member B is cluster-leader of Cluster 1.

Spread for Cluster 2 : All routers except 3 and 5. Routers 1, 2, 4, 6 do not lead to members.

Member S is cluster-leader of Cluster 2.

Figure 5: “Spill-over” of cluster traffic to routers outside cluster

want this traffic to be localized to within the routers in the cluster topology only. Consider the example in Figure 5. For cluster 1, multicasts from the cluster-leader B, will need a TTL value of 5 to reach the other members. If both the clusters use the same multicast address then all the routers will be joined to the multicast delivery tree. Hence, traffic of cluster 1 will be carried by routers $1--8$ and by router 12. However, the traffic at routers 6 to 8 and 12 do not lead to any members of cluster 1. We denote such routers as “spill-over” routers for cluster 1. If two different multicast addresses are used for the clusters, then there are no spill-over routers for either of the clusters [3] (some routers on the path to the *core* of the multicast address can be considered to be spill-over routers, for core-based architectures [2], [9] and this overhead is unavoidable).

We use two metrics to measure the efficiency of clustering using the simple algorithm:

- For each router, r, in the topology, we record the number of clusters whose traffic reaches r. For a perfect spatial clustering, each router should not handle the local traffic of more than one cluster. However some routers will be required to carry traffic from multiple clusters, when multiple clusters share the same multicast address.

- For each cluster, we record the number of spill-over routers. The number of spill-over

[3] In an architecture that has the capability to perform directed multicast to a subset of members (e.g. AIM [18], GRA [5]), such a facility can be used instead of allocating separate multicast addresses to the different clusters.

routers is zero if each cluster had its own multicast address and increases progressively as more clusters share the same address [4].

In the ideal scenario, each cluster is given a separate multicast address. However, an application with $100,000$ members with clusters of size 10, would then need $10,000$ multicast addresses. More realistically, a fixed number of multicast addresses need to be rationed to the different clusters. We experimented with a few distributed schemes of multicast address allocation to the clusters. The optimal address allocation is equivalent to a graph coloring problem, and is NP-hard. We experimented with a number of address allocation heuristics, and report results from one of them. Each cluster of depth D chooses one address at random from a set of addresses allotted to that partition. The number of addresses alloted to each partition of clusters is weighted by a factor, $d^D \times$ *size of the class*, where d is the average degree of the tree. This scheme is completely distributed with no coordination being required between clusters (as long as the maximum cluster depth is known or can be estimated). We ran a large number of experiments with varying multicast group and cluster sizes. Next, we present results from groups containing $68,000$ members and cluster size between 8 and 16. These results are representative of the other results in the other experiments.

Cluster Traffic at the Routers : In Figure 6 we plot the cumulative distribution of the number of routers for different number of clusters whose traffic pass through the routers (averaged over 100 different trees). When a single multicast address is used, many of the routers have to handle non-local cluster traffic, and more than 40% of routers see traffic from 30 or more clusters. Distributing even a small number of multicast addresses among the clusters is useful in reducing the router overheads. When 64 addresses are used, less than 20% of the routers handle traffic of more than 5 clusters. The ideal (of using one multicast address per cluster or a directed multicast capability in the network) leads to 90% or more routers handling traffic of ≤ 2 clusters.

Spill-over Routers : In Figure 7, we plot the cumulative distribution of the clusters, for the number of spill-over routers for each cluster (again averaged over 100 trees). In the ideal scenario of directed multicast, all cluster traffic is carried usefully. With 256 multicast

[4] Usage of multiple multicast addresses for a single application has previously been proposed to provide out-of-band retransmission of lost packets to a set of members in reliable multicast [16] and to partition heterogeneous receiver sets by their traffic preferences [29]; we will use this technique to limit spill-over multicast traffic.

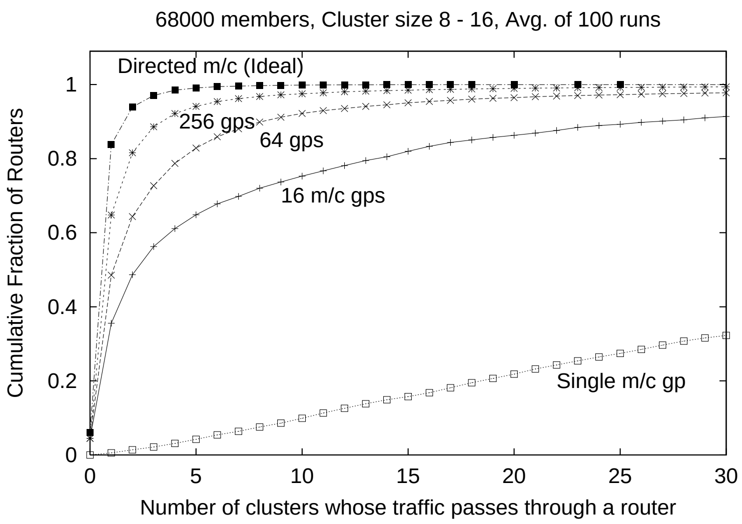


Figure 6: Cumulative distribution of the routers that handle cluster traffic for different number of clusters

addresses used, 80% of the clusters, reach 7 or less spill-over routers. We also experimented with a set of threshold based schemes that use heuristics to estimate this spill-over. Such schemes, in general, are able to dramatically reduce the spill-over router count but use more groups (e.g. using 280 addresses, 90% of the clusters spill over to < 7 routers). However, these schemes require co-ordination between clusters.

6 Final Clustering Protocol

The simple algorithm presented in Section 4, creates a set of clusters of size between k and $2k$. It may leave at most one cluster of size less than k rooted at the root of the tree. The depth of the clusters have not been considered by this algorithm.

For multicast applications with large member sets, the number of clusters will be correspondingly large. When a relatively small number of multicast group addresses are to distributed among these clusters, each address may need to be given to many clusters. In the absence of directed multicast capability, the only way to localize the cluster traffic to within the cluster is to use a hop-limiting mechanism. Hence, clusters with large depth will need to use a higher hop limit for the traffic to reach all the members. This will cause the traffic of these large clusters to extend to many other routers outside the cluster. Therefore,

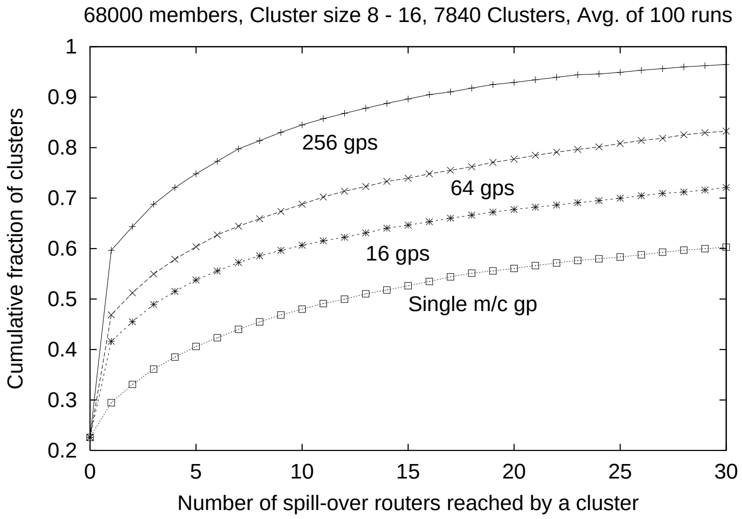


Figure 7: Cumulative distribution of the clusters for the number of spill-over routers with different number of multicast addresses allocated

it is important to bound both cluster size and cluster depth. We introduce a cost model for the clusters that depends on both the size and the depth of the clusters.

6.1 A Cost Structure for Clusters

In order to extend the notion of desirable clusters to be clusters that meet defined size and depth bounds, we need an extra parameter, D, which is the maximum acceptable depth of a "good" cluster. We define a function that assigns a cost to each cluster with the following property: *Clusters of size between k and $2k$ and of depth less than D are assigned "low" costs, while clusters that have size outside the k–$2k$ range or have higher depth will be assigned higher but configurable costs.* Further, as the size of a cluster approaches the range k–$2k$, its cost decreases; similarly, the cost of a cluster increases at in increases in proportion to its depth at a slow rate, up to the parameter, D, and increases at a much higher rate whenever the cluster depth crosses D.

Thus, the cost of a cluster has two components — cost due to size and cost due to depth. In our model, the final cost of the cluster is just a sum of the cost due to size and the cost due to depth. We model cluster cost as follows:

- The cost component, due to size of a cluster, is zero if the size is between k and $2k$

	$d \leq D$	$d > D$
$s < k$	$\alpha_L d + \beta(k - s)$	$\alpha_L D + \alpha_H(d - D) + \beta(k - s)$
$k \leq s < 2k$	$\alpha_L d$	$\alpha_L D + \alpha_H(d - D)$
$s \geq 2k$	$\alpha_L d \quad + \quad \beta(s - 2k + 1)$	$\alpha_L D + \alpha_H(d - D) \quad + \quad \beta(s - 2k + 1)$

Table 1: Cost of cluster of size s and depth d

and increases linearly (with slope β) as the size of the cluster decreases below k or increases beyond $2k$.

- The cost of a cluster, due to its depth, d increases slowly with increase in depth (α_L), up to the chosen parameter, D, and increases at a higher rate (α_H) beyond that. Obviously, α_L should be less than α_H.

Using this model, the cost of a cluster of size s and depth d can be expressed of these two parameters shown in Table 1. The purely size-based clustering scheme can exactly be mapped into this model by choosing appropriate constants in the function.

To control the stability of the protocol we use two cost thresholds: a low watermark, $\mathcal{L}$, and a high watermark, $\mathcal{H}$. Cluster of cost less than or equal to $\mathcal{L}$ are considered stable. Hence, $\mathcal{L}$ is set to the maximum cost of a cluster that meets both the size and depth bounds. The gap between the two watermarks are used to dampen frequent reclustering due to small changes in the cost for every join and leave by members using the multicast application. Only when the cost of a cluster increases beyond $\mathcal{H}$ does it attempt to re-cluster.

6.2 Cost-Based Clustering Protocol

The cost-based clustering protocol is a distributed implementation of the simple size-based clustering algorithm, with the newly defined costs replacing the size metric The labeling protocol layered below this clustering protocol also tracks the subtree size and the subtree depth at each member by aggregating this information from the periodic messages received from each child member in the tree.

The distributed implementation proceeds as follows :

1. Each member initially creates a trivial cluster, with itself as the only member, when it joins the tree.

2. Each member attempts to add its children on the tree to its current cluster, but only if by doing so, the new cost of the cluster is reduced (the cluster cost may go up, if the cluster depth increases beyond the depth parameter, D). In this way, the initial clusters start growing.

3. Whenever a cluster becomes "stable" i.e. its cost is below $\mathcal{L}$, its *cluster-root*, (the member in the cluster closest to the root) does not include these members in its subtree size report to its parent. This is the equivalent of subtree deletion in the simple size-based algorithm.

4. Through member aditions and deletions, if the cost of a once-stable cluster exceeds, $\mathcal{H}$, the current cluster-root of the cluster, re-starts reporting the presence of the cluster members in its subtree size and depth information to its parent. Hence, all these members can be incrementally re-clustered by step 2.

7 Simulation Results

In this section, we present simulation results from our packet-level simulation of the cost-based clustering protocol on the `ns` (version 2) simulator. Due to the high processing and memory demands of `ns`, we were limited to simulations on topologies with upto 1000 nodes.

We added a new agent object to perform the task of the clustering and labeling protocols in `ns`. All clustering related messages are piggybacked on periodic labeling protocol messages. During membership changes, the labeling protocol can would temporarily mislabel the parent-child relationship of the members; in these cases clustering protocol would also be subject to errors. These conditions are modeled in our simulations.

7.1 Experimental Methodology

The simulation experiments using `ns` were run on different multicast tree topologies. We used randomly generated multicast trees, from graphs with the same average degree as the SCAN Internet topology map. The labeling protocol is implemented using TTL-scoped Expanding Ring Search messages that serve as periodic probes to discover the member topology and changes to the topology. The time taken by the labeling protocol to discover this topology is directly related to the heart-beat period of the periodic probes. Since all clustering messages are piggybacked over these labeling messages, the convergence time

		Total Num. of Clusters	Num. of Low Cost Cl.	Avg. Cl. Size	Avg. Cl. Depth	Avg. (Max) Stab. Time (s)
$D = 6$	$k = 4$	47	43	5.8	3.6	11.7 (22.9)
	$k = 10$	19	16	13.5	4.4	14.1 (25.0)
	$k = 20$	11	6	22.9	4.9	14.2 (27.3)
$D = 12$	$k = 4$	50	41	5.5	3.3	11.6 (21.1)
	$k = 10$	17	16	15.0	4.5	14.7 (31.0)
	$k = 20$	9	8	28.2	5.8	17.8 (36.9)

Table 2: Cluster characteristics as k and D change.

for the clustering protocol is also directly related to this heartbeat period. For all the experiments reported in this paper, we use the labeling heartbeat period fixed at 2 seconds.

7.2 Performance Metrics and Results

We use the following metrics to evaluate the clustering:

- **Size and depth of clusters:** This metric shows how well the clustering algorithm can be used to create clusters that meet the required size and depth bounds.
- **Time to stabilize:** We measure the amount of time taken for all the clusters to converge after membership changes. Note that the actual time value is directly related to the heartbeat period since this is the minimum granularity at which the clustering protocol can exchange messages.

7.2.1 Creating configurable clusters

A fundamental goal of the clustering protocol is to create configurable receiver clusters of bounded size and depth. We simulated the protocol for values of the size parameter k varying from 3–20 and for the depth parameter varying from 4–12. For this experiment, there were 250 group members in a 500 node graph. A representative subset of the results is presented in Table 2. (The stability time column is discussed in the next section). In our analysis of the results, we noticed the following trends: The algorithm creates clusters with the specified size and depth properties subject to topological constraints. Thus, if the value of k is large and the value of D is small, then the protocol is not able to construct low cost clusters simply because they do not exist — this is the why only about 50% of the clusters

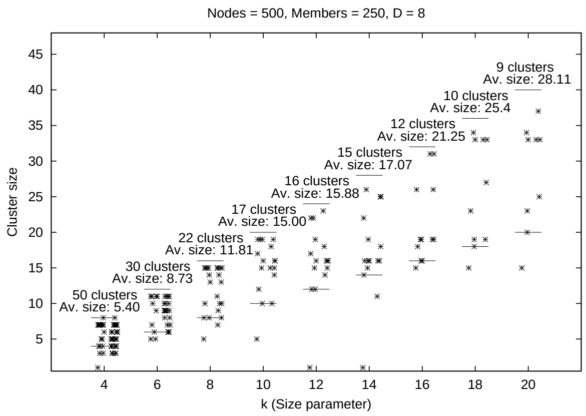


Figure 8: Individual cluster sizes as the size parameter k varies.

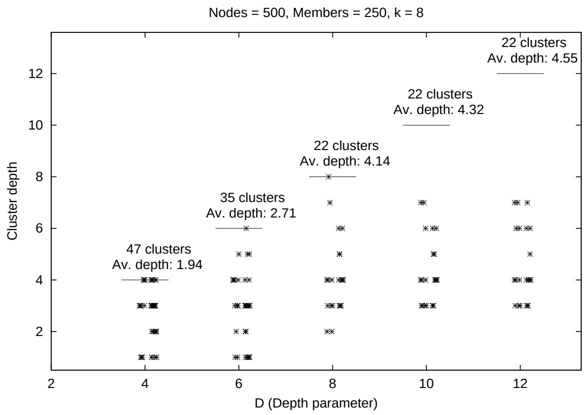


Figure 9: Individual cluster sizes as the depth parameter D varies.

formed for $k = 20$ and $D = 6$ are low cost clusters. Note that when the permitted depth is increased, the fraction of low cost clusters increase as well.

Figures 8 and 9 show details of a single representative run as the size and depth parameters vary. In Figure 8, the maximum depth of the clusters is fixed at 8, and the size parameter is varied from 4–20. Individual cluster sizes for different values of k are plotted in Figure 8. In all cases, the majority of clusters are constrained to the specified size limits; however, for each value of k, there are a few clusters that are always smaller than the requisite size. Further analysis of our data showed that the members in these clusters were located in isolated parts of the topology: the only way to increase these clusters in size, would be by increasing the depth of the clusters, which would have led to increase in the cost of the resulting clusters. Figure 9 shows the corresponding results when the size parameter k is fixed at 8, and the depth parameter D is varied from 4–12. We note that for a fixed, k, as D is increased, the distribution of cluster depths do not increase proportionately over the entire range. This is because once the clusters within requisite size bounds (8–16 in this experiment) are formed, the algorithm does not provide any incentive to expand clusters further and increase depth.

7.2.2 Stability

In this section, we present results on the convergence properties of the algorithm. All the results here are shown using simulated elapsed time; the convergence times depend entirely on the rate at which updates are sent which is fixed at 2 seconds.

Initial Stability

In this experiment, we start with an empty member set and simulate worst case starting conditions by allowing 250 group members to join the group at the time 1 second. We consider a member "stable" when the protocol assigns it to a cluster and the assignment does not change further.

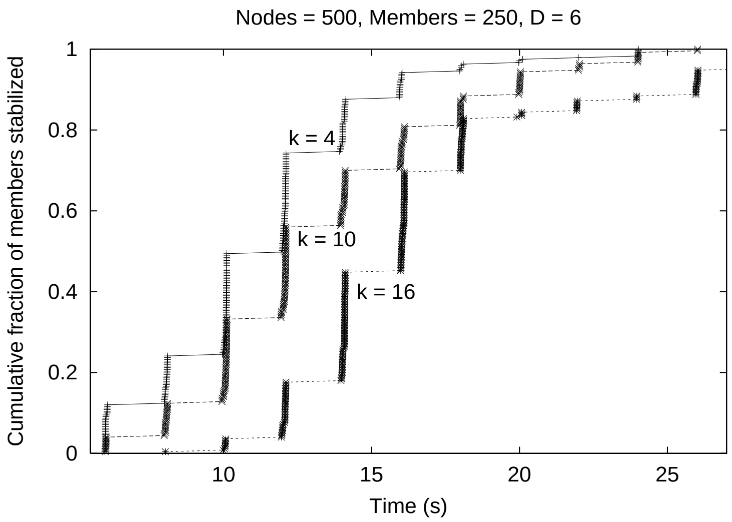


Figure 10: Cumulative cluster convergence times for 250 members joining at once.

Table 2 shows the average and maximum time taken for clusters to stabilize as k and

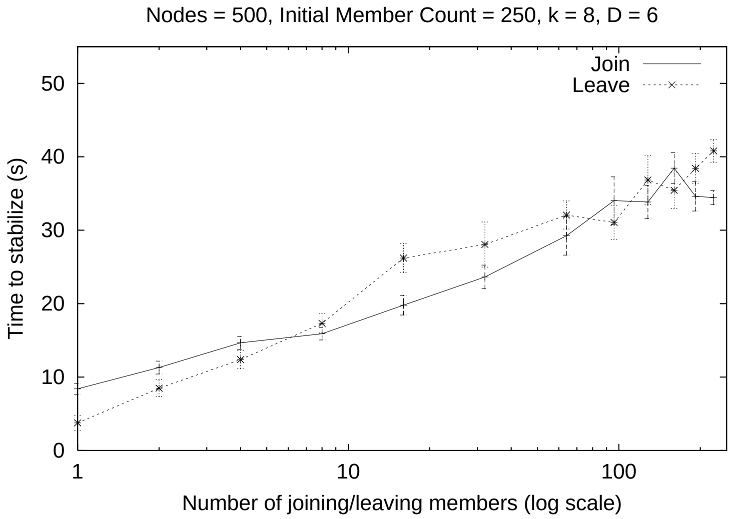


Figure 11: Convergence times as the number of group members joining and leaving is varied.

D changes. As expected, the stabilization times increase with both k and D. Even with all members starting at the same time, the worst case convergence is fairly quick, less than 30 seconds for most cases (with a 2 second heartbeat value). On average, most members converge a lot quicker. This is shown in detail in Figure 10. This figure plots the cumulative fraction of members stabilized over time for representative runs of the simulation. The different curves correspond to different settings of the cluster size parameter with the cluster depth parameter set to 6. Since all members start at the same time and all messages are sent using the same heartbeat period, the curves show a pronounced step function shape. This result shows that for these experiments, though the maximum time to stabilize is around 30 seconds for a few group members, 90% of the group stabilizes within about 20 seconds. It takes longer for larger clusters to stabilize since they must expand to a accommodate a higher minimum number of members.

Member Joins and Leaves

In this experiment, instead of all members joining an empty group at once, we initially allow 250 members to join the multicast group. Once these members stabilize to a set of clusters, we experiment with a sets of different members joining the tree at once. The nodes in the graph at which the members join are chosen uniformly at random. We measure time for the clusters to stabilize after these members join. After the clusters have stabilized (and the join times recorded), the new members are made to leave the group at the same time — once again, we measure the time it takes for the clusters to stabilize. The results of this experiment is shown in Figure 11.

We plot the number of members that join (and subsequently leave) on the x-axis (log scale) the time the clusters take to stabilize on the y-axis. The plot also shows the 95% confidence interval obtained after repeating the experiments between 10–50 times (we had to conduct more experiments with the smaller sets of members joining and leaving to get useful confidence intervals). From Figure 11, we observe that when the membership changes are minor, the time required to re-stabilize the clusters is small as well (once again, the heartbeat values are set at 2 seconds). As expected, the time to stabilize is proportional to the change in the tree. Interestingly, when the membership change is extremely large, the time to stabilize converges to about 40 seconds for the joins, and about 30 seconds for the leaves. This is because the incremental cost of joins and leaves for extra members decrease as more and more members change the multicast membership within a small time period. Thus, the clustering protocol exhibits desired scaling properties as the rate of change in the member size joined to the tree increases.

8 Related Work

Clustering : Cluster decomposition techniques have traditionally been used to scale distributed algorithms and communication networks [24]. Clustering has been recently been used in other contexts as a mechanism to scale multicast applications. Wong et. al. [29] propose a preference clustering scheme for multicast applications, where members with similar preferences (e.g. video quality requirements for a video multicast) are grouped into clusters. Other work by Cheung et al [7] groups receivers of a video multicast from a single source into separate groups, and each group is handled separately by the video server. Our clustering approach differs from the prior work since we balance the size and the depths of the clusters, and minimize overlap between clusters which minimizes interference of intra-cluster traffic. Topology-based clustering is used for routing in mobile ad-hoc networks in the Cluster-Based Routing Protocol (CBRP) [14] to reduce routing information that needs to be flooded through such networks due to frequent changes, and in [17] as a mechanism to self-organize large networks

Secure Multicasting : There have been a set of particularly elegant protocols proposed for secure multicast in recent literature. The first secure multicast protocol that did not require $O(N)$ time for re-keying was proposed by Wong et. al. [28]. They define a tree hierarchy *of keys* distributed between different sets of members. In non-trivial configurations, a *key server* manages $2N$ keys, where N is the size of the group, while group members hold h keys, where h is the depth of the tree. In contrast we define a hierarchy *of the members*. In our approach, in the worst case a member needs to maintain $O(\log N)$ keys, while the average number of keys for a member is a constant dependent on k, the cluster size parameter.

Iolus [22] is an administratively scoped secure multicast scheme. A set of pre-configured members, Group Security Controllers (GSCs), are deployed into the network and perform the task equivalent to our cluster leaders – thus the partitioning of members into sets of clusters is statically dependent on the location of the member, and not on the changing member dynamics of the delivery tree. Hence, Iolus cannot provide the low processing, storage and communication cost that our technique guarantees, by dynamically adapting the clusters to within required size and depth properties.

A boolean function minimization technique has been proposed by Chang et. al. [6]. In this scheme, each member is given a set of keys. The number of keys at a member is $O(\log N)$, where N is an identifier space at least as large as the number of members in the group. The authors show that their scheme uses $O(\log N)$ messages to redistribute the new keys. Further, the incremental cost of re-keying decreases as the membership changes increase. Compared to [6], members in our scheme only maintain an amortized constant number of keys, instead of $O(\log N)$.

In MARKS [3], a low overhead key distribution scheme has been proposed where a key manager reveals a key sequence to each member, at the time it joins. However, the protocol assumes an accurate knowledge of the duration over which a member stays joined to the secure group at the time the member joins (and a key sequence of appropriate length is revealed to that member).

9 Conclusions

We have introduced spatial clustering as a mechanism to scale wide-area multicast applications and applied it to derive the theoretically best known secure multicast key distribution scheme. We have shown the viability of such clustering on Internet-like topologies and developed mechanisms for efficient intra-cluster communication using a small number of multicast groups. Our detailed simulations show that the clustering protocol is robust across a wide range of cluster configurations, group topologies, member densities, and has scalable convergence properties.

References

[1] A. Ballardie. Scalable Multicast Key Distribution. Network Working Group, RFC 1949., May 1996.

[2] Tony Ballardie, Paul Francis, and Jon Crowcroft. Core based trees (CBT). In *Proceedings of SIGCOMM*, pages 85–95, San Francisco, September 1993. ACM.

[3] B. Briscoe. Marks : Zero side effect multicast key management using arbitrarily revealed key sequences. In *1st International Workshop on Networked Group Communication, Pisa, Italy, November 1999.*, Pisa, Italy, November 1999.

[4] G. Parulkar C. Papadopoulos and G. Varghese. An error control scheme for large-scale multicast applications. In *Proceedings of Infocom'98*, 1998.

[5] B. Cain, T. Speakman, and D. Towsley. Generic router assist (gra) building block motivation and architecture. Internet Draft, Internet Engineering Task Force, March 2000. Work in progress.

[6] I. Chang, R. Engel, D. Kandlur, D. Pendarakis, and D. Saha. Key management for secure internet multicast using boolean function minimization techniques. In *Proceddings of Infocom*, New York, March 1999.

[7] S.Y. Cheung, M.H. Ammar, and X. Li. On the use of destination set grouping to improve fairness in multicast video distribution. In *Proceedings of Infocom*, San Fransisco, March 1996.

[8] ATM Forum Technical Committee. Private network network interface, version 1.0, May 1996.

[9] D. Estrin, D. Farinacci, A. Helmy, D. Thaler, S. Deering, M. Handley, V. Jacobson, C. Liu, and P. Sharma. Protocol independent multicast-sparse mode, rfc 2117. Technical report, IETF, 1997.

[10] S. Floyd, V. Jacobson, C.-G. Liu, S. McCanne, and L. Zhang. Reliable multicast framework for light-weight sessions and application level framing. In *Proceedings of SIGCOMM*, Cambridge, Massachusetts, September 1995.

[11] R. Govindan and H. Tangmunarunkit. Heuristics for internet map discovery. In *Proceedings of Infocom*, Tel Aviv, Israel, March 2000.

[12] H. Harney and C. Muckenhirn. Group key management protocol (GKMP) architecture. Request for Comments (Experimental) 2094, Internet Engineering Task Force, July 1997.

[13] S. Jamin, C. Jin, Y. Jin, D. Raz, Y. Shavitt, and L. Zhang. On the placement of internet instrumentation. In *Proceedings of Infocom'00*, Tel Aviv, Israel, March 2000.

[14] M. Jiang, J. Li, and Y. Tay. Cluster based routing Protocol(CBRP). Internet Draft, IETF, August 1999.

[15] S. Kasera, J. Kurose, and D. Towsley. A Comparison of Server-Based and Receiver-Based Local Recovery Approaches for Scalable Reliable Multicast. In *Proceedings of Infocom*, April 1998.

[16] S. Kasera, D. Towsley, and J. Kurose. Scalable reliable multicast using multiple multicast channels. In *Proceedings of the ACM International Conference on Measurement and Modeling of Computer Systems (Sigmetrics '97)*, Seattle, WA, jun 1997.

[17] R. Krishnan, R. Ramanathan, and M. Steenstrup. Optimization algorithms for large self-structuring networks. In *Proceedings of Infocom*, New York, March 1999.

[18] B.N. Levine and J.J. Garcia-Luna-Aceves. Improving internet multicast with routing labels. In *Proc. IEEE International Conference on Network Protocols*, pages 241–50, October 1997.

[19] B.N. Levine, S. Paul, and J.J. Garcia-Luna-Aceves. Organizing multicast receivers deterministically according to packet-loss correlation. In *Proc. Sixth ACM International Multimedia Conference (ACM Multimedia 98)*, September 1998.

[20] C.R. Lin and M. Gerla. Adaptive clustering for mobile, wireless networks. *Journal on Selected Areas of Communication*, 15(7), September 1997.

[21] J.C. Lin and S. Paul. RMTP: a reliable multicast transport protocol. In *Proceedings of INFOCOM*, San Fransisco, California, March 1996.

[22] S. Mittra. Iolus: A framework for scalable secure multicasting. *Computer Communications Review*, 27(4):277–288, October 1997. Proceedings of ACM SIGCOMM'97, Sept. 1997.

[23] J. Moy. Open shortest path first (version 2). *RFC 2178*, 1997.

[24] K. Muralidhar and M. M. Sundareshan. On the decomposition of large communication networks for hierarchical control implementation. *IEEE Transactions on Communications*, COM-34, 10:985–987, 1986.

[25] R. Ramanathan and M. Steenstrup. Hierarchically organzied, multi-hop mobile wireless networks for quality-of-service support. *Mobile Networks and Applications*, 3(1), June 1998.

[26] X. Rex Xu, A.C. Myers, H. Zhang, and R. Yavatkar. Resilient multicast support for continuous-media applications. In *Proceedings of NOSSDAV*, St. Louis, Missouri, May 1997.

[27] W. Theilmann and K. Rothermel. Dynamic Distance Maps of the Internet. In *Proceedings of Infocom*, Tel Aviv, March 2000.

[28] C.K. Wong, M. Gouda, and S. Lam. Secure group communications using key graphs. *Proceedings of SIGCOMM*, 28(4):68–79, September 1998.

[29] T. Wong, R. Katz, and S. McCanne. An Evaluation of Preference Clustering in Large-scale Multicast Applications. In *Proceedings of INFOCOM*, Tel-Aviv, March 2000.