Approximation Algorithms for Key Management in Secure Multicast

Agnes Chan* Rajmohan Rajaraman† Zhifeng Sun‡ Feng Zhu §

February 9, 2009

Abstract

A number of data dissemination systems, such as interactive gaming, stock data distribution, video conferencing, and publish-subscribe systems need to guarantee the privacy and authenticity of the participants. Many such systems rely on symmetric key cryptography, whereby all legitimate group members share a common key for group communication. An important problem in such a system is to maintain the shared group key as the group membership changes. The focus of this paper is on a well-studied key management approach that uses a hierarchy of auxiliary keys to update the shared group key and maintain the desired security properties. In this key hierarchy scheme, a group controller distributes auxiliary keys to subgroups of members; when the group membership changes, an appropriate subset of the keys are updated and multicast to the relevant subgroups.

We consider the problem of determining a key hierarchy that minimizes the average communication cost of an update, given update frequencies of the group members and an edge-weighted undirected graph that captures routing costs. We first consider the objective of minimizing the average number of multicast messages needed for an update (thus ignoring the underlying routing graph), for which we present polynomial-time approximation scheme (PTAS). We next show that when routing costs are considered, the problem is NP-hard even when the underlying routing network is a tree network or even when every group member has the same update frequency. Our main result is a polynomial time constant-factor approximation algorithm for the general case where the routing network is an arbitrary weighted graph and group members have nonuniform update frequencies. We obtain improved constant approximation factors for the special cases where the routing network is a tree and when the update frequencies are uniform.

*College of Computer and Information Science, Northeastern University, Boston, MA 02115, Email: ahchan@ccs.neu.edu
†College of Computer and Information Science, Northeastern University, Boston, MA 02115, Email: rraj@ccs.neu.edu
‡College of Computer and Information Science, Northeastern University, Boston, MA 02115, Email: austin@ccs.neu.edu
§Cisco Systems, San Jose, CA, Email: zhufeng@cisco.com

1 Introduction

A number of data dissemination and publish-subscribe systems, such as interactive gaming, stock data distribution, and video conferencing, need to guarantee the privacy and authenticity of the participants. Many such systems rely on symmetric key cryptography, whereby all legitimate group members share a common key, henceforth referred to as the *group key*, for group communication. An important problem in such a system is to maintain the shared group key as the group membership changes. The main security requirement is *confidentiality*: only valid users should have access to the multicast data. In particular this means that any user should have access to the data only during the time periods that the user is a member of the group.

There have been several proposals for multicast key distribution for the Internet and ad hoc wireless networks [2, 7, 8, 16, 22]. A simple solution proposed in early Internet RFCs is to assign each user a *user key*; when there is a change in the membership, a new group key is selected and separately unicast to each of the users using their respective user keys [8, 7]. A major drawback of such a key management scheme is its prohibitively high update cost in scenarios where member updates are frequent.

The focus of this paper is on a natural key management approach that uses a hierarchy of auxiliary keys to update the shared group key and maintain the desired security properties. Variations of this approach, commonly referred to as the *Key Graph* or the *Logical Key Hierarchy* scheme, were proposed by several independent groups of researchers [2, 4, 19, 21, 22]. The main idea is to have a single group key for data communication, and have a group controller (a special server) distribute auxiliary subgroup keys to the group members according to a key hierarchy. The leaves of the key hierarchy are the group members and every node of the tree (including the leaves) has an associated *auxiliary* key. The key associated with the root is the shared group key. Each member stores auxiliary keys corresponding to all the nodes in the path to the root in the hierarchy. When an update occurs, say at member u, then all the keys along the path from u to the root are rekeyed from the bottom up (that is, new auxiliary keys are selected for every node on the path). If a key at node v is rekeyed, the new key value is multicast to all the members in the subtree rooted at v using the keys associated with the children of v in the hierarchy.[1] A detailed example is given in Figure 1. It is not hard to see that the above key hierarchy approach, suitably implemented, yields an exponential reduction in the number of multicast messages needed on a member update, as compared to the scheme involving one auxiliary key per user.

The effectiveness of a particular key hierarchy depends on several factors including the organization of the members in the hierarchy, the routing costs in the underlying network that connects the members and the group controller, and the frequency with which individual members join or leave the group. Past research has focused on either the security properties of the key hierarchy scheme [3] or concentrated on minimizing either the total number of auxiliary keys updated or the total number of multicast messages [20], not taking into account the routing costs in the underlying communication network.

1.1 Our contributions

In this paper, we consider the problem of designing key hierarchies that minimize the average update cost, given an arbitrary underlying routing network and given arbitrary update frequencies of the members, which we refer henceforth to as weights. Let S denote the set of all group members. For each member v, we are given a weight w_v representing the update probability at v (a join/leave action at v). Let G denote an edge-weighted undirected routing network that connects the group members with a group controller r. The cost of any multicast from r to any subset of S is determined by G. The cost of a given key hierarchy is then given by the weighted average, over the members v, of the sum of the costs of the multicasts performed when an update occurs at v. A formal problem definition is given in Section 2.

[1] We emphasize here that auxiliary keys in the key hierarchy are only used for maintaining the group key. Data communication within the group is conducted using the group key.

- We first consider the objective of minimizing the average number of multicast messages needed for an update, which is modeled by a routing tree where the multicast cost to every subset of the group is the same. For uniform multicast costs, we precisely characterize the optimal hierarchy when all the member weights are the same, and present a polynomial-time approximation scheme when member weights are nonuniform. These results appear in Section 3.
- We next show in Section 4 that the problem is NP-hard when multicast costs are nonuniform, even when the underlying routing network is a tree or when the member weights are uniform.
- Our main result is a constant-factor approximation algorithm in the general case of nonuniform member weights and nonuniform multicast costs captured by an arbitrary routing graph. We achieve a 75-approximation in general, and achieve improved constants of approximation for tree networks (11 for nonuniform weights and 4.2 for uniform weights). These results are in Section 5.

Our approximation algorithms are based on a simple divide-and-conquer framework that constructs "balanced" binary hierarchies by partitioning the routing graph using both the member weights and the routing costs. A key ingredient of our result for arbitrary routing graphs is the algorithm of [12] which, given any weighted graph, finds a spanning tree that simultaneously approximates the shortest path tree from a given node and the minimum spanning tree of the graph.

We have formulated the key hierarchy design as a static optimization problem, capturing the update frequencies as weights instead of explicitly modeling the time-varying membership of the group. Our formulation is applicable in scenarios where (a) the communication group is large with frequent updates, yet the update probability of any individual member is small; or (b) an update at a member may occur due to reasons other than change in membership, e.g., if the key is compromised, or if each "member" in the problem formulation actually represents a collection of members in a local network, one of whom is joining/leaving; or (c) the key hierarchy is periodically redesigned by solving the static optimization problem. Furthermore, the key hierarchies that we design in this paper are simple and may be amenable to maintain efficiently in a dynamic setting. We plan to investigate this aspect in future work.

1.2 Related work

Variants of the key hierarchy scheme studied in this paper were proposed by several independent groups [2, 4, 19, 21, 22]. The particular model we have adopted matches the Key Graph scheme of [22], where they show that a balanced hierarchy achieves an upper bound of $O(\log n)$ on the number of multicast messages needed for any update in a group of n members. In [20], it is shown that $\Theta(\log n)$ messages are necessary for an update in the worst case, for a general class of key distribution schemes. Lower bounds on the amount of communication needed under constraints on the number of keys stored at a user are given in [3]. Information-theoretic bounds on the number of auxiliary keys that need to be updated given member update frequencies are given in [17].

In recent work, [14] and [18] have studied the design of key hierarchy schemes that take into account the underlying routing costs and energy consumption in an ad hoc wireless network. The results of [14, 18], which consist of hardness proofs, heuristics, and simulation results, are closely tied to the wireless network model, relying on the broadcast nature of the medium. In this paper, we present approximation algorithms for a more basic routing cost model given by an undirected weighted graph.

The special case of uniform multicast costs (with nonuniform member weights) bears a strong resemblance to the Huffman encoding problem [9]. Indeed, it can be easily seen that an optimal *binary* hierarchy in this special case is given by the Huffman code. The truly optimal hierarchy, however, may contain internal nodes of both degree 2 and degree 3, which contribute different costs, respectively, to the leaves. In this sense, the problem seems related to Huffman coding with unequal letter costs [10], for which a PTAS is given in [6]. Another related problem is the broadcast tree scheduling problem where the goal is to determine

a schedule for broadcasting a message from a source node to all the other nodes in a heterogeneous network where different nodes may incur different delays between consecutive message transmissions [11, 15]. Both the Key Hierarchy Problem and the Broadcast Tree problem seek a rooted tree in which the cost for a node may depend on the degrees of the ancestors; however, the optimization objectives are different.

As mentioned in Section 1.1, our approximation algorithm for the general key hierarchy problem uses the elegant algorithm of [12] for finding spanning trees that simultaneously approximates both the minimum spanning tree weight and the shortest path tree weight (from a given root). Such graph structures, commonly referred to as *shallow-light trees* have been extensively studied (e.g., see [1, 13]).

2 Problem definition

An instance of the Key Hierarchy Problem is given by the tuple (S, w, G, c), where S is the set of group members, $w : S \to Z$ is the weight function (capturing the update probabilities), $G = (V, E)$ is the underlying communication network with $V \supseteq S \cup \{r\}$ where r is a distinguished node representing the group controller, and $c : E \to Z$ gives the cost of the edges in G.

In the remaining, we fix an instance (S, w, G, c). We define a *hierarchy* on a set $X \subseteq S$ to be a rooted tree H whose leaves are the elements of X. For a hierarchy T over X, the cost of a member $x \in X$ with respect to T is given by

$$\sum_{\text{ancestor } u \text{ of } x} \; \sum_{\text{child } v \text{ of } u} M(T_v) \tag{1}$$

where T_v is the set of leaves in the subtree of T rooted at v and for any set $Y \subseteq S$, $M(Y)$ is the cost of multicasting from the root r to Y in G. The cost of a hierarchy T over X is then simply the sum of the weighted costs of all the members of X with respect to T. The goal of the Key Hierarchy Problem is to determine a hierarchy of minimum cost. An example instance of the Key Hierarchy Problem, together with the calculation of the cost of a candidate hierarchy for the instance, is given in Figure 1.

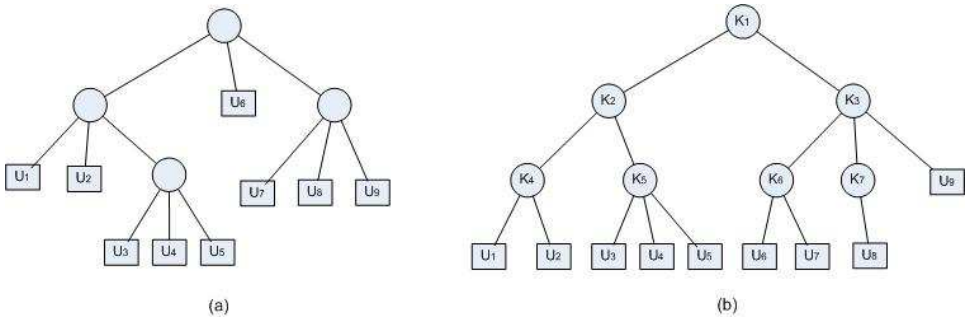

Figure 1: An instance of the Key Hierarchy Problem with 9 group members, connected to the group controller by a tree given in (a). Suppose the update frequency of every group member is 1 and the cost of every edge in the routing tree 1. An update at member U_4 will require the rekeying of keys K_5, K_2, and K_1. Key K_5 is rekeyed by unicasting to members U_3, U_4 and U_5 at a cost of 3 each. Key K_2 is rekeyed by multicasting to $\{U_1, U_2\}$ and to $\{U_3, U_4, U_5\}$ at a cost of 3 and 5, respectively. Finally, key K_1 is rekeyed by multicasting to $\{U_1, U_2, U_3, U_4, U_5\}$, to $\{U_6\}$ and to $\{U_7, U_8, U_9\}$ at a cost of 7, 1, and 4, respectively. Thus, the total cost of an update at member U_4 is 29. Using similar calculations, the average cost of an update can be determined to be 219/9.

We introduce some notation that is useful for the remainder of the paper. We use $\text{OPT}(S)$ to denote the cost of an optimal hierarchy for S. We extend the notation W to hierarchies and to sets of members: for any hierarchy T (resp., set X of members), $W(T)$ (resp., $W(X)$) denotes the sum of the weights of the leaves of T (resp., members in X). Our algorithms often combine a set $\mathcal{H}$ of two or three hierarchies to form another

hierarchy T': $\texttt{combine}(\mathcal{H})$ introduces a new root node R, makes the root of each hierarchy in $\mathcal{H}$ as a child of R, and returns the hierarchy rooted at R.

Using the above notation, a more convenient expression for the cost of a hierarchy T over X is the following reorganization of the summation in Equation 1:

$$\sum_{u \in T} W(T_u) \sum_{\text{child } v \text{ of } u} M(T_v) \tag{2}$$

3 Uniform multicast cost

In this section, we consider the special case of the Key Hierarchy problem where the multicast cost to any subset of group members is the same. Thus, the objective is to minimize the average number of multicast messages sent for an update. We note that the number of multicast messages sent for an update at a member u is simply the sum of the degrees of its ancestors in the hierarchy (as is evident from Equation 1). We start by establishing a basic structural property of an optimal hierarchy and a lower bound on the optimum cost.

Lemma 1. *For any given member set S with at least two members, there exists an optimal hierarchy in which the degree of every internal node is either two or three.*

Proof. Let T^* be an optimal hierarchy for S. Since any internal node with degree one can be replaced by its child, yielding a decrease in cost, the degree of every internal node of T^* is at least two. Let, if possible, v be an internal node of T^* with degree $d \geq 4$. We divide its children into two groups C_1 and C_2, containing $\lceil d/2 \rceil$ and $\lfloor d/2 \rfloor$ children, respectively. We add two new internal nodes v_1 and v_2, make them children of v, and set v_1 and v_2 to be the parents of the nodes in C_1 and C_2, respectively.

We now consider the cost of the new hierarchy. The cost of any member that does not have v as an ancestor in T^* does not change. The cost of a member that has v as an ancestor in T^* decreases by at least $d - \lceil d/2 \rceil - 2 \geq 0$; thus, this cost is nonincreasing. If $d > 4$, there exists a member whose cost decreases by at least $d - \lfloor d/2 \rfloor - 2 > 0$, contradicting the optimality of T^*. If $d = 4$, then we have a new hierarchy whose cost is no more than that of T^* and has fewer internal nodes with degree greater than three. Repeating this process until there are no internal nodes with degree greater than 3 yields the desired claim. □

Lemma 2. *For any member set S, we have $OPT(S) \geq \sum_{v \in S} 3w_v \log_3(W(S)/w_v)$.*

Proof. The proof is by induction on the size of S. The claim is trivially true for $|S| = 1$. For the induction hypothesis, we assume that the claim is true for member sets of size less than $m \geq 2$. Consider an optimal hierarchy for S with $|S| = m \geq 2$. Let the degree of the root be d, and let the member set in the subtree rooted at the ith child be S_i with $|S_i| = m_i$, $1 \leq i \leq d$. We place the following lower bound on $\text{OPT}(S)$:

$$\begin{aligned}
\text{OPT}(S) &\geq dW(S) + \sum_{1 \leq i \leq d} \sum_{v \in S_i} 3w_i \log_3(W(S_i)/w_v) \\
&= dW(S) + 3W(S) \sum_{1 \leq i \leq d} \log_3 W(S_i) - \sum_{v \in S} 3w_v \log_3 w_v \\
&\geq dW(S) + 3W(S) \log_3(W(S)/d) - \sum_{v \in S} 3w_v \log_3 w_v \\
&= dW(S) - 3W(S) \log_3 d + \sum_{v \in S} 3w_v \log_3(W(S)/w_v) \\
&\geq \sum_{v \in S} 3w_v \log_3(W(S)/w_v).
\end{aligned}$$

(The third step holds owing to the convexity of the function $x \log_3 x$. The last step holds since $d \geq 3 \log_3 d$ for all positive integers d.) □

3.1 Structure of an optimal hierarchy for uniform member weights

When all the members have the same weight, we can easily characterize an optimal key hierarchy. Let n be the number of members, and $k = 3^{\lfloor \log_3 n \rfloor}$. We now show that the following key hierarchy is optimal. We first build a balanced degree-3 tree T with k leaves. If $k \leq n < 2k$, then we add two members as children to the first $n-k$ leaves of T, and place the remaining $2k-n$ members at the remaining $2k-n$ leaves of T. If $2k \leq n < 3k$, we add three members as children for the first $n-2k$ leaves of T, and place the remaining $6k-2n$ members as the children, two at a time, of the remaining $3k-n$ leaves of T. It is easy to verify by a simple case analysis that the cost of this hierarchy is given by:

$$f(n) = \begin{cases} 3n\lfloor \log_3 n \rfloor + 4(n-k) & \text{when } k \leq n < 2k \\ 3n\lfloor \log_3 n \rfloor + 5n - 6k & \text{when } 2k \leq n < 3k \end{cases}$$

Theorem 1. *For uniform multicast costs and uniform member weights, the above key hierarchy is optimal.*

Proof: We prove this by induction on the number of members. We first note that $f(n)$ is a convex function. By Lemma 1 we know the degree of the root can only be either two or three. Since the proposed hierarchy is obtained by having a degree-3 root and balanced distribution of members among the children, all we need to show is that for any $n \geq 3$, there exists an optimal hierarchy in which the root has three children. If the number of members is at most 5, this can be checked by hand.

We now consider $n \geq 6$. Assume the root of an optimal key hierarchy has degree 2. By the convexity of $f(n)$, we know each subtree has more than 3 members. And by induction hypothesis we know the root of each subtree has degree 3. Let the two children of the root be u_1 and u_2. Let the children of u_i be u_{i1}, u_{i2}, and u_{i3}, $1 \leq i \leq 2$. We transform this hierarchy into another key hierarchy with the same cost by adding a third child u_3 to the root that has as its children u_{13} and u_{23}. The cost of every member in the new hierarchy remains the same as that in the optimal hierarchy, which means this new hierarchy is also optimal and its root has degree 3. This completes the proof of the theorem. □

3.2 A polynomial-time approximation scheme for nonuniform member weights

We give a polynomial-time approximation scheme for the Key Hierarchy Problem when the multicast cost to every subset of the group is identical and the members have arbitrary weights. Given a positive constant ε, we present an polynomial-time algorithm that produces a $(1+O(\varepsilon))$-approximation. We assume that $1/\varepsilon$ is a power of 3; if not, we can replace ε by a smaller constant that satisfies this condition. We round the weight of every member up to the nearest multiple of $1+\varepsilon$ at the expense of a factor of $1+\varepsilon$ in approximation. Thus, in the remainder we assume that every weight is a power of $(1+\varepsilon)$. Our algorithm PTAS(), which takes as input a set S of members with weights, is as follows.

1. Divide the members of S into two sets, a set H of the $3^{1/\varepsilon^2}$ members with the largest weight and the set $L = S - H$.

2. Initialize $\mathcal{L}$ to be the set of hierarchies consisting of one depth-0 hierarchy for each member of L.

3. Repeat the following step until it can no longer be executed: if T_1, T_2, and T_3 are hierarchies in $\mathcal{L}$ with identical weight, then replace T_1, T_2, and T_3 in $\mathcal{L}$ by `combine`$(\{T_1, T_2, T_3\})$. (Recall the definition of `combine` from Section 2.)

4. Repeat the following step until $\mathcal{L}$ has one hierarchy: replace the two hierarchies T_1, T_2 with least weight by `combine`$(\{T_1, T_2\})$. Let T_L denote the hierarchy in $\mathcal{L}$.

5. Compute an optimal hierarchy T^* for H. Determine a node in T^* that has weight at most $W(S)\varepsilon$ and height at most $1/\varepsilon$. We note that such a node exists since every hierarchy with at least ℓ leaves has a set N of at least $1/\varepsilon$ nodes at depth at most $1/\varepsilon$ with the property that no node in N is an ancestor of another. Set the root of T_L as the child of this node. Return T^*.

We now analyze the above algorithm. At the end of step 3, the cost of any hierarchy T in $\mathcal{L}$ is exactly equal to

$$\sum_{v \in T} 3w_v \log_3(W(T)/w_v).$$

If $\mathcal{L}$ is the hierarchy set at the end of step 3, then the additional cost incurred in step 4 is at most

$$\sum_{T \in \mathcal{L}} 2W(T) \log_2(W(L)/W(T)).$$

Since there are at most two hierarchies in any weight category in $\mathcal{L}$ at the start of step 4, at least $1 - 1/\varepsilon^2$ of the weight in the hierarchy set is concentrated in the heaviest $4/\varepsilon^3$ hierarchies of $\mathcal{L}$. Step 4 is essentially the Huffman coding algorithm and yields an optimal binary hierachy. Using Lemma 3 of Section 5, we note that it achieves a 3-approximation. (In fact, one can show using a more careful argument that it achieves an approximation of $2\lg((1+\sqrt{5})/2)/(3\lg 3) \approx 1.52$, but the factor 3 will suffice for our purposes here.) This yields the following bound on the increase in cost due to step 4:

$$3\left(\varepsilon^2 W(L) \log_{1+\varepsilon} 3 + (1-\varepsilon^2) W(L) \log_2(4/\varepsilon^2)\right) \leq W(L)/\varepsilon,$$

for ε sufficiently small. The final step of the algorithm increases the cost by at most $W(L)/\varepsilon + \varepsilon W(S)$. Thus, the total cost of the final hierarchy is at most

$$\begin{aligned} & \mathrm{OPT}(H) + \mathrm{OPT}(L) + W(L)/\varepsilon + W(L)/\varepsilon + \varepsilon W(S) \\ \leq\ & \mathrm{OPT}(H) + \mathrm{OPT}(L) + 2\varepsilon \mathrm{OPT}(S) + \varepsilon \mathrm{OPT}(S) \\ \leq\ & (1+3\varepsilon)\mathrm{OPT}(S). \end{aligned}$$

(The second step holds since $\mathrm{OPT}(S) \geq \sum_{v \in L} w_v \log_3(W(S)/w_v) \geq W(L)/\varepsilon^2$.)

4 Hardness results

In this section, we present the hardness results for Key Hierarchy problem with non-uniform multicast cost, i.e. the multicast cost depends on the underlying routing structure. First we show that the Key Hierarchy problem is strongly NP-complete if group members have different weights and the underlying routing network is a tree. Then we show the problem is also NP-complete even if group members have the same weights and the underlying routing network is a general graph.

4.1 Weighted key hierarchy problem with routing tree

Our reduction is from the NP-complete problem 3-PARTITION, which is defined as follows [5]. The input consists of a set A of $3m$ elements, a bound $B \in Z^+$, and a set of sizes $S(a) \in Z^+$ for each $a \in A$ such that $B/4 < S(a) < B/2$, and $\sum_{a \in A} S(a) = mB$. The goal of the problem is to determine whether A can be partitioned into m disjoint sets $A_1, A_2, \ldots, A_m$ such that for $1 \leq i \leq m$, $\sum_{a \in A_i} S(a) = B$.

Theorem 2. *When group members have different weights and the routing network is a tree, the Key Hierarchy Problem is NP-complete.*

Proof: The membership in NP is immediate. We reduce 3-PARTITION to the Key Hierarchy Problem. Let P denote the given 3-PARTITION instance. If the number $3m$ of elements in the P is not a power of three, then we add new elements in groups of three with sizes B, 0, and 0, respectively, to make the total number of elements a power of 3. It is easy to verify that the original problem instance has the desired partition if and only if the new instance has the desired partition. Thus, for the remainder of the proof, we assume that the number of elements, $3m$, in P is a power of 3.

In P, let set A be $\{a_1, a_2, \ldots, a_{3m}\}$, and the size of element a_i in set A be w_i'. We create a routing tree T consisting of a root r connected to a single internal node u, which in turn has edges to $3m$ leaves v_i for $i = 1, 2, \ldots, 3m$, one for each of the $3m$ members. Root r is the group controller. For member i, we set its weight w_i to be $w + w_i'$, where w is chosen such that $\frac{w_{max}}{w_{min}} < \frac{3 \cdot 3m \log_3 3m+1}{3 \cdot 3m \log_3 3m}$, where $w_{max} = \max_i\{w + w_i\}$ and $w_{min} = \min_i\{w + w_i\}$. We set the cost of edge (r, u) to be C, a constant which will be specified later, and the cost of (u, v_i) to be w_i for $i = 1, 2, \ldots, 3m$, and the weight of leaf v_i to be w_i. We now show that P has a partition if and only if the optimal key hierarchy of T has cost $C \cdot 3W \log_3 3m + W^2 \cdot (1 + 1/3 + 1/9 + \cdots + 1/m)$, where W is the sum of the weights of all the members.

The basic idea of the argument is as follows. By setting C much larger than $\sum_i w_i$, we will ensure that the multicast cost to any subset of the leaves would be approximately the same. Thus, we can use the structural result proved in Theorem 1. In an optimal key hierarchy, each internal node has 3 children, which means elements in P problem are divided into m groups, each having 3 elements. We will then argue that the cost of this key hierarchy is minimized when the total weight of the leaves in every group is identical.

Let $W = \sum_i w_i$. If $C > W^2 \log_3 3m$ (this is the condition for C to be the dominant factor in multicast costs), then the cost of an optimal key hierarchy is smaller than $C \cdot 3 \cdot 3m \log_3 3m \cdot w_{max}$ (This is the optimal key hierarchy cost if there are $3m$ members, each of which has weight w_{max}. And the multicast cost is uniform for every subset of members, which is c). If in an optimal key hierarchy, not every internal node has degree 3, then its cost is at least $C \cdot (3 \cdot 3m \log_3 n + 1) \cdot w_{min}$. Since $\frac{w_{max}}{w_{min}} < \frac{3 \cdot 3m \log_3 3m+1}{3 \cdot 3m \log_3 3m}$, we know such key hierarchy cannot be optimal. So balanced degree-3 tree is the only optimal key hierarchy in this case. Now we analyze the cost of this optimal key hierarchy more carefully. The cost contributed by edge (r, u) is exactly $C \cdot 3W \log_3 3m$. Let C_i denote the set of nodes at depth i in the hierarchy, the depth of the root being set to 0. By Equation 2, the cost contributed by edges (u, v_i), $i = 1, 2, \ldots, 3m$, equals

$$\begin{aligned} \sum_{0 \le i \le \log_3 m} \sum_{x \in C_i} W(T_x) \cdot (M(T_x) - C) &= \sum_{0 \le i \le \log_3 m} \sum_{x \in C_i} W(T_x)^2 \\ &\ge W^2 + 3(W/3)^2 + 9(W/9)^2 + \cdots + m(W/m)^2. \end{aligned}$$

In the last step, equality only holds when $W(T_x) = W/3^i$ for all $x \in C_i$ (by Jensen's inequality). Thus, the 3-partition problem has a solution if and only if the optimal key hierarchy achieves its minimum, which is $C \cdot 3W \log_3 3m + W^2 \cdot (1 + 1/3 + 1/9 + \cdots + 1/m)$. □

4.2 Unweighted key hierarchy problem

Our reduction is from the NP-complete 3D-MATCHING problem which is defined as follows [5]. We are given finite disjoint sets W, U, V of size q, and a set of triples $M \subseteq W \times U \times V$. The goal is to determine whether there are q pairwise disjoint triples.

Theorem 3. *When group members have the same key update weights and the routing network is a general graph, the Key Hierarchy Problem is NP-complete.*

Proof: We reduce 3D-MATCHING to the Key Hierarchy problem. Let I be a given instance of 3D-MATCHING. If the set size q is not a power of 3 and q' is the smallest power of 3 larger than q, then we construct a new instance of 3D-MATCHING by adding $q' - q$ new elements to each of W, U, and V as

follows: for $1 \leq i \leq q' - q$, add w'_i to W, u'_i to U, v'_i to V, and (w'_i, u'_i, v'_i) to M. It is easy to see that the original 3D-Matching instance has a solution if and only if this new 3D-Matching instance has a solution. So from now on we can assume that q is a power of 3.

For given instance I, we construct a routing graph as follows. Create vertices $w_1, w_2, \ldots, w_q$ to represent each element in set W, $u_1, u_2, \ldots, u_q$ to represent each element in set U, and $v_1, v_2, \ldots, v_q$ to represent each element in set V. Then create $|M|$ vertices $t_1, t_2, \ldots, t_{|M|}$, and for each element $m_i = (w_x, u_y, v_z) \in M$, add edges $(t_i, w_x), (t_i, u_y), (t_i, v_z)$ of unit cost to the routing graph. Create another vertex s, and add edges (s, t_i) for $i = 1, 2, \ldots, |M|$ of unit cost. Finally, create vertex r, and add an edge (r, s) with cost c. Vertex r is the group controller, and $W \cup U \cup V$ is the set of group members.

If we set c to be greater than $(|M| + 3q) \cdot 3 \cdot 3q \log_3 3q$, then using an argument similar to the proof of Theorem 2, we can show that the optimal key hierarchy is a balanced degree-3 tree. We will next argue that there is a matching in I if and only if the cost of the optimal key hierarchy is $c \cdot 3 \cdot 3q \log_3 3q + 12q^2 \log_3 3q$.

We now calculate the cost of the optimal hierarchy using Equation 2. The cost contributed by edge (r, s) is exactly $c \cdot 3 \cdot 3q \log_3 3q$. The cost contributed by edges (t_i, w_x), (t_i, u_y) and (t_i, v_z) where $i = 1, 2, \ldots, |M|$ and $x, y, z = 1, 2, \ldots, q$, is $3q$. The cost contributed by edges (s, t_i), $i = 1, 2, \ldots, |M|$, is q. This minimum is achieved only if there is a 3D-Matching. So there is a solution to the 3D-Matching problem if and only if the cost of the optimal logical tree is $c \cdot 3 \cdot 3q \log_3 3q + 12q^2 \log_3 3q$. And this complete the proof of this theorem. □

5 Approximation algorithms for nonuniform multicast costs

In this section, we present constant-factor approximation algorithms for the Key Hierarchy Problem with nonuniform multicast costs. We first show that for any instance, there always exists a binary hierarchy that is 3-approximate. This guides the design of our approximation algorithms. We next present, in Section 5.1, an 11-approximation algorithm for the case where the underlying communication network is a tree. Finally, we present, in Section 5.2 a 75-approximation algorithm for the most general case of our problem, where the communication network is an arbitrary weighted graph.

Lemma 3. *For any instance, there exists a 3-approximate binary hierarchy.*

Proof. Consider any optimal hierarchy T. Following Equation 2, we associate with each node u of T a cost equal to $W(T_u) \sum_{\text{child } v \text{ of } u} M(T_v)$; we refer to this cost as $\text{nc}(u)$. We show how to transform T to a binary hierarchy by repeatedly replacing a node, say u, with degree $d \geq 2$, by a node u' of degree two and a set U of at most two other nodes, each with degree strictly less than d. To argue the bound on the cost of the binary hierarchy, we use a charging argument: in particular, we show that $3\text{nc}(u) \geq \text{nc}(u') + \sum_{v \in U} 3\text{nc}(v)$.

Consider any node u of T of degree greater than two. We consider two cases, as illustrated in Figure 2. The first case is where there is no child of u that has weight at least one-third of the weight under u. We divide the children of u into two groups such that each group has at least one-third of weight under u. If such a partition exists, then we replace u by three nodes: u', u_1, and u_2. The parent of node u' is the same as the parent of u (if it exists). The node u' is the parent for both u_1 and u_2. Finally, u_1 and u_2 are the parents

of the children of u in the two groups of the partition, respectively.

$$\begin{aligned}
3\mathrm{nc}(u) &= 3W(T_u) \sum_{\text{child } v \text{ of } u} M(T_v) \\
&\leq W(T_u)(M(T_{u_1}) + M(T_{u_2})) + 2W(T_u) \sum_{\text{child } v \text{ of } u} M(T_v) \\
&= \mathrm{nc}(u') + 2W(T_u) \sum_{\text{child } v \text{ of } u_1} M(T_v) + 2W(T_u) \sum_{\text{child } v \text{ of } u_2} M(T_v) \\
&\leq \mathrm{nc}(u') + 3\mathrm{nc}(u_1) + 3\mathrm{nc}(u_2).
\end{aligned}$$

The second case is where u has a child u_1 with weight at least two-third of the total weight under u. In this case, we replace u by two nodes u' and u_2, with u' becoming the parent of u_1 and u_2, and u_2 becoming the parent of the other children of u. The parent of u' is the same as that of u (if it exists). Using a similar argument as above, we obtain

$$\begin{aligned}
3\mathrm{nc}(u) &= 3W(T_u) \sum_{\text{child } v \text{ of } u} M(T_v) \\
&\leq \mathrm{nc}(u') + 3\mathrm{nc}(u_2).
\end{aligned}$$

□

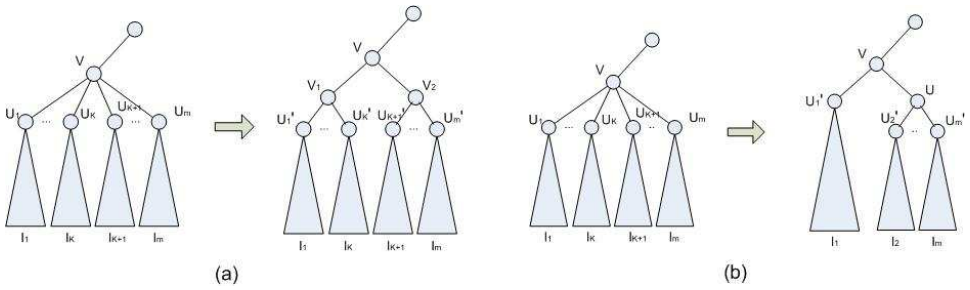

Figure 2: Binary logical tree transformation

5.1 Approximation algorithms for routing trees

In this section, we first give an 11-approximation algorithm for the case where weights are nonuniform and the routing network is a tree. Then we analyze the more special case with uniform weights, and improve the approximation factor to 4.2.

Given any routing tree, let S be the set of members. We start with defining a procedure `partition`$(\cdot)$ that takes as input the set S and returns a pair (X, v) where X is a subset of S and v is a node in the routing tree. First, we determine if there is an internal node v that has a subset C of children such that the total weight of the members in the subtrees of the routing tree rooted at the nodes in C is between $W(S)/3$ and $2W(S)/3$. If v exists, then we partition S into two parts X, which is the set of members in the subtrees rooted at the nodes in C, and $S \setminus X$. It follows that $W(S)/3 \leq W(X) \leq 2W(S)/3$. (This case is illustrated in Figure 3, where Y denotes $S \setminus X$.) If v does not exist, then it is easy to see that there is a single member with weight more than $2W(S)/3$. In this case, we set X to be the singleton set which contains this heavy node which we call v. The procedure `partition`(S) returns the pair (X, v). In the remainder, we let Y denote $S \setminus X$.

Having found such a partition (X, Y) we recursively obtain hierarchies for X and Y, using the PTAS of Section 3.2 if the routing cost between the root and the partition node v is a large fraction of the total multicast cost. Let T_1 and T_2 be the key hierarchies thus obtained for X and Y, respectively. The hierarchy for S is obtained by combining T_1 and T_2 using the `combine` subroutine. The following is our full approximation algorithm. (PTAS is the algorithm introduced in Section 3.2)

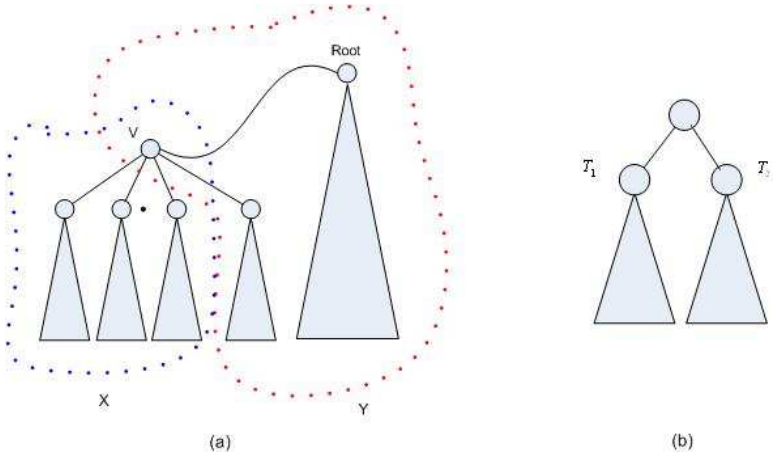

Figure 3: An arbitrary routing tree

ApproxTree(S)

1. If S is a singleton set, then return the trivial hierarchy with a single node.
2. $(X, v) =$ `partition`(S); let Y denote $S \setminus X$.
3. Let Δ be the cost from root to partition node v. If $\Delta \leq M(S)/5$, $T_1 =$ApproxTree(X); otherwise $T_1 =$ PTAS(X).
4. $T_2 =$ApproxTree(Y).
5. Return `combine`(T_1, T_2).

We now show that ApproxTree is a constant-factor approximation algorithm. Let ALG(S) be the key hierarchy constructed by our algorithm, OPT(S) be the optimal key hierarchy, OPT'(X) be the induced key hierarchy from OPT(S) by set X, and OPT'(Y) be the induced key hierarchy from OPT(S) by set Y. In the following proof, we abuse our notation and use ALG($\cdot$) and OPT($\cdot$), to refer to both the key hierarchies and their cost. We first note that OPT'(X) $\geq$ OPT(X), OPT'(Y) $\geq$ OPT(Y), and OPT(S) $\geq$ OPT'(X) + OPT'(Y). So OPT(S) $\geq$ OPT(X) + OPT(Y).

Theorem 4. *Algorithm **ApproxTree** is an $(11 + \varepsilon)$-approximation, where $\varepsilon > 0$ can be made arbitrarily small.*

Proof. We prove by induction on the number of members in S that ALG(S) $\leq \alpha \cdot$OPT(S)$+\beta \cdot W(S)M(S)$, for constants α and β specified later. The induction base, when $|S| \leq 2$, is trivial. For the induction step, we consider three cases depending on the distance to the partition node and whether we obtain a balanced partition; we say that a partition (X, Y) is balanced if $\frac{1}{3}W(S) \leq W(X), W(Y) \leq \frac{2}{3}W(S)$.

The first case is where $\Delta \leq M(S)/5$ and the partition is balanced. In this case, we have

$$\begin{aligned}
\text{ALG}(S) &= \text{ALG}(X) + \text{ALG}(Y) + W(S)\,[M(X) + M(Y)] \\
&\leq \alpha \cdot \text{OPT}(X) + \beta \cdot W(X)M(X) + \alpha \cdot \text{OPT}(Y) + \beta \cdot W(Y)M(Y) + W(S)\,[M(X) + M(Y)] \\
&\leq \alpha\,[\text{OPT}(X) + \text{OPT}(Y)] + \left(\frac{2}{3}\beta + 1\right) W(S)\,[M(X) + M(Y)] \\
&\leq \alpha \cdot \text{OPT}(S) + \left(\frac{2}{3}\beta + 1\right) W(S)\,[M(S) + \Delta] \\
&\leq \alpha \cdot \text{OPT}(S) + \frac{6}{5}\left(\frac{2}{3}\beta + 1\right) w(S)M(S) \\
&\leq \alpha \cdot OPT(S) + \beta \cdot w(S)M(S)
\end{aligned}$$

as long as $\frac{6}{5}\left(\frac{2}{3}\beta + 1\right) \leq \beta$, which is true if $\beta \geq 6$.

The second case is where $\Delta > M(S)/5$ and the partition is balanced. In this case, we only call the algorithm recursively on Y and use PTAS on X. So we have the following.

$$\begin{aligned}
\text{ALG}(S) &= \text{PTAS}(X) + \text{ALG}(Y) + W(S)\,[M(X) + M(Y)] \\
&\leq 5(1+\varepsilon) \cdot \text{OPT}(X) + \alpha \cdot \text{OPT}(Y) + \beta \cdot W(Y)M(Y) + W(S)\,[M(X) + M(Y)] \\
&\leq \alpha\,[\text{OPT}(X) + \text{OPT}(Y)] + \frac{2}{3}\beta W(S)M(S) + 2W(S)M(S) \\
&\leq \alpha \cdot \text{OPT}(S) + \left(\frac{2}{3}\beta + 2\right) W(S)M(S) \\
&\leq \alpha \cdot \text{OPT}(S) + \beta \cdot W(S)M(S)
\end{aligned}$$

as long as $\alpha \geq 5(1+\varepsilon)$ and $\frac{2}{3}\beta + 2 \leq \beta$ which is true if $\beta \geq 6$.

The third case is when the partition is not balanced (i.e. $W(X) > \frac{2}{3}W(S)$). In this case, our algorithm connects the heavy node directly to the root of the hierarchy. So we have the following.

$$\begin{aligned}
\text{ALG}(S) &= \text{ALG}(Y) + W(S)\,[M(X) + M(Y)] \\
&\leq \alpha \cdot \text{OPT}(Y) + \beta \cdot W(Y)M(Y) + W(S)\,[M(X) + M(Y)] \\
&\leq \alpha \cdot \text{OPT}(S) + \frac{1}{3}\beta W(S)M(S) + 2W(S)M(S) \\
&\leq \alpha \cdot \text{OPT}(S) + \left(\frac{1}{3}\beta + 2\right) W(S)M(S) \\
&\leq \alpha \cdot \text{OPT}(S) + \beta \cdot W(S)M(S)
\end{aligned}$$

as long as $\frac{1}{3}\beta + 2 \leq \beta$ which is true if $\beta \geq 3$.

So, by induction, we have shown $\text{ALG}(S) \leq \alpha \cdot \text{OPT}(S) + \beta \cdot W(S)M(S)$ for $\alpha \geq 5(1+\varepsilon)$ and $\beta \geq 6$. Since $\text{OPT}(S) \geq W(S)M(S)$, we obtain an $(11+\varepsilon)$-approximation. □

If the member weights are uniform, then we can improve the approximation ratio to 4.2 using a more careful analysis of the same algorithm. We refer the reader to the appendix for details.

5.2 Approximation algorithms for routing graphs

In this section, we give a constant-factor approximation algorithm for the case where weights are nonuniform and the routing network is an arbitrary graph. In our algorithm, we compute light approximate shortest-path trees (LAST) [12] of subgraphs of the routing graph. An (α, β)-LAST of a given weighted graph G is a

spanning tree T of G such that the the shortest path in T from a specified root to any vertex is at most α times the shortest path from the root to the vertex in G, and the total weight of T is at most β times the minimum spanning tree of G. For any $\gamma > 0$, the algorithm of [12] yields a an (α, β)-LAST with $\alpha = 1 + \sqrt{2}\gamma$ and $\beta = 1 + \sqrt{2}/\gamma$, where γ can be chosen as an input parameter.

ApproxGraph(S)

1. If S is a singleton set, return the trivial hierarchy with one node.
2. Compute the complete graph on $S \cup \{root\}$. The weight of an edge (u, v) is the length of shortest path between u and v in the original routing graph.
3. Compute the minimum spanning tree on this complete graph. Call it MST(S).
4. Compute an (α, β)-LAST L of MST(S).
5. $(X, v) = \texttt{partition}(L)$.
6. Let Δ be the cost from root to partition node L. If $\Delta \leq M(S)/5$, T_1 =ApproxGraph(X). Otherwise, T_1 = PTAS(X).
7. T_2 =ApproxGraph(Y).
8. Return $\texttt{combine}(T_1, T_2)$.

When we calculate the multicast cost to a subset of members in the routing graph, the optimum multicast is by a minimum Steiner tree, computing which is NP-hard. But one can easily approximate it, for instance by selecting a minimum spanning tree in the metric space connecting the root to the desired members (the metric being the shortest path cost in the routing graph). So in the following proof, we define $M(S)$ to be the cost of the minimum spanning tree connecting the root to S in the complete graph $G(S)$ whose vertices are the ones in $S \cup \{root\}$ and the weight of an edge (u, v) is the shortest path distance between u and v in the routing graph. And the cost we obtain by this definition is a 2-approximation of the cost obtained by using the optimal multicasts in the graph.

Theorem 5. *The algorithm **ApproxGraph** is a constant-factor approximation.*

Proof. We prove by induction on the number of members in S that ALG(S) $\leq \alpha \cdot$OPT(S)$+\beta \cdot W(S)M(S)$, for constants α and β specified later. The induction base, when $|S| \leq 2$, is trivial. For the induction step, we consider three cases. The first case is $\Delta \leq M(S)/5$ and the partition is balanced (as defined in the proof of Theorem 4). Let $M_L(S)$ be the multicast cost to S in LAST. From the description of LAST we know $M_L(S) \leq \left(1 + \sqrt{2}/\gamma\right) \cdot M(S)$. Also we have $M_L(S) \geq M_L(X) + M_L(Y) - \Delta \geq M(X) + M(Y) - \Delta$. So $\left(1 + \sqrt{2}/\gamma\right) \cdot M(S) \geq M(X) + M(Y) - \Delta$.

$$\begin{aligned}
\text{ALG}(S) &= \text{ALG}(X) + \text{ALG}(Y) + W(S)\left[M(X) + M(Y)\right] \\
&\leq \alpha \cdot \text{OPT}(X) + \beta \cdot W(X)M(X) + \alpha \cdot \text{OPT}(Y) + \beta \cdot W(Y)M(Y) \\
&\quad + W(S)\left[M(X) + M(Y)\right] \\
&\leq \alpha\left[\text{OPT}(X) + \text{OPT}(Y)\right] + \left(\frac{2}{3}\beta + 1\right) W(S)\left[M(X) + M(Y)\right] \\
&\leq \alpha \cdot \text{OPT}(S) + \left(\frac{2}{3}\beta + 1\right) W(S)\left[\left(1 + \sqrt{2}/\gamma\right) M(S) + \Delta\right] \\
&\leq \alpha \cdot \text{OPT}(S) + \left(\frac{6}{5} + \sqrt{2}/\gamma\right)\left(\frac{2}{3}\beta + 1\right) W(S)M(S) \\
&\leq \alpha \cdot OPT(S) + \beta \cdot W(S)M(S)
\end{aligned}$$

as long as $\left(\frac{6}{5} + \sqrt{2}/\gamma\right)\left(\frac{2}{3}\beta + 1\right) \leq \beta$.

The second case is $\Delta > M(S)/5$ and the partition is balanced. In this case, we only call the algorithm recursively on Y and use the PTAS for X. Since $\Delta > M(S)/5$, the distance from the root to any element in X is at least $\frac{\Delta}{1+\sqrt{2}\gamma} = \frac{M(S)}{5(1+\sqrt{2}\gamma)}$. So the multicast cost to any subset of X is between $\frac{M(S)}{5(1+\sqrt{2}\gamma)}$ and $M(S)$. By using the PTAS, we have a $5(1+\varepsilon)(1+\sqrt{2}\gamma)$-approximation on $\text{OPT}(X)$. So we have the following bound on $\text{ALG}(S)$.

$$\begin{aligned}
\text{ALG}(S) &= \text{PTAS}(X) + \text{ALG}(Y) + W(S)\left[M(X) + M(Y)\right] \\
&\leq 5\left(1 + \sqrt{2}\gamma\right)(1+\varepsilon) \cdot \text{OPT}(X) + \alpha \cdot \text{OPT}(Y) + \beta \cdot W(Y)M(Y) + W(S)\left[M(X) + M(Y)\right] \\
&\leq \alpha\left[\text{OPT}(X) + \text{OPT}(Y)\right] + \frac{2}{3}\beta W(S)M(S) + 2W(S)M(S) \\
&\leq \alpha \cdot \text{OPT}(S) + \left(\frac{2}{3}\beta + 2\right) W(S)M(S) \\
&\leq \alpha \cdot \text{OPT}(S) + \beta \cdot W(S)M(S)
\end{aligned}$$

as long as $\alpha \geq 5\left(1 + \sqrt{2}\gamma\right)(1+\varepsilon)$ and $\beta \geq 6$.

The third case is when the partition is not balanced. In this case, our algorithm connecta the heavy node directly to the root of key hierarchy. So we have the following bound on $\text{ALG}(S)$.

$$\begin{aligned}
\text{ALG}(S) &= \text{ALG}(Y) + W(S)\left[M(X) + M(Y)\right] \\
&\leq \alpha \cdot \text{OPT}(Y) + \beta \cdot W(Y)M(Y) + W(S)\left[M(X) + M(Y)\right] \\
&\leq \alpha \cdot \text{OPT}(S) + \frac{1}{3}\beta W(S)M(S) + 2W(S)M(S) \\
&\leq \alpha \cdot \text{OPT}(S) + \left(\frac{1}{3}\beta + 2\right) W(S)M(S) \\
&\leq \alpha \cdot \text{OPT}(S) + \beta \cdot W(S)M(S)
\end{aligned}$$

as long as $\beta \geq 3$. So, this algorithm has a constant approximation.

So, by induction, we have shown $\text{ALG}(S) \leq \alpha \cdot \text{OPT}(S) + \beta \cdot W(S)M(S)$, implying an $(\alpha + \beta)$-approximation. When $\gamma = 7$, from the constraints, we obtain $\alpha \geq 54$ and $\beta \geq 21$. So we have a 75-approximation. $\square$

6 Discussion

We have presented a constant-factor approximation algorithm for the Key Hierarchy Problem for the general case where the member weights are nonuniform and the communication network is an arbitrary graph. While we do obtain improved approximation factors when the communication network is a tree, the factors achieved are large and need to be improved. We have also given a polynomial-time approximation scheme for the problem instance where all multicasts cost the same. We do not know, however, whether this problem is NP-complete. As discussed in Section 1.2, the problem is related to the classic Huffman coding problem with nonuniform letter costs, whose complexity (P vs NP-hardness) is also not yet resolved.

There are several other directions for future research. We are currently exploring the dynamic maintenance of our key hierarchies, explicitly modeling the joining and leaving of members, while maintaining the constant-factor approximation in cost. We would also like to study the design of key hierarchies where the members have a bound on the number of auxiliary keys they store. Also of interest is the case where we have no (or limited) information on the update frequencies of the members.

References

[1] Baruch Awerbuch, Alan E. Baratz, and David Peleg. Cost-sensitive analysis of communication protocols. In *PODC*, pages 177–187, 1990.

[2] Ran Canetti, Juan Garay, Gene Itkis, Daniele Micciancio, Moni Naor, and Benny Pinkas. Multicast security: A taxonomy and some efficient constructions. In *INFOCOMM'99*, 1999.

[3] Ran Canetti, Tal Malkin, and Kobbi Nissim. Efficient communication-storage tradeoffs for multicast encryption. In *EUROCRYPT*, pages 459–474, 1999.

[4] Germano Caronni, Marcel Waldvogel, Dan Sun, and Bernhard Plattner. Efficient security for large and dynamic multicast groups. In *Proceedings of WETICE*, pages 376–383, Washington, DC, USA, 1998. IEEE Computer Society.

[5] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman, New York, 1979.

[6] Mordecai J. Golin, Claire Kenyon, and Neal E. Young. Huffman coding with unequal letter costs. In *Proceedings of STOC*, pages 785–791, New York, NY, USA, 2002. ACM.

[7] H. Harney and C. Muckenhirn. Group key management protocol (GKMP) architecture. Internet RFC 2094, 1997.

[8] H. Harney and C. Muckenhirn. Group key management protocol (GKMP) specification. Internet RFC 2093, 1997.

[9] D. Huffman. A method for the construction of minimum-redundancy codes. *Proceedings of the IRE*, 40:1098–1101, 1952.

[10] Richard Karp. Minimum-redundancy coding for the discrete noiseless channel. *IRE Transactions on Information Theory*, 7:27–39, 1961.

[11] Samir Khuller and Yoo Ah Kim. Broadcasting in heterogeneous networks. *Algorithmica*, 48(1):1–21, 2007.

[12] Samir Khuller, Balaji Raghavachari, and Neal E. Young. Balancing minimum spanning trees and shortest-path trees. *Algorithmica*, 14(4):305–321, 1995.

[13] Guy Kortsarz and David Peleg. Approximating shallow-light trees (extended abstract). In *SODA*, pages 103–110, 1997.

[14] L. Lazos and R. Poovendran. Cross-layer design for energy-efficient secure multicast communications in ad hoc networks. In *Proc. IEEE Int. Conf. Communications*, pages 3633–3639, 2004.

[15] Pangfeng Liu. Broadcast scheduling optimization for heterogeneous cluster systems. *J. Algorithms*, 42(1):135–152, 2002.

[16] Suvo Mittra. Iolus: A framework for scalable secure multicasting. In *SIGCOMM*, pages 277–288, 1997.

[17] Radha Poovendran and John S. Baras. An information-theoretic approach for design and analysis of rooted-tree-based multicast key management schemes. *IEEE Transactions on Information Theory*, 47(7):2824–2834, 2001.

[18] Javier Salido, Loukas Lazos, and Radha Poovendran. Energy and bandwidth-efficient key distribution in wireless ad hoc networks: a cross-layer approach. *IEEE/ACM Trans. Netw.*, 15(6):1527–1540, 2007.

[19] Clay Shields and J. J. Garcia-Luna-Aceves. KHIP—a scalable protocol for secure multicast routing. In *Proceedings of SIGCOMM*, pages 53–64, New York, NY, USA, 1999. ACM.

[20] Jack Snoeyink, Subhash Suri, and George Varghese. A lower bound for multicast key distribution. In *Proceedings IEEE Infocomm 2001*, 2001.

[21] D. Wallner, E. Harder, and R. Agee. Key management for multicast: Issues and architectures. Internet RFC 2627, 1999.

[22] Chung Kei Wong, Mohamed G. Gouda, and Simon S. Lam. Secure group communications using key graphs. In *Proceedings of SIGCOMM*, pages 68–79, 1998.

A Improved approximation for the routing tree case when weights are uniform

For the case where group members have the same key update probability and the communication network is a tree, using the same algorithm we can show the approximation ratio is 4.2 by a different analysis, shown as follows.

Claim 1. *Balanced partition node can always be found if the members have the same key update weight.*

Proof. Suppose this kind of partition node doesn't exist, which means for all internal node v, its number of leaves is either $< n/3$ or $> 2n/3$. We call nodes with less than $n/3$ leaves small nodes, and nodes with more than $2n/3$ leaves large nodes. Consider the large node with only small nodes as its children, there must be a combination of its children whose total number of leaves is between $n/3$ and $2n/3$. This means this kind of partition node exists. □

Lemma 4. $ALG(S) \leq ALG(X) + ALG(Y) + \frac{4\Delta}{3\log 3/2} n \log n + n\,(M(X) + M(Y))$.

Proof: (1) The cost of nodes in ALG(Y) is the same as their cost in ALG(S). (2) Similarly, the cost of nodes in ALG(X) is equal to their cost in ALG(S) + $\frac{4\Delta}{3\log 3/2} n\log n$. The reason we add $\frac{4\Delta}{3\log 3/2} n\log n$ is the multicast cost of each node in ALG(X) increased by Δ compared to its cost in ALG(X). Since in the worst case ALG(X) has $\log_{\frac{3}{2}} \frac{n}{3}$ levels, the increased cost is at most $2|X|\Delta \log_{\frac{3}{2}} \frac{n}{3} \leq 2\frac{2n}{3}\Delta \log_{\frac{3}{2}} \frac{n}{3} \leq \frac{4\Delta}{3\log 3/2} n\log n$. Combine (1) and (2), then add the cost of the root of ALG(X) and ALG(Y), we know this lemma is correct. □

Lemma 5. *OPT*(S) $\geq$ *OPT*(X) + *OPT*(Y) + $\frac{3\Delta}{\log 3} n\log n$

Proof: To any subset of X, the multicast cost calculated in OPT(S) is Δ more than the cost calculated in OPT(X). From Theorem 1, we know the increased cost is at least $3\Delta \cdot n\log_3 n = \frac{3\Delta}{\log 3} n\log n$. □

Theorem 6. *This is a 4.2-approximation algorithm.*

Proof:

$$
\begin{aligned}
\text{ALG}(S) &\leq \text{ALG}(X) + \text{ALG}(Y) + \frac{4\Delta}{3\log 3/2} n\log n + n\left(M(X) + M(Y)\right) \\
&\leq \alpha\cdot\text{OPT}(X) + \beta\cdot|X|M(X) + \alpha\cdot\text{OPT}(Y) + \beta\cdot|Y|M(Y) + \frac{4\Delta}{3\log 3/2} n\log n + n\left(M(X) + M(Y)\right) \\
&\leq \alpha\left[\text{OPT}(X) + \text{OPT}(Y)\right] + \frac{4\Delta}{3\log 3/2} n\log n + \left(\frac{2}{3}\beta + 1\right) n\left(M(X) + M(Y)\right) \\
&\leq \alpha\left[\text{OPT}(S) - \frac{3\Delta}{\log 3} n\log n\right] + \frac{4\Delta}{3\log 3/2} n\log n + \left(\frac{2}{3}\beta + 1\right) n\left(M(X) + M(Y)\right) \\
&\leq \alpha\left[\text{OPT}(S) - \frac{3\Delta}{\log 3} n\log n\right] + \frac{4\Delta}{3\log 3/2} n\log n + \left(\frac{2}{3}\beta + 1\right) n\left(M(S) + \Delta\right) \\
&= \alpha\cdot\text{OPT}(S) + \left(\frac{2}{3}\beta + 1\right) nM(S) - \alpha\cdot\frac{3\Delta}{\log 3} n\log n + \frac{4\Delta}{3\log 3/2} n\log n \\
&\leq \alpha\cdot\text{OPT}(S) + \beta\cdot nM(S)
\end{aligned}
$$

as long as $\alpha \geq 1.2$ and $\beta \geq 3$. This means this is a 4.2-approximation. □