A Scalable Multicast Key Management Scheme for Heterogenous Wireless Networks [*]

Yan Sun, Wade Trappe[†] and K.J. Ray Liu
Department of Electrical and Computer Engineering
and the Institute for System Research
University of Maryland, College Park, MD 20742
ysun, kjrliu@glue.umd.edu

Abstract

In the near future, many multicast services will involve wireless devices. Before these services can successfully be deployed, a security infrastructure that provides access control to the content must be developed. In wireless networks, where the error rate is high and the bandwidth is limited, the design of key management schemes should place emphasis on reducing the communication burden associated with key updating. A communication-efficient class of key management schemes are those that employ a tree hierarchy. However, these tree-based key management schemes do not exploit issues related to the delivery of keying information that provide opportunities to further reduce the communication burden of rekeying. In this paper, we propose a method for designing multicast key management trees that match the network topology associated with mobile wireless applications. By matching the key management tree to the cellular network topology and localizing the transmission of keying information, the proposed key management scheme significantly reduces the communication burden of rekeying. Further, in the wireless scenario, the issue of user handoff between base stations may cause user relocation on the key management tree, which introduces extra rekeying overhead. We address the problem of user handoff by proposing an efficient handoff scheme for our topology-matching key management trees. We then provide a procedure to generate a topology-matching key management tree that addresses the heterogeneity of the network. For multicast applications containing several thousands of users, simulations indicate a 55-80% reduction in the communication cost compared to trees that are independent of the network topology. Analysis and simulations also show that the communication cost of the proposed topology-matching key management tree scales better than topology-independent trees as the size of multicast group grows.

[*]This work is partially supported by the U.S. Army Research Office under Award No. DAAD190110494.

[†]The author is currently at the Wireless Information Network Laboratory (WINLAB) at Rutgers University, and may be reached by email at trappe@winlab.rutgers.edu.

1 Introduction

The rapid progress in the technologies underlying multicast networking have led to the deployment of many multicast services, such as streaming stock quotes and multimedia services [1]. At the same time, there has been significant advancements in building a global wireless infrastructure that will free users from the confines of static communication networks. Users will be able to access the Internet from anywhere at anytime. As wireless connections become ubiquitous, consumers will desire to have multicast applications running on their mobile devices. In order to meet such a demand, there has been increasing research efforts in the area of wireless multicast [2–4].

Many multicast applications require *access control* mechanisms to guarantee that only authorized users can access the multicast content. Access control is achieved by encrypting the content using an encryption key, known as the session key (SK), that is shared by all legitimate group members. Since users may join and leave at anytime, it is necessary to change the SK in order to prevent the leaving user from accessing future communication and prevent the joining user from accessing previous communication [5–14].

In a typical multicast key management scheme, a trusted third party, known as the key distribution center (KDC), is responsible for securely communicating new key material to the group members. In cellular networks, the KDC may either be the service provider or a trusted third party connected to the network. In order to accomplish key distribution, the KDC shares auxiliary keys, known as key encrypting keys (KEKs), which are used solely for the purpose of updating the session key and other KEKs. In addition, each user has a private key that is only known by himself and the KDC. A popular class of multicast key management schemes employ a tree hierarchy for the maintenance of keying material [7–10].

Rekeying messages used to update key information are sent to group members when there are users joining or leaving the multicast group. For many key management schemes, such as tree-based schemes, the amount of rekeying messages per join/leave increases linearly with logarithm of the group size [6–9, 15]. In applications where there are many users and frequent additions or deletions to the group membership, even such scalable key management schemes can introduce a significant communication burden. Additionally, rekeying messages must be delivered reliably because the loss of rekeying messages results in severe performance degradation [6]. If a user loses one key, he will not be able to access multicast content encrypted by this key and may not be able to acquire future keys from future rekeying messages either. Further, in real-time multicast applications the rekeying messages should also be delivered in a timely manner so that users receive the rekeying messages before the new key takes effect. These reasons alone motivate the need for building communication-efficient key management schemes. In wireless multicast scenarios, however, the need is even more pronounced since the bandwidth is limited and data typically experience a higher transmission error rate than in conventional environments.

In this paper, we propose a method for designing the multicast key management tree for a group of users in a cellular network. Traditional tree-based multicast key management schemes do not consider the effect of the network topology upon the delivery of the rekeying messages, and therefore waste network resources by sending rekeying messages to users who do not need them. We address this issue by proposing to match the key management tree to the network topology, thereby localizing the delivery of the rekeying messages and reducing the communication costs. In Section 2, we introduce the concept of matching the key tree to the network topology and motivate the reduction in the communication cost associated with rekeying. In mobile environments, the user will subscribe to a multicast service under an initial host agent, and through the course of his service undergo *handoff* to different base stations. In Section 3, we discuss issues arising from user relocation and present a handoff scheme that is suitable for topology-matching key management. In Section 4, we analyze the effect that matching the key tree to the topology has upon the communication overhead. We then address the complexity of designing the key management tree in Section 5 by proving that optimizing the proposed key tree is equivalent to optimizing a set of independent smaller-scale subtrees. This significantly reduces the complexity of the tree design. We describe, in Section 6, a tree structure that can easily adapt to changes in the number of users and a tree generation algorithm that considers the heterogeneity of the network. We then describe a procedure to build the key tree and determine the parameters that optimize the tree. Finally, simulation results are presented in Section 7 and conclusions are drawn in Section 8.

2 Topology-Matching Key Management Tree

In this section, we introduce the benefits of matching the key tree to the network topology. We outline a procedure to design the key management tree and define the cost functions that we use in the remainder of the paper for measuring the communication burden associated with key updating.

The most common class of multicast key management schemes employ a tree hierarchy of KEKs [6–10], as depicted in Figure 1. Each user stores his private key u_i, the session key K_s, and a set of KEKs on the path from himself to the root of the key tree. Since the size of the rekeying messages sent for a member join operation is much less than that for member departure [7, 10], we shall only focus on the communication cost of member departure. For example, referring to Figure 1, when user 16 leaves the multicast service, all of his keys, $\{u_{16}, K_s, K_\epsilon, K_1, K_{11}, K_{111}\}$, should be updated. Let x^{old} denote the old version of key x, x^{new} denote the new version of key x, and $\{y\}_x$ denote the key y encrypted by key x. Then, the key updating can be achieved by sending the following rekeying messages:

1. $\{K_{111}^{new}\}_{u_{15}}$ is sent and user 15 acquires K_{111}^{new}.
2. $\{K_{11}^{new}\}_{K_{111}^{new}}$ and $\{K_{11}^{new}\}_{K_{110}^{old}}$ are sent and users 13, 14, 15 acquire K_{11}^{new}.
3. $\{K_{1}^{new}\}_{K_{11}^{new}}$ and $\{K_{1}^{new}\}_{K_{10}^{old}}$ are sent and users $9, 10, \cdots, 15$ acquire K_{1}^{new}.
4. $\{K_{\varepsilon}^{new}\}_{K_{1}^{new}}$ and $\{K_{\epsilon}^{new}\}_{K_{0}^{old}}$ are sent and users $1, 2, \cdots, 15$ acquire K_{ϵ}^{new}.
5. $\{K_{s}^{new}\}_{K_{\epsilon}^{new}}$ is sent and users $1, 2, \cdots, 15$ acquire the new session key K_{s}^{new}.

The above key updating procedure achieves key updating without leaking new key information to the leaving user. This example also illustrates that most rekeying messages are only useful to a subset of users, who are always neighbors on the key management tree. In fact, the first rekeying message is only useful to user 15, the second rekeying message is only useful to users 13, 14, 15, the third rekeying message is useful to users $9, 10, \cdots, 15$, and the fourth and fifth rekeying messages are useful to all users. Therefore, rekeying messages do not have to be sent to every user in the multicast group.

We propose to exploit this observation in designing a key management tree. Our key management tree will match the network topology in such a way that the neighbors on the key tree are also physical neighbors on the network. By delivering the rekeying messages only to the users who need them, we may take advantage of the fact that the key tree matches the network topology, and localize the delivery of rekeying messages to small regions of the network. This lessens the amount of traffic crossing portions of the network that do not have users who need to be rekeyed. In order to accomplish this, it is necessary to have the assistance of entities that would control the rekeying message transmission, such as the base stations in cellular wireless networks.

A cellular network model, as depicted in Figure 2 and proposed in [16], consists of mobile users, base stations (BS) and supervisor hosts (SH). The SHs administrate the BSs and handle most of the routing and protocol details for mobile users. The service provider, the SHs, and the BSs are connected through high-speed wired connections, while the BSs and the mobile users are connected through wireless channels. In this paper, the SHs can represent any entity that administers BSs, such as the region servers presented in [17] or the radio network controllers (RNCs) in 3G networks [18]. In cellular wireless networks, multicast communication can be implemented efficiently by exploiting the inherent broadcasting nature of the wireless media [19–21]. In this case, multicast data is first routed to the BSs using multicast routing techniques designed for wireline networks [1], and then broadcast by the BSs to mobile users.

If we assume that both the SHs and the BSs can determine whether the rekeying messages are useful for the users under them, then the cellular wireless network has the capability of sending messages to a subset of users. In particular, the SHs multicast a rekeying message to their BSs if and only if the message is useful to one or several of their BSs, and the BSs broadcast the rekeying message to their users if and only if the message is useful to the users under them. The information needed to identify whether a SH or BS needs a rekeying message can be sent in the rekeying message header. We shall not consider the size of this overhead information in our calculation since this overhead is typically small compared to the size of the actual rekeying messages, and is implementation-dependent. Hence, when the key tree matches the network topology, we can localize the delivery of rekeying messages.

We design a key management tree that matches the network topology in three steps:

- Step 1: Design a subtree for the users under each BS. These subtrees are referred to as *user subtrees*.

- Step 2: Design subtrees that govern the key hierarchy between the BSs and each SH. These subtrees are referred to as *BS subtrees*.

- Step 3: Design a subtree that governs the key hierarchy between the SHs and the KDC. This subtree is referred to as the *SH subtree*.

The combined key management tree is called a Topology-Matching Key Management (TMKM) tree. Figure 3 illustrates a TMKM tree for the network topology shown in Figure 2. Traditional key management trees, such as those in [7–10], are independent of the network topology, and we call them Topology Independent Key Management (TIKM) trees. When using a TIKM tree, the users are scattered all over the network, and therefore it is not possible to localize the delivery of rekeying messages.

We study the communication burden of the rekeying messages in the wired portion and in the wireless portion of the network separately. Under each SH, we introduce two costs: the *wireline-message-size*, which is defined as the total size of the rekeying messages multicast by the SHs to the BSs; and the *wireless-message-size*, which is defined as the total size of the rekeying messages broadcast by the BSs. The message size is measured in units whose bit length is the same size as the key length. We note that the wireline message size does not capture all of the wired cost in the network, as there may be additional wired costs between the SHs and the KDC. This was done since it is not easy to consider a single formulation for the costs between the SHs and the KDC that captures the many possible cellular architectures between the KDC and the SHs. On the other hand, it is possible to use a single formulation to capture the behavior between the SH and the BS since most cellular systems model the SH-BS connection using a single link, such as the Iub in 3G cellular systems. Further, if we assume that the network connections between the KDC and the SHs are trivial, that is the KDC is placed at the SHs, or if the network connections between the KDC and the SHs have ample enough bandwidth resources, then the KDC-SH cost may be considered negligible compared to the SH-BS costs. Therefore, we have made a simplification to consider only the SH-BS wired cost and do not consider KDC-SH wired costs in this paper.

Let S_1^l denote the wireline-message-size under the l^{th} SH and S_2^l denote the wireless-message-size under the l^{th} SH, where $l = 1, 2, \cdots, n_{sh}$ and n_{sh} is the total number of SHs. For example, when the length of the session key and KEKs is 128 bits each, if a 256 bit long rekeying message is multicast by the l^{th} SH and then broadcast by 3 BSs under the l^{th} SH, then $S_1^l = 2$ and $S_2^l = 6$. Assuming that users do not leave simultaneously, then the rekeying wireline cost, C_{wire}, the rekeying wireless cost $C_{wireless}$, and the total rekeying cost C_T, are defined as:

$$\begin{aligned} C_{wire} &= \sum_{l=1}^{n_{sh}} \alpha_1^l E[S_1^l] \ ; \quad C_{wireless} = \sum_{l=1}^{n_{sh}} \alpha_2^l E[S_2^l] \\ C_T &= \gamma \cdot C_{wireless} + (1-\gamma) \cdot C_{wire}, \end{aligned} \tag{1}$$

where $E[.]$ indicates expectation over the statistics governing the user joining and leaving behavior. Here, $0 \le \gamma \le 1$ is the *wireless weight*, which represents the importance of considering the wireless cost, and $\{\alpha_1^l\}$ and $\{\alpha_2^l\}$ are the sets of weight factors that describe the importance of considering the wireline-message-size and wireless-messages-size under the l^{th} SH respectively. When SHs administrate areas with similar physical network structure and channel conditions, we can approximate $\{\alpha_1^l\}$ and $\{\alpha_2^l\}$ by 1. In addition, we define the *combined-message-size* as $S_T^l = \gamma \cdot S_2^l \alpha_2^l + (1-\gamma) \cdot S_1^l \alpha_1^l$. Thus, C_T can also be expressed as $C_T = \sum_{l=1}^{n_{sh}} E[S_T^l]$.

For a given wireless weight γ, $\{\alpha_1^l\}$, and $\{\alpha_2^l\}$, both the TMKM and TIKM trees should be designed to minimize the total communication cost, C_T.

3 Handoff Schemes for TMKM Tree

In mobile environments, the user will subscribe to a multicast service under an initial host agent, and through the course of his service move to different cells and undergo *handoff* to different base stations. Although the user has moved, he still maintains his subscription to the multicast group. Since the TMKM tree depends on the network topology, the physical location of a user affects the user's position on the key management tree. When a user moves from one cell to another cell, the user needs to be relocated on the TMKM tree. In this section, we propose an efficient handoff scheme for our TMKM trees. In this context, the expression *handoff scheme* will only refer to the process of relocating a user on the key tree.

One solution to the handoff problem is to treat the moving user as if he departs the service from the cell that he is leaving from and then rejoins the service in the cell that he has moved to. This scheme, referred to as the *simple handoff scheme*, is not practical for mobile networks with frequent handoffs since rekeying messages are sent whenever handoffs occur.

During handoff, if a user remains subscribed to the multicast group, it is not necessary to remove the user from the cell where he previously stayed. Allowing a mobile user to have more than one set of valid keys while he stays in the service does not compromise the requirements of access control, as long as all of the keys that he possesses are updated when he finally leaves the service. In order to trace both the users' handoff behavior and the key updating process, we employ a wait-to-be-removed (WTBR) list for each cell. The WTBR list of cell i, denoted by $WTBR_i$, contains the users who (1) possess a set of valid keys on the user subtree of cell i and (2) are currently in the service but not in cell i. These WTBR lists are maintained by the KDC.

Let t^i_{update} denote the time of the last key update that occurs due to a departure occurring in cell i, and let t^u_{join} denote the time when the user u first joins the service. In addition, we define $keyset^u_i$ to be the set of keys possessed by the user u while he is in cell i. We propose an *efficient handoff scheme* that is illustrated in Figure 4, as:

- When user u moves from cell i to cell j,

  1. Put u on the WTBR list of cell i, i.e. $WTBR_i$, and remove him from the user subtree of cell i.
  2. If u has been in cell j before and is on $WTBR_j$, put u back on the branch of the subtree that he previously belonged to and remove him from $WTBR_j$. If u is not on $WTBR_j$, put u on the most recently updated branch of the user subtree of cell j. We note that the set of keys associated with u's new position, $keyset^u_j$, was updated at time t^j_{update}.
  3. If $t^u_{join} > t^j_{update}$, the keys in $keyset^u_j$ are updated using the procedure for user join described in [10]. If $t^u_{join} \leq t^j_{update}$, the keys do not need to be updated.
  4. The keys in $keyset^u_j$ are sent to u through unicast.

The purpose of step 3 is to prevent u from taking advantage of the handoff process to access the communication that occurred before he joined. To see this, let u join the service at $t^u_{join} = t_0$ in cell i, and then immediately move to cell j. After relocation, user u obtains keys in $keyset^u_j$ that is updated at time $t^j_{update} = t_0 - \Delta$, where Δ is a positive number. In this case, if we do not update the keys in $keyset^u_j$ and u has recorded the communication in cell

j before joining, u will be able to decrypt the multicast content transmitted in $[t_0 - \Delta, t_0)$, during which time he is not a valid group member.

- When user u leaves the multicast service from cell j:

  1. The keys that are processed by u and still valid should be updated. In particular, the keys in $keyset_j^u$ and $\{keyset_i^u : WTBR_i \text{ contains } u\}$ are updated using the procedure for user departure in [10].
  2. Check other users on the WTBR lists that contain u. If u and another user u^* are both on $WTBR_i$, and $keyset_i^u = keyset_i^{u^*}$, remove u^* from $WTBR_i$. It is noted that u^* is removed from $WTBR_i$ when u^* does not have valid keys associated with cell i any more. Step 2 does not require extra rekeying messages.
  3. Remove u from all WTBR lists.

Thus, a user will be removed from the WTBR lists not only when he leaves the service, but also when other users who share the same keys leave the service. Compared with the simple handoff scheme, the efficient handoff scheme can reduce the key updating caused by user relocation because the number of cells that need to update keys is smaller than the number of cells that a user has ever visited.

When the key tree matches the network topology, handoffs result in user relocation on the key tree, which inevitably introduces extra cost to the task of key management. In this work, we assume that the KDC has significant computation and storage resources and do not investigate the cost for the KDC to maintain and update the WTBR lists. We will focus on the extra communication cost due to the fact that more than one set of keys may need to be updated for a departing user when handoffs exist.

4 Analysis of the TMKM Tree

Matching the key management tree with the network topology has two contrasting effects on the rekeying message communication cost. First, the cost of sending one rekeying message is reduced because only a subset of the BSs broadcast the message. Second, the number of rekeying messages may increase due to handoffs. In this section, we analyze these two effects and investigate the influence that user mobility and the wireless weight have upon the performance of the TMKM scheme.

To simplify the analysis, we assume that the system has a^{L_0} SHs, each SH administrates a^{L_1} BSs, and each BS has a^{L_2} users, where $a \geq 2$, L_0, L_1 and L_2 are positive integers. We also assume that the SHs administer areas with similar network structure and conditions. Therefore, $\{\alpha_1^l\}$ and $\{\alpha_2^l\}$ are approximated by 1. The user subtrees, BS subtrees, and SH subtree are designed as balanced trees with degree a and level L_2, L_1, and L_0, respectively. For fair comparison, the TIKM tree is also designed as an *a-ary* balanced tree with $(L_0+L_1+L_2)$ levels. In this work, the level of a tree is defined as the maximum number of nodes on the path from a leaf node to the root excluding the leaf node. Since the SHs are usually in charge of large areas, the probability of a user moving between SHs during a multicast service is much smaller than the probability of handoffs that are under one SH. In this analysis, we assume that there are no SH level handoffs. For the present computation, we only calculate the communication cost caused by *one* departing user based on the rekeying procedure described in [6, 7, 10].

As illustrated by the example in Section 2, rekeying messages of size $(a \cdot L)$ need to be transmitted when one user leaves from a balanced key tree with degree a and level L. When using the TIKM tree, rekeying messages of size $a(L_0 + L_1 + L_2)$ are transmitted under a^{L_0} SHs and broadcast by $a^{L_0+L_1}$ BSs. Therefore, when one user leaves the service, the wireline-message-size, denoted by $\tilde{C}_w^{tikm}$, and the wireless-message-size, denoted by $\tilde{C}_{wl}^{tikm}$, are computed as:

$$\tilde{C}_w^{tikm} = (aL_0 + aL_1 + aL_2)a^{L_0} \tag{2}$$

$$\tilde{C}_{wl}^{tikm} = (aL_0 + aL_1 + aL_2)a^{L_0+L_1}. \tag{3}$$

The performance of the TMKM tree is affected by the user handoff behavior. We define the random variable I as the number of WTBR lists that contain the departing member when he leaves the service. We also introduce the function $B(b, i, a)$ that describes the number of intermediate KEKs that need to be updated. $B(b, i, a)$ is equivalent to the expected number of occupied boxes when putting i items in b boxes with repetition, where each box can have at most a items. A box is called occupied when one or more items are put into the box. The detailed calculation of $B(b, i, a)$ is given in Appendix A.

When one user leaves the service and he is on $I = i$ WTBR lists, we can show that:

- We need to update $(i \cdot L_2)$ keys on the user subtrees. Thus, rekeying messages with a total size $(iaL_2 - 1)$ are transmitted under one SH and broadcast by a single BS.

- We need to update $B(a^{L_1-m}, i, a^m)$ KEKs on the level $(L_1 - m)$ of the BS subtree. Thus, messages with size $aB(a^{L_1-m}, i, a^m)$ are transmitted under one SH and broadcast by a^m BSs. Here, $m = 1, \cdots, L_1$, and the level 0 of a tree is just the root.

- We need to update (a^t) KEKs on the level $(L_0 - t)$ of the SH subtree. Thus, messages with size (a^{t+1}) are sent under (a^t) SHs and broadcast by $(a^{L_1} \cdot a^t)$ BSs. Here, $t = 1, 2, \cdots, L_0$.

- In addition, we need one message to update the session key K_s. This message is sent to all a^{L_0} SHs and $a^{L_0+L_1}$ BSs.

Therefore, when the departing user belongs to i WTBR lists, the expected value of the wireline-message-size, denoted by $C_w^{tmkm}(i)$, and the expected value of the wireless-message-size, denoted by $C_{wl}^{tmkm}(i)$, are computed as:

$$C_w^{tmkm}(i) = iaL_2 + \sum_{m=1}^{L_1} aB(a^{L_1-m}, i, a^m) + \sum_{t=1}^{L_0} a^{t+1} \tag{4}$$

$$C_{wl}^{tmkm}(i) = iaL_2 - 1 + \sum_{m=1}^{L_1} a^{m+1}B(a^{L_1-m}, i, a^m) + a^{L_1}\sum_{t=1}^{L_0} a^{t+1} + a^{L_0+L_1}. \tag{5}$$

The performance of the TIKM tree and the TMKM tree can be compared by examining the values of $\tilde{C}_w^{tikm}$ and $C_w^{tmkm}(i)$, $\tilde{C}_{wl}^{tikm}$ and $C_{wl}^{tmkm}(i)$. In Figure 5, these values are plotted for different i and L_0, when the other parameters are fixed as $a = 2$, $L_1 = 3$, and $L_2 = 6$. Since the TIKM tree is not affected by handoffs, $\tilde{C}_w^{tikm}$ and $\tilde{C}_w^{tikm}$ are constant. Figure 5(a) and Figure 5(b) show the wireline-message-size and wireless-message-size respectively, when the system has only one SH. Figure 5(c) and Figure 5(d) show the corresponding curves for 2 SHs, while Figure 5(e) and Figure 5(f) depict the corresponding curves for systems with 8 SHs. We observe that:

- Both $C_w^{tmkm}(i)$ and $C_{wl}^{tmkm}(i)$ are increasing functions of i.

- The TMKM tree always reduces the wireless-message-size, and this advantage becomes larger when the system contains more SHs.

- For systems containing only one SH, i.e. $L_0 = 0$, the TMKM trees introduce larger wireline-message-sizes than TIKM trees due to the handoff effects. When there are multiple SHs, the TMKM scheme can take advantage of the fact that some SHs do not need to transmit rekeying messages to their BSs, and can reduce the wireline-message-size when i is small. It should be noted that the wireline cost will be larger than that given in (4) if there are SH-level handoffs.

Since TMKM trees reduce the wireless-message-size more effectively than reducing the wireline message size, a larger wireless weight γ leads to an improved advantage of TMKM trees over TIKM trees. Using large γ is reasonable since the wireless portion of the network usually experiences a higher error rate and has less available bandwidth when compared to the wireline portion, which makes the wireless cost the major concern in many realistic systems. In addition, the communication cost of the TMKM tree increases with the number of cells that need to update keys when a user leaves. Therefore, when handoffs are less likely to happen, the TMKM tree has larger advantage over the TIKM tree.

Scalability is another important performance measure of key management schemes [6]. We define $N = a^{L_0}$ as the number of SHs. When $N \to \infty$, the scalability properties can be easily obtained from (2)-(5), and are summarized in Table 1. Both Figure 5 and Table 1 demonstrate that the communication cost of TMKM trees scales better than that of TIKM trees when more SHs participate in the multicast service.

5 Separability of the Optimization Problem

The TMKM tree consists of user-subtrees, BS-subtrees, and SH-subtrees. In this section, we show that optimizing the entire TMKM tree is equivalent to optimizing those subtrees individually. This is desirable since optimizing the subtrees separately reduces the dimension of the search space for optimal tree parameters and significantly reduces the complexity of tree design.

In this work, we assume that the users under the same SH have the same joining, departure and mobility behavior. Thus, the user subtrees under the same SH have the same structure. It is easy to verify that the main results in this section still hold in scenarios where the dynamic behavior of the users varies under different BSs. However, for the discussion in this paper, we will restrict our attention to the case where the dynamic behavior of the users between different BSs is identical. In addition, we assume that the number of participating SHs and BSs do not change during the multicast service. In order to make the presentation more concise, we introduce the notation $D_{k,l}$ to represent the situation where k users are under the l^{th} SH and one of these users leaves the service.

As discussed in Section 2, the total communication cost, C_T, is expressed as

$$C_T = \sum_{l=1}^{n_{sh}} E[S_T^l]. \tag{6}$$

Based on the definition of S_T^l, one can see that

$$E[S_T^l] = \sum_k p^l(k) G^l(k) E^l(k), \tag{7}$$

where

$p^l(k)$:	pmf of the number of users under the l^{th} SH,
$G^l(k)$:	probability that a user leaves from the l^{th} SH given that k users are under the l^{th} SH,
$E^l(k)$:	the expected value of the combined-message-size given the condition $D_{k,l}$.

When a user leaves, the keys that need to be updated are divided into three categories: (1) the keys on the user subtrees, (2) the keys on the BS subtrees, and (3) the keys on the SH subtree. Under the condition $D_{k,l}$, let $A_1^l(k)$, $A_2^l(k)$ and A_3^l denote the expected value of the combined-message-size under the l^{th} SH resulting from updating the keys on the user-subtrees, BS-subtrees and SH-subtrees, respectively. We note that A_3^l is not a function of k when there are no SH-level handoffs, and that $E^l(k) = A_1^l(k) + A_2^l(k) + A_3^l$. Then, (6) becomes

$$C_T = \sum_{l=1}^{n_{sh}} \left(\sum_k p^l(k) G^l(k) A_1^l(k) \right) + \sum_{l=1}^{n_{sh}} \left(\sum_k p^l(k) G^l(k) A_2^l(k) \right) + \sum_{l=1}^{n_{sh}} A_3^l \cdot \left(\sum_k p^l(k) G^l(k) \right). \tag{8}$$

We observe that the structure of the user-subtrees only affects $A_1^l(k)$, the structure of the BS-subtrees only affects $A_2^l(k)$, and the structure of the SH-subtrees only affects A_3^l. Therefore, for the TMKM tree, the user-subtrees, BS-subtrees and SH subtree can be designed and optimized separately. Particularly, the user-subtrees under the l^{th} SH should be designed to minimize $\sum_k p^l(k) G^l(k) A_1^l(k)$, the BS subtree under the l^{th} SH should be designed to minimize $\sum_k p^l(k) G^l(k) A_2^l(k)$, and the SH subtree should be designed to minimize $\sum_{l=1}^{n_{sh}} A_3^l \cdot \left(\sum_k p^l(k) G^l(k) \right)$.

6 Design of the TMKM Tree

Key management schemes are closely related to the *key management architecture*, which describes the entities in the network that perform key management [6]. In cellular wireless networks, the BSs are not trusted to perform key management because they can be easily tampered with [16]. The SHs are able to perform key management if they are trusted and have the necessary computation and storage capabilities. The trustiness of the SHs depends on both the business model and the protection on the SHs. Based on whether SHs perform key management, the systems can be classified into two categories:

- In the first category, each SH performs key management for a subset of the group members who reside in the region where this SH is in charge. Each SH can be looked at as a local key distribution center (KDC). Without loss of generality, since the SHs are independent and may even adopt different key management schemes, we can study systems containing only one SH, which we shall refer to as *one-SH systems.*

- In the second category, SHs do not perform key management. Instead, there is a single KDC that manages keys for all of the users. This KDC can be the service provider or a trusted third party. The systems containing many SHs are referred to as *multiple-SH systems.*

In one-SH systems, the TMKM tree consists of user-subtrees and a BS subtree. In multiple-SH systems, the TMKM tree consists of user-subtrees, BS-subtrees and a SH subtree.

In this section, we introduce a model describing the joining and leaving behavior of the users, and a flexible tree structure that can be used to design the user and BS subtrees. We then examine the optimization of the user and BS subtrees and the design of the SH subtree.

6.1 Dynamic membership model

Mlisten [22] is a tool that can collect the join/leave times for multicast group members in MBone sessions. Using this tool, [23, 24] studied the characteristics of the membership dynamics of MBone multicast sessions and showed that the user arrival process can be modeled as Poisson and the membership duration of short sessions (that usually last several hours) is accurately modeled using an exponential distribution while the membership duration of long sessions (that usually last several days) is accurately modeled using the Zipf distribution [25]. Based on the population model of short MBone sessions, we made the following assumptions on the membership dynamics:

1. Under the l^{th} SH, the user's arrival process is Poisson with rate λ_l and the service duration is governed by an exponential random variable with mean $1/\mu_l$, where $l = 1, 2, \cdots, n_{sh}$.

2. A user's joining and leaving behavior is independent of other users.

Based on the first assumption, the number of users under the l^{th} SH is a Poisson random variable with rate θ_l, i.e. $p^l(k) = \frac{\theta_l^k}{k!}e^{-\theta_l}$, where $\theta_l = \lambda_l/\mu_l$ [26]. In addition, it can be shown that $G^l(k)$ approximately equals to $k \cdot \mu_l$. It is noted that the second assumption is reasonable in some types of multicast services, such as periodic news multicast, while it may not be correct for services such as a scheduled pay-per-view multicast, where different users are related with each other through watching the same content.

In this work, we use this Poisson arrival and exponential service duration model to optimize the TMKM tree. In Section 7, we will use simulations to demonstrate that the performance of the TMKM tree is not sensitive to users' statistical membership models.

6.2 ALX tree structure

The TMKM scheme matches the key tree to the network topology by decomposing the key tree into user subtrees, BS subtrees, and SH subtrees. The TMKM scheme does not have constraints on the specific structure of these subtrees. In this section, we propose a tree structure that is capable of handling membership additions, deletions, or relocations with minimal changes to the tree's structure.

As illustrated in Figure 6 and parameterized by the triple $(a, L, \mathbf{x})$, this $(a, L, \mathbf{x})$-logical tree has $L+1$ levels. The upper L levels, which comprise a full balanced subtree with degree a, are fixed during the multicast service. The users are represented by the leaf nodes on the $(L+1)^{st}$ level. We use a vector $\mathbf{x}$ to describe the $(L+1)^{st}$ level, where x_i is the number of users attached to the i^{th} node of the L^{th} level, and $i = 1, 2, \cdots, a^L$. In the example shown in Figure 6, $\mathbf{x} = [4, 2, 3, 3, 2, 4, 3, 3, 3]$, $a = 3$ and $L = 3$. We will refer to this tree structure as the ALX tree.

When using the ALX tree, the joining user is always put on the branch with the smallest value of x_i. The maximum number of users on an ALX tree is not restricted. When a user leaves, the average rekeying message size is $(\frac{k}{a^L} - 1 + aL)$, where k is the number of users on the ALX tree. When the user's arrival process is Poisson with rate λ, and the service time is an exponential random variable with mean $1/\mu$, the probability that a user leaves the key tree is approximately $k \cdot \mu$, and the pmf of k is $p(k) = \frac{\theta^k}{k!}e^{-\theta}$, where $\theta = \lambda/\mu$. The performance of the ALX tree is evaluated by the expected value of the rekeying message size, denoted by C_{alx}, and is calculated as

$$C_{alx} = \sum_{k=1}^{\infty} p(k) \cdot k \cdot \mu \cdot (\frac{k}{a^L} - 1 + aL). \tag{9}$$

It follows that the optimization problem of the ALX tree can be formulated as:

$$\tilde{C}_{alx} = \min_{a>1, L>0} C_{alx}. \tag{10}$$

Balanced trees whose degree is pre-determined, such as binary and trinary trees, are widely used to design key trees [6,10]. Next, we compare the ALX tree structure with balanced trees that have a pre-defined degree, which we refer to as fixed-degree trees in this section.

Adding or removing a user from balanced fixed-degree trees often requires splitting or merging nodes. For example, when a new user is added to the key tree shown in Figure 1, one leaf node must be split to accommodate the joining user. In this case, a new KEK is created and must be transmitted to at least one existing user. When using the ALX tree structure, however, no new KEKs are created during membership changes. We know that updating existing KEKs for user join can be achieved without sending any rekeying messages, as suggested in [10], because existing users can update KEKs using one-way functions after being informed of the need to update their keys. Therefore, the ALX tree structure allows for a key updating operation that does not require sending any rekeying messages during user joins. In addition, the ALX tree introduces minimal change to the tree structure with dynamic membership and therefore is easy to implement and analyze.

Further, the ALX tree should be optimized over the distribution of the group size. If we take individual snapshots of the system when the group size is very small or large, the ALX tree may not perform as well as fixed degree trees that adjust themselves according to the group size. However, we will derive the performance lower bound for fixed degree trees and then demonstrate that the cost for ALX trees, $\tilde{C}_{alx}$, is in fact very close to this lower bound. Similar to (9), the expected rekeying message size when using a tree with fixed degree n, denoted by $C_{fix}(n)$, is calculated as:

$$C_{fix}(n) = \sum_{k=1}^{\infty} p(k) \cdot k \cdot \mu(n - 1 + n \cdot (P - 1)) \ ,$$

where P is the average length of branches for a tree with k leaves and degree n. It is well known that P equals the expected codeword length of a source code containing k symbols with equal probability. The bounds on P are known to be $\log_n(k) \leq P < \log_n(k) + 1$ [27]. Therefore,

$$C_{fix}(n) > \sum_{k=1}^{\infty} p(k) \cdot k \cdot \mu \cdot (n \log_n(k) - 1). \tag{11}$$

Based on (11), the performance lower bound for the fixed degree trees is given by

$$\tilde{C}_{fix} = \min_{n} \sum_{k=1}^{\infty} p(k) \cdot k \cdot \mu \cdot (n \log_n(k) - 1). \tag{12}$$

It is noted that no fixed degree trees can reach this lower bound. In fact, $\tilde{C}_{fix}$ would be achieved if and only if we could (1) reorganize the tree immediately after user join or departure in such a way that the rekeying message size for the next user join/leave operation is minimized; and (2) reorganize the tree without adding any extra communication cost. However, reorganizing trees, such as splitting or merging nodes, requires sending extra keying information to users. These above two conditions can never be achieved simultaneously.

The lower bound in (12) is used as a reference for evaluating the performance of the ALX tree. In Figure 7(a), $\tilde{C}_{fix}$ and $\tilde{C}_{alx}$ are compared for different user joining rates, λ. In Figure 7(b), $\tilde{C}_{fix}$ and $\tilde{C}_{alx}$ are compared for different average service durations, $1/\mu$. We observe that the relative difference between the lower bound and the performance of the ALX tree is less than 3.5%.

The ALX tree has the advantage of maintaining tree structure during user joins and departures, while its performance is very close to the lower bound of fixed degree trees. Although ALX tree is not the optimal solution amongst all possible tree structures, its practical nature makes the ALX tree an ideal candidate for designing the user and BS subtrees.

6.3 User Subtree Design

The user subtrees are designed as ALX trees. Under the l^{th} SH, the optimal tree parameters, a and L, solve

$$\min_{a,L} \sum_{k} p^l(k) G^l(k) A_1^l(k), \tag{13}$$

where a and L are positive integers and $G^l(k) \approx k\mu_l$. Let $T^u_w(k,i)$ and $T^u_{wl}(k,i)$ respectively represent the expected value of the wireline-message-size and wireless-message-size caused by updating keys on the user subtrees, given that k users are under the l^{th} SH, one of them leaves and he is on i WTBR lists. We can show that

$$T^u_w(k,i) = T^u_{wl}(k,i) = (\frac{k/n^l_{bs}}{a^L} - 1 + aL)i.$$

Then, $A^l_1(k)$ is computed as

$$\begin{aligned} A^l_1(k) &= \sum_{i=1}^{n^l_{bs}} p^l_h(i)(\alpha^l_2\gamma T^u_{wl}(k,i) + \alpha^l_1(1-\gamma)T^u_w(k,i)) \\ &= (\alpha^l_2\gamma + \alpha^l_1(1-\gamma))(\frac{k/n^l_{bs}}{a^L} - 1 + aL)E[I^l], \end{aligned} \tag{14}$$

where $E[I^l] = \sum_{i=1}^{n^l_{bs}} p^l_h(i)i$, and α^l_1 and α^l_2 are defined in Section 2. By substituting (14) into (13), the optimization problem for the user-subtrees under the l^{th} SH is

$$\min_{a,L} \sum_k k \cdot p^l(k) \cdot (\frac{k/n^l_{bs}}{a^L} - 1 + aL). \tag{15}$$

The optimum a and L can be obtained by searching the space of possible a and L values.

6.4 BS Subtree Design

We also design BS subtrees as ALX trees. We denote the degree and the level of a BS subtree by a_{bs} and L_{bs}, respectively. Let $T^b_w(k,i)$ and $T^b_{wl}(k,i)$ respectively denote the expected value of the wireline-message-size and wireless-message-size caused by key updating on the BS subtree under the l^{th} SH given the condition $D_{k,l}$ and the condition that the departing member is on i WTBR lists. We can show that:

$$T^b_w(k,i) = s \cdot B({a_{bs}}^{L_{bs}}, i, s) + \sum_{m=1}^{L_{bs}} a_{bs} \cdot B({a_{bs}}^{L_{bs}-m}, i, s \cdot a^m_{bs}) \tag{16}$$

$$T^b_{wl}(k,i) \approx s^2 \cdot B({a_{bs}}^{L_{bs}}, i, s) + \sum_{m=1}^{L_{bs}} a_{bs} \cdot {a_{bs}}^m \cdot s \cdot B({a_{bs}}^{L_{bs}-m}, i, s \cdot a^m_{bs}), \tag{17}$$

where $s = \frac{n^l_{bs}}{a^{L_{bs}}_{bs}}$. Equation (16) and (17) are derived based on the following intermediate results:

- On average, $B({a_{bs}}^{L_{bs}-m}, i, s \cdot a^m_{bs})$ keys need to be updated on level $(L_{bs} - m)$ of the BS subtree.

- To update one KEK at level L_{bs}, the average message size is (s) and these messages are broadcast to an average of (s) BSs. To update one KEK at level $(L_{bs} - m), m > 0$, the message size is (a_{bs}) and these messages are broadcast by (a_{bs}^m) BSs.

From the definition of A_2^l and using both (16) and (17), we can see that

$$\begin{aligned} A_2^l &= \sum_{i=1}^{n_{bs}^l} p_h^l(i)(\alpha_2^l \gamma T_{wl}^b(k,i) + \alpha_1^l(1-\gamma)T_w^b(k,i)) \\ &= \sum_{i=1}^{n_{bs}^l} p_h^l(i)\Bigg(B({a_{bs}}^{L_{bs}}, i, s)\left(s^2\alpha_2^l\gamma + s\alpha_1^l(1-\gamma)\right) \\ &\quad + \sum_{m=1}^{L_{bs}} B({a_{bs}}^{L_{bs}-m}, i, s\cdot a_{bs}^m)a_{bs}\left({a_{bs}}^m s\alpha_2^l\gamma + \alpha_1^l(1-\gamma)\right)\Bigg), \end{aligned} \tag{18}$$

where n_{bs}^l is the number of BSs under the l^{th} SH. In practice, it is difficult to obtain an analytic expression for $p_h^l(i)$ that depends on the statistical behavior of the users during membership joins and departures, as well as their mobility behavior and how handoffs are addressed. Thus, we introduce random variable $\tilde{I}^l$, which is the number of cells that a leaving user has ever visited. Obviously, $\tilde{I}^l \geq I^l$. The pmf of $\tilde{I}^l$, denoted by $\tilde{p}_h^l(i)$, can be derived from user mobility behavior and the distribution of the service duration, as described in Appendix B. Let $\tilde{A}_2^l$ denote the right hand side of (18) when replacing $p_h^l(i)$ by $\tilde{p}_h^l(i)$. We can show that $\tilde{A}_2^l$ is an upper bound of A_2^l. We notice that $\tilde{A}_2^l$ is not a function of k.

As discussed in Section 5, the parameters of the BS subtree under the l^{th} SH should be chosen such that $\sum_k p^l(k)G^l(k)A_2^l$ is minimized. Since $G^l(k)$ is not a function of a_{bs} and L_{bs}, minimizing $\sum_k p^l(k)G^l(k)A_2^l$ is equivalent to minimizing A_2^l. Due to the unavailability of $p_h^l(i)$, we choose the parameters of the BS subtrees under the l^{th} SH that minimize the upper bound of A_2^l, as:

$$\min_{a_{bs}>1, L_{bs}>0} \tilde{A}_2^l. \tag{19}$$

6.5 SH Subtree Design

In a typical cellular network, each SH administrates a large area where both the user dynamics and the network conditions may differ significantly from the areas administered by other SHs. The heterogeneity among the SHs should be considered when designing the SH subtree. Due to SH heterogeneity, the ALX tree structure, which treats every leaf equally, is not an appropriate tree structure to build the SH subtree. Instead, the SH heterogeneity may be addressed by building a tree where the SHs have varying path lengths from the root to their leaf node. In this section, we will first formulate the SH subtree design problem and then provide a sub-optimal tree generation procedure.

The root of the SH subtree is the KDC, and the leaves are the SHs. The design goal is to minimize the third term in equation (8), which shall be denoted by C_{sh} and is given by

$$C_{sh} = \sum_{l=1}^{n_{sh}} q_l \cdot A_3^l, \tag{20}$$

where $q_l = \sum_k p^l(k)G^l(k)$. Let β_l denote the communication cost of transmitting one rekeying message to all the users under the l^{th} SH. Based on the definition of α_1^l and α_2^l in Section 2, it is easy to show that $\beta_l = (1-\gamma)\alpha_1^l + \gamma n_{bs}^l \alpha_2^l$.

The value of A_3^l can be calculated directly from β_l where $l = 1, 2, \cdots, n_{sh}$. In the simple example demonstrated in Figure 8, when a user under SH$_1$ leaves the multicast service, K_{00}, K_0, K_ϵ and K_s, need to be updated. The communication cost of updating K_{00} is $2(\beta_1 + \beta_2)$. The communication cost of updating K_0 is $2(\beta_1 + \beta_2 + \beta_3)$. The communication cost of updating K_ϵ is $2(\beta_1 + \beta_2 + \beta_3 + \beta_4 + \beta_5)$. Since the communication cost of updating K_s does not depend on SH subtree structure, it is not counted in the total communication cost. Then, we have:

$$A_3^1 = 2(\beta_1 + \beta_2) + 2(\beta_1 + \beta_2 + \beta_3) + 2(\beta_1 + \beta_2 + \beta_3 + \beta_4 + \beta_5).$$

The goal of SH subtree design is to find a tree structure that minimizes C_{sh} given β_l and q_l. However, it is very difficult to do so based on (20). Thus, we compute C_{sh} in a different way.

We assume that the SH subtree has the fixed degree n. We shall assign a *cost pair*, which is a pair of positive numbers, to each node on the tree as follows. The cost pair of the leaf node that represents the l^{th} SH is (q_l, β_l). The cost pair of the intermediate nodes are the element-wise summation of their children nodes' cost pairs, as illustrated in Figure 9. The cost pairs of all intermediate nodes are represented by (x_m, y_m), where $m = 1, 2, \cdots, M$, and M is the total number of intermediate nodes on the tree. Then, C_{sh} can be calculated as:

$$C_{sh} = n \sum_{m=1}^{M} x_m \cdot y_m. \tag{21}$$

It is easy to verify that (21) is equivalent to (20). Based on (21), we propose a tree construction method for $n = 2$ as:

1. Label all the leaf nodes using their cost pairs, and mark them to be active nodes.
2. Choose two active nodes, (x_i, y_i) and (x_j, y_j), such that $(x_i+x_j)\cdot(y_i+y_j)$ is minimized among all possible pairs of active nodes. Mark those two nodes to be inactive and merge them to generate a new active node with the cost pair $(x_i + x_j, y_i + y_j)$.
3. Repeat step 2 until there is only one active node left.

This method, which we call the greedy-SH subtree-design (GSHD) algorithm, can be easily extended to $n > 2$ cases. We can prove that the GSHD algorithm produces the optimal solution when $\beta_1 = \beta_2 = \cdots \beta_{n_{sh}}$, but is not optimal in general cases. Since the optimization problem for the SH-subtree is non-linear, combinatorial, and even does not have a closed expression for the objective function, we do not seek the optimal SH subtree structure in this paper. In Section 7, we will compare the performance of the GSHD algorithm and the optimal solution obtained by exhaustive search.

7 Simulation Results

7.1 One-SH Systems

We first compare the performance of the TMKM tree and the TIKM tree in one-SH systems using both analysis and simulations. Similar to [28, 29], we employ a homogeneous cellular network that consists of 12 concatenated cells, and wrap the cell pattern to avoid edge effects. We use the mobility model proposed in [30], where R denotes the radius of the cells, and V_{max} denotes the maximum speed of the mobile users. Since the wireless connection usually experiences a high transmission error rate and the number of users under one BS is larger than the number of BSs, the wireless communication cost of the multicast communication is assigned a larger weight than the wireline communication cost, i.e. $\gamma > 0.5$.

For the purpose of fair comparison, the TIKM tree is designed as an ALX tree, which is optimized for the statistics of the number of participating users. The wireline cost of the TIKM tree, denoted by C^{tikm}_{wire}, is computed using (9), where $p(k)$ denotes the pmf of the number of users in the multicast service. The wireless cost of the TIKM tree is computed as $C^{tikm}_{wireless} = n_{bs}C^{tikm}_{wire}$, where n_{bs} is the total number of BSs. In one-SH systems, the total communication cost is $C^{tikm}_{T} = \gamma C^{tikm}_{wireless} +$

$(1-\gamma)C_{wire}^{tikm}$. We define the *performance ratio* η as the total communication cost of the TMKM tree divided by the total communication cost of the TIKM tree, i.e. $\eta = C_T^{tmkm}/C_T^{tikm}$. When η is less than 1, the TMKM tree has smaller communication cost than the TIKM tree, and smaller η indicates an improved advantage that the TMKM tree has over the TIKM tree.

Figure 10(a) shows the total communication cost of the TMKM tree and the TIKM tree for different wireless weights (γ), when the cellular cells have a radius of 4 miles, the maximum mobile speed is 50 miles/hour, and the user joining rate is 16 users per minute per cell. The corresponding performance ratio is shown in Figure 10(b). In this simulation, we have used two models, which have been shown to accurately capture multicast session behavior [23, 24], to describe users' join/departure characteristics. The first one, representing short sessions, uses a Poisson arrival and exponential service time duration model. The second one, representing long sessions, uses a Poisson arrival and Zipf service time duration model. The users stay in the service for an average of 20 minutes in both cases. Three observations are made. First, the communication cost of the TMKM tree is always less than 42% of the communication cost of the TIKM tree. Second, the performance ratio η is smaller for larger γ, which supports the argument in Section 4 that the advantage of the TMKM tree is larger when more emphasis is placed on the wireless cost. Third, when the wireless transmission is the bottleneck of the system, i.e. $\gamma = 1$, the TMKM tree can reduce the communication burden by as much as 65%, i.e. $\eta = 35\%$. In addition, two models yield similar results, which indicates that the performance of the TMKM is not sensitive to the models. In the remainder of this section, we adopt the short session model.

Figure 11(a) shows both the analysis and the simulation results of η for different user join rates (λ) when the radius of the cellular cells is 4 miles, the maximum mobile speed is 50 miles per hour, the average service time ($1/\mu$) is 20 minutes, and $\gamma = 2/3$. Since the exact expression for the pmf of I^l is not available, to calculate analytical results, we use an empirically estimated pmf of I^l, which is obtained from simulations with the same user join/departure and mobility models.

We can see that the advantage of the TMKM tree is larger when the system contains more users. This property can be verified by studying the cost functions derived in the previous sections. In Figure 11(b), the performance ratio is shown for different V_{max} when the user joining rate is 16 users per minute per cell. The performance ratio is an increasing function of V_{max} when other parameters are fixed since handoffs occur more frequently as users move faster.

7.2 Multiple-SH Systems

As discussed in Section 6, when the system contains multiple SHs and the SHs do not perform key management, the design the TMKM tree should consider the topology of the SHs.

7.2.1 SH subtree design methods

In this section, we compare the GSHD algorithm with the optimal tree obtained by exhaustive search, and with a balanced tree that treats the SHs equally and represents traditional key management schemes. We assume that half of the $\{\beta_l\}$ are uniformly distributed between 1 and 20, which represent rural areas, and the other half of $\{\beta_l\}$ uniformly distributed between 101 and 120, which represent metropolitan areas. We also assume that q_l, which is defined in Section 6.5 and represents the probability of a user leaving, is proportional to β_l, where $l = 1, 2, \cdots, n_{sh}$. Here, $\{q_l\}$ are normalized such that $\sum q_l = 1$. In Figure 12, the communication cost caused by updating keys on SH-subtrees, C_{sh}, is shown when using different SH subtrees. Results are averaged over 500 realizations. Since exhaustive search is very computationally expensive, it is only done for 10 and fewer SHs. The simulation results indicate that the performance of the GSHD is very close to optimal. Compared with the balanced tree, the GSHD algorithm reduces the communication cost contributed by the SH subtree by up to 18%.

7.2.2 Performance of TMKM trees and TIKM trees in multiple-SH systems

For the TMKM trees in multiple-SH systems, we designed the user-subtrees and BS-subtrees as ALX trees, while the SH-subtrees were constructed using the GSHD algorithm. We simulated a multiple-SH system where each SH administers 12 concatenated identical cells. The SH-subtrees are constructed as binary trees. We first study a simple case where the user statistics and network conditions are identical under all SHs. In this case, α_1^l's and α_2^l's are set to be 1. The radius of the cells is $R = 4$ miles, the maximum velocity is $V_{max} = 50$ miles/hr, and we also choose $\mu_l = 1/30$ and $\lambda_l = 10$ for all SHs.

In Figure 13, the wireless cost and the wireline cost of the TMKM trees and the TIKM tree are shown for different quantities of participating SHs. We observed that the TMKM trees have both smaller wireless cost and smaller wireline costs than the TIKM trees when the number of SHs are equal or greater than 2, and the advantages of the TMKM trees are more significant when the system contains more SHs, which verifies the analysis in Section 4. In addition, the corresponding performance ratio is drawn in Figure 14 for $\gamma = 2/3$. In this system, the communication cost of the TMKM trees can be as low as 20% of the communication cost of the TIKM trees. This indicates an 80% reduction in the communication cost.

A more complicated system containing 5 SHs with different user joining rates was also simulated. In this scenario, the λ_l values for the five SHs were set to 5, 10, 15, 20 and 25 respectively, and $R = 4$ miles, $V_{max} = 50$ miles/hr, and $\mu_l = 1/20$ for all SHs. The TMKM tree structure is shown in Figure 15. The TIKM tree is simply an ALX tree with degree 3 and level 6. In this system, the wireless cost of the TMKM tree is 21.8% of that of the TIKM tree, and the wireline cost of the TMKM tree is 34.0% of that of the TIKM tree. When the wireless weight γ is set to 2/3, the TMKM tree reduced total communication cost by 74%.

8 Conclusion

In this paper, we presented a method for designing the multicast key management tree for the mobile wireless environment. By matching the key management tree to the cellular network topology and localizing the delivery of rekeying messages, a significant reduction of 55-80% in the communication burden associated with rekeying was observed compared to trees that are independent of the topology.

We designed a topology-matching key management (TMKM) tree that consists of user-subtrees, BS-subtrees and SH-subtrees. It was shown that the problem of optimizing the communication cost for the TMKM tree is separable and can be solved by optimizing each of those subtrees separately. The ALX tree structure, which easily adapts to changes in the number of users, was introduced to build user-subtrees and BS-subtrees. The performance of the ALX tree is very close to the performance lower bound for any fixed degree tree. The GSHD algorithm, which considers the network heterogeneity where the SHs administer areas with varying network conditions, was introduced to build the SH subtree. The performance of the GSHD algorithm is very close to optimal and has better performance than treating SHs equally. Additionally, we addressed the consequences that user mobility has upon the topology-matching key management tree, and presented an efficient handoff scheme to reduce the communication burden associated with rekeying.

A popular user joining/leaving procedure was used to study the performance of the TMKM and TIKM trees. Both simulations and analysis were provided. For systems consisting of only one SH, simulations performed for different user-join rates and mobile user speeds show that the cost of the TMKM tree is approximately 33-45% of the cost of the TIKM tree, which indicates a reduction of 55-67% in the total communication cost. For systems consisting of multiple SHs, simulations were performed for different amounts of participating SHs, and indicated that the TMKM tree can reduce the communication burden by as much as 80%. In addition, both analysis and simulations indicate that the communication cost of the TMKM tree scales better than that of topology-independent trees as the number of participating SHs increases.

Appendix A Calculation of $B(b, i, a)$

We define $n(b, i, a)$ to be the number of non-empty boxes when randomly placing i identical items into b identical boxes with repetition, where each box can hold at most a items. In this appendix, we calculate $B(b, i, a) = E[n(b, i, a)]$, the expected value of $n(b, i, a)$. It is obvious that the value of $n(b, i, a)$ is bounded as $B_0 \leq n(b, i, a) \leq B_1$, where $B_0 = \left\lceil \frac{i}{a} \right\rceil$ and $B_1 = \min(i, b)$.

We define an intermediate quantity $w(y, i, a)$ as the number of ways of putting i items into y boxes such that each box contains at least 1 and at most a items. $w(y, i, a)$ can be calculated recursively as:

$$w(B_0, i, a) = \binom{aB_0}{i} \tag{22}$$

$$w(B_0+1, i, a) = \binom{a(B_0+1)}{i} - \binom{B_0+1}{B_0} w(B_0, i, a) \tag{23}$$

$$\vdots$$

$$w(B_0+k, i, a) = \binom{a(B_0+k)}{i} - \sum_{m=0}^{k-1} \binom{B_0+k}{B_0+m} w(B_0+m, i, a), \tag{24}$$

where $0 \leq k \leq B_1 - B_0$. Then, the pmf of $n(b, i, a)$ can be expressed as:

$$Prob\{n(b, i, a) = B_0 + k\} = \frac{1}{N} \binom{b}{B_0+k} w(B_0+k, i, a), \tag{25}$$

where $N = \binom{ab}{i}$ represents the total number of ways of putting i items into b boxes. By substituting (24) into (25), we get:

$$\begin{aligned} & Prob\{n(b,i,a) = B_0 + k\} \\ = \; & \frac{1}{N}\binom{b}{B_0+k}\binom{a(B_0+k)}{i} - \sum_{m=0}^{k-1} \frac{\binom{b}{B_0+k}\binom{B_0+k}{B_0+m}}{\binom{b}{B_0+m}} Prob\{n(b,i,a) = B_0 + m\}. \end{aligned}$$

It can be shown that:

$$\frac{\binom{b}{B_0+k}\binom{B_0+k}{B_0+m}}{\binom{b}{B_0+m}} = \binom{b - B_0 - m}{k - m}.$$

Therefore,

$$\begin{aligned} & Prob\{n(b,i,a) = B_0 + k\} \\ = \; & \frac{1}{N}\binom{b}{B_0+k}\binom{a(B_0+k)}{i} - \sum_{m=0}^{k-1} \binom{b - B_0 - m}{k - m} Prob\{n(b,i,a) = B_0 + m\}. \end{aligned} \qquad (26)$$

By substituting (22) into (25), we have:

$$Prob\{n(b,i,a) = B_0\} = \frac{1}{N}\binom{b}{B_0}\binom{aB_0}{i}. \qquad (27)$$

Based on (26) and (27), we can calculate $Prob\{n(b,i,a) = B_0 + k\}$ for $k = 0, 1, \cdots, B_1 - B_0$ recursively. Then, we can calculate $B(b,i,a)$ as:

$$B(b,i,a) = E[n(b,i,a)] = \sum_{k=0}^{B_1 - B_0} (B_0 + k) \cdot Prob\{n(b,i,a) = B_0 + k\}. \qquad (28)$$

Appendix B Calculation of pmf of $\tilde{I}$ in Section 6.4

Let t_M denote the service duration, t_n denote the new cell dwell time, and t_h denote the previously handed-off cell dwell time [30]. We assume that t_M follows exponential distribution. The distributions of t_n and t_h are often presented together with the mobility models. For the mobility model used in Section 7, the distribution of t_n and t_h can be found in [30].

Using these distributions, we can calculate $p_n = Prob\{t_M < t_n\}$ and $p_h = Prob\{t_M < t_h\}$. The number of cells that a user ever visited before departure, denoted by $\tilde{I}$, has the pmf as $Prob\{\tilde{I} = 1\} = p_n$, $Prob\{\tilde{I} = 2\} = (1 - p_n)p_h$, $Prob\{\tilde{I} = 3\} = (1 - p_n)(1 - p_h)p_h$, and $Prob\{\tilde{I} = i\} = (1 - p_n)(1 - p_h)^{i-2}p_h$.

References

[1] S. Paul, *Multicast on the Internet and its applications*, Kluwer Academic Publishers, 1998.

[2] C. Diot, B.N. Levine, B. Lyles, H. Kassem, and D. Balensiefen, "Deployment issues for the IP multicast service and architecture," *IEEE Network*, vol. 14, pp. 78 –88, Jan.-Feb 2000.

[3] A. Acharya and B.R. Badrinath, "A framework for delivering multicast messages in networks with mobile hosts," *Journal of Special Topics in Mobile Networks and Applications*, vol. 1, no. 2, pp. 199–219, Oct. 1996.

[4] H-S Shin and Y-J Suh, "Multicast routing protocol in mobile networks," *Proc. IEEE International Conference on Communications*, vol. 3, pp. 1416 –1420, June 2000.

[5] D.M. Wallner, E.J. Harder, and R.C. Agee, "Key management for multicast: issues and architectures," Internet Draft Report, Sept. 1998, Filename: draft-wallner-key-arch-01.txt.

[6] M.J. Moyer, J.R. Rao, and P. Rohatgi, "A survey of security issues in multicast communications," *IEEE Network*, vol. 13, no. 6, pp. 12–23, Nov.-Dec. 1999.

[7] C. Wong, M. Gouda, and S. Lam, "Secure group communications using key graphs," *IEEE/ACM Trans. on Networking*, vol. 8, pp. 16–30, Feb. 2000.

[8] R. Canetti, J. Garay, G. Itkis, D. Miccianancio, M. Naor, and B. Pinkas, "Multicast security: a taxonomy and some efficient constructions," *Proc. IEEE INFOCOM'99*, vol. 2, pp. 708–716, March 1999.

[9] W. Trappe, J. Song, R. Poovendran, and K.J.R. Liu, "Key distribution for secure multimedia multicasts via data embedding," *Proc. IEEE ICASSP'01*, pp. 1449–1452, May 2001.

[10] M. Waldvogel, G. Caronni, D. Sun, N. Weiler, and B. Plattner, "The VersaKey framework: Versatile group key management," *IEEE Journal on selected areas in communications*, vol. 17, no. 9, pp. 1614–1631, Sep. 1999.

[11] D. Balenson, D. McGrew, and A. Sherman, "Key management for large dynamic groups: one-way function trees and amortized initialization," Internet Draft Report.

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

[13] S. Mittra, "Iolus: A framework for scalable secure multicasting," in *Proc. ACM SIGCOMM '97*, 1997, pp. 277–288.

[14] S. Banerjee and B. Bhattacharjee, "Scalable secure group communication over IP multicast," *JSAC Special Issue on Network Support for Group Communication*, vol. 20, no. 8, pp. 1511 –1527, Oct. 2002.

[15] G. Caronni, K. Waldvogel, D. Sun, and B. Plattner, "Efficient security for large and dynamic multicast groups," *Proc. Seventh IEEE International Workshop on Enabling Technologies: Infrastucture for Collaborative Enterprises (WET ICE '98)*, pp. 376 – 383, June 1998.

[16] K. Brown and S. Singh, "RelM: Reliable multicast for mobile networks," *Computer Communication*, vol. 2.1, no. 16, pp. 1379–1400, June 1996.

[17] E. Ha, Y. Choi, and C. Kim, "A multicast-based handoff for seamless connection in picocellular networks," *Proc. IEEE Asia Pacific Conference on Circuits and Systems*, pp. 167 –170, Nov. 1996.

[18] Universal Mobile Telecommunications System (UMTS) Technical Specification, Digital cellular telecommunications system (Phase 2+ (GSM)), "Network architecture," 3GPP TS 23.002 version 5.9.0 Release 5, 2002-12.

[19] J.E. Wieselthier, G.D. Nguyen, and A. Ephremides, "On the construction of energy-efficient broadcast and multicast trees in wireless networks," *Proc. IEEE INFOCOM'00*, vol. 2, pp. 585 –594, March 2000.

[20] L. Gong and N. Shacham, "Multicast security and its extension to a mobile environment," *Wireless Networks*, vol. 1, no. 3, pp. 281–295, 1995.

[21] M. Hauge and O. Kure, "Multicast in 3G networks: employment of existing IP multicast protocols in umts," in *Proceedings of the 5th ACM international workshop on Wireless mobile multimedia*. 2002, pp. 96–103, ACM Press.

[22] "Mlisten," available at www.cc.gatech.edu/computing/Telecomm.mbone.

[23] K. Almeroth and M. Ammar, "Collecting and modeling the join/leave behavior of multicast group members in the mbone," in *Proc. High Performance Distributed Computing (HPDC'96), Syracuse, New York*, 1996, pp. 209–216.

[24] K. Almeroth and M. Ammar, "Multicast group behavior in the internet's multicast backbone (MBone)," *IEEE Communications*, vol. 35, pp. 224–229, June 1999.

[25] G.K. Zipf, *Human Behavior and the Principle of Least Effort*, Addison-Wesley Press, 1949.

[26] A. Leon-Garcia, *Probability and Random Processes for Electrical Engineering*, Addison Wesley, 2nd edition, 1994.

[27] T. M. Cover and J. A. Thomas, *Elements of Information Theory*, Wiley-Interscience, 1991.

[28] M. Rajaratnam and F. Takawira, "Nonclassical traffic modeling and performance analysis of cellular mobile networks with and without channel reservation," *IEEE Trans. on Vehicular Technology*, vol. 49, no. 3, pp. 817–834, May 2000.

[29] M. Sidi and D. Starobinski, "New call blocking versus handoff blocking in cellular networks," *Proc. IEEE INFOCOM '96*, vol. 1, pp. 35–42, March 1996.

[30] M. M. Zonoozi and P. Dassanayake, "User mobility modeling and characterization of mobility patterns," *IEEE Journal on Selected Areas in Communications*, vol. 15, no. 7, pp. 1239–1252, Sep. 1997.

Tables

	wireline-message-size	wireless-message-size
TIKM	$\sim aN\log_a N$	$\sim a^{L_1+1}N\log_a N$
TMKM	$\sim a^2\log_a N$	$\sim a^{L_1+2}\log_a N$

Table 1: Scalability comparison between TMKM and TIKM trees when the number of SHs(N)$\rightarrow\infty$.

Figures

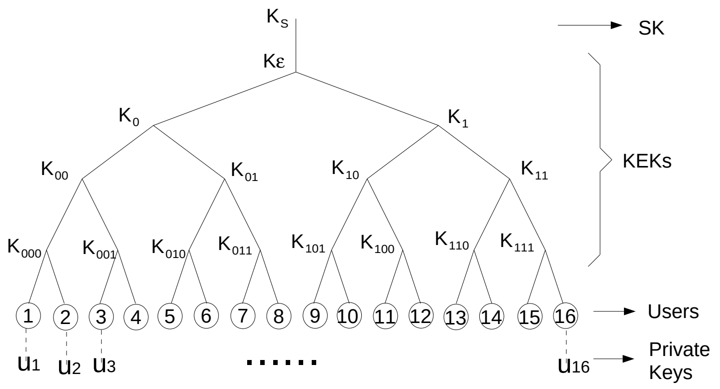


Figure 1: A typical key management tree

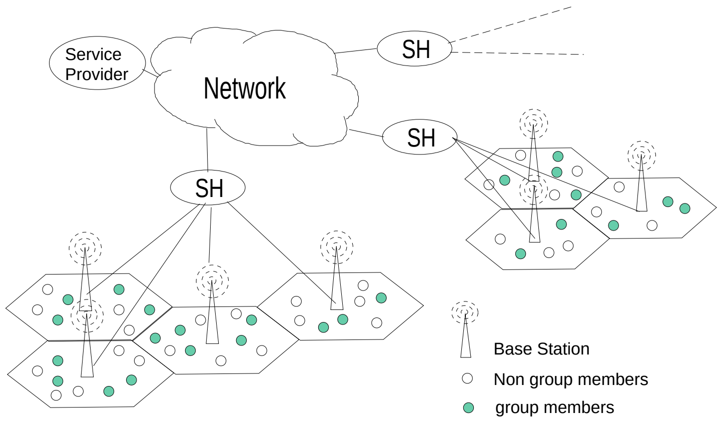


Figure 2: A cellular wireless network model

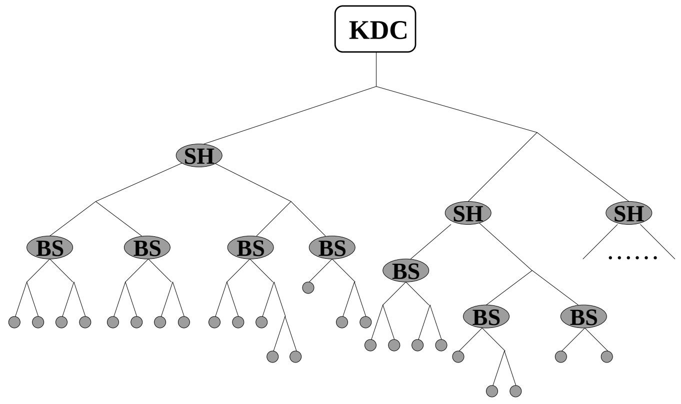


Figure 3: A Topology Matching Key Management Tree

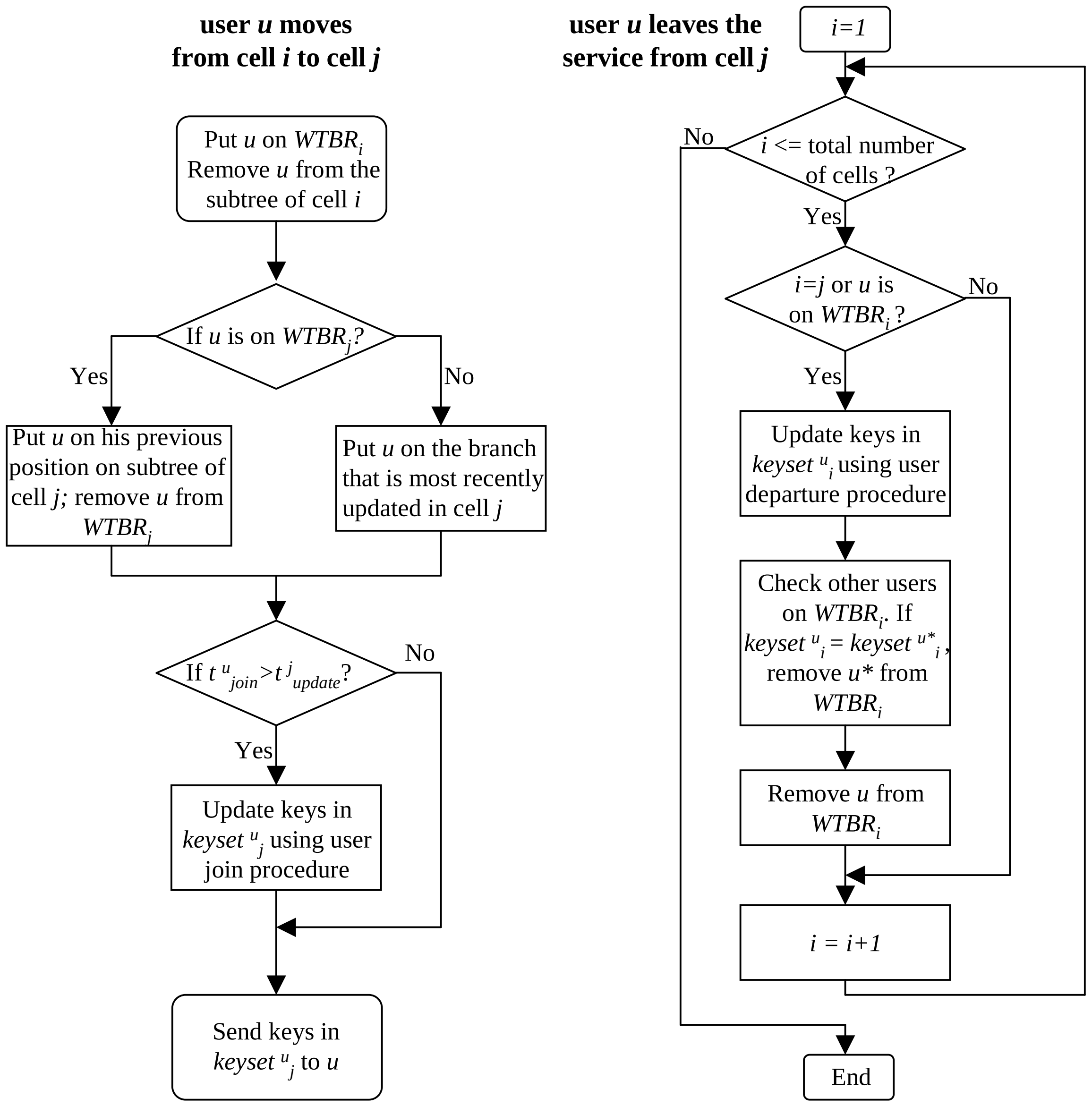

user u moves from cell i to cell j
Put u on WTBR_i Remove u from the subtree of cell i
If u is on WTBR_j?
Yes
No
Put u on his previous position on subtree of cell j; remove u from WTBR_j
Put u on the branch that is most recently updated in cell j
If t^u_join > t^j_update?
No
Yes
Update keys in keyset^u_j using user join procedure
Send keys in keyset^u_j to u
user u leaves the service from cell j
i=1
i <= total number of cells ?
No
Yes
i=j or u is on WTBR_i ?
No
Yes
Update keys in keyset^u_i using user departure procedure
Check other users on WTBR_i. If keyset^u_i = keyset^u*_i, remove u* from WTBR_i
Remove u from WTBR_i
i = i+1
End

Figure 4: User relocation and departure in TMKM

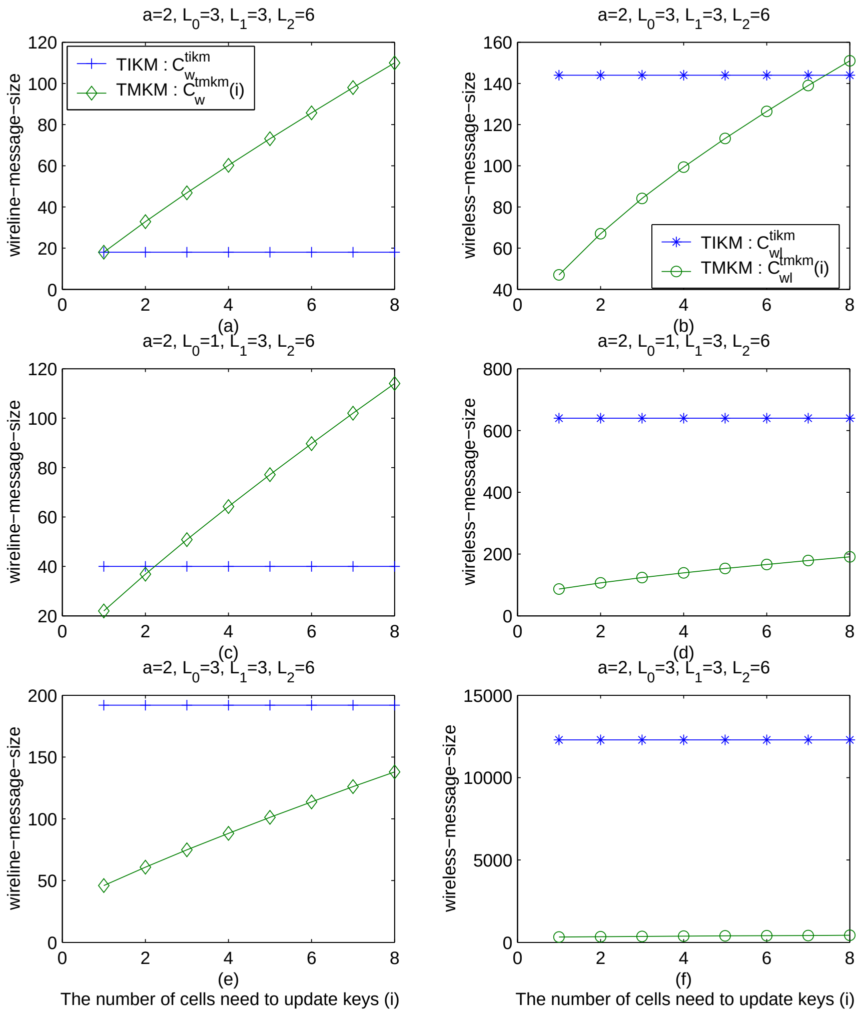


Figure 5: Comparison of the wireless cost and the wireline cost when one user leaves. Top row corresponds to one SH, middle row corresponds to two SHs, and bottom row corresponds to eight SHs.

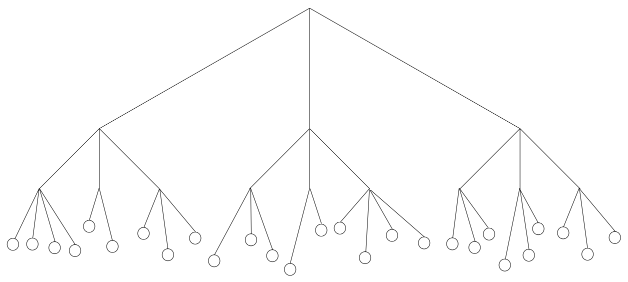


Figure 6: ALX tree

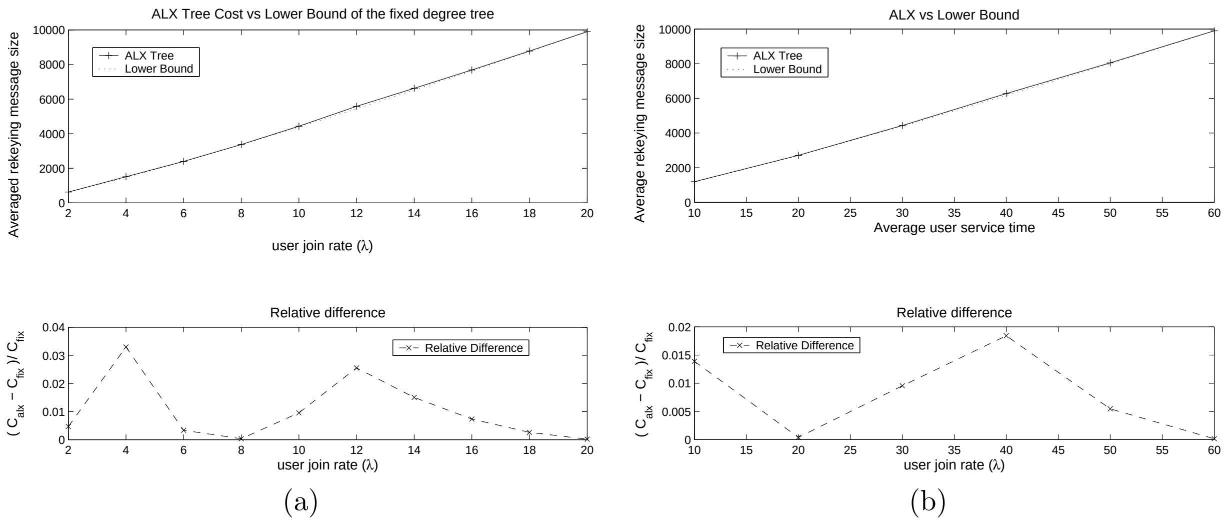


Figure 7: Comparison of the ALX tree performance and the Lower Bound

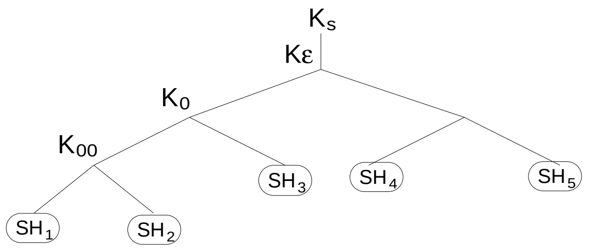


Figure 8: An example of the SH subtree

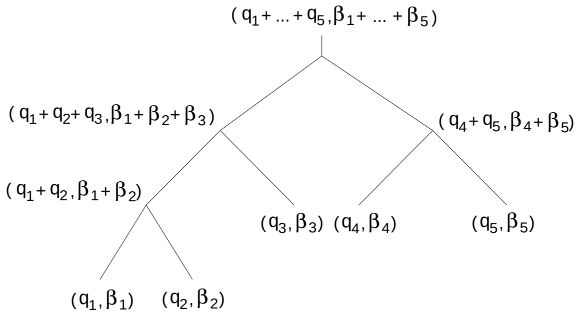
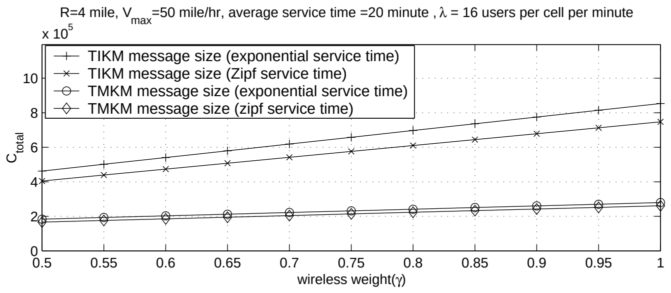
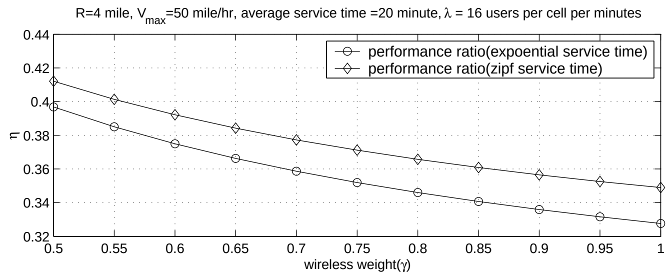

$(q_1 + ... + q_5, \beta_1 + ... + \beta_5)$
$(q_1 + q_2 + q_3, \beta_1 + \beta_2 + \beta_3)$
$(q_4 + q_5, \beta_4 + \beta_5)$
$(q_1 + q_2, \beta_1 + \beta_2)$
(q_3, β_3)
(q_4, β_4)
(q_5, β_5)
(q_1, β_1)
(q_2, β_2)

Figure 9: The cost pairs on the SH subtree

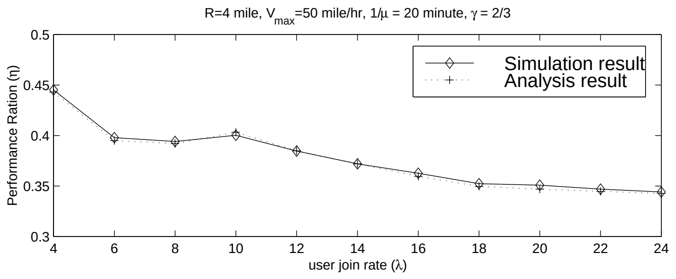


(a)

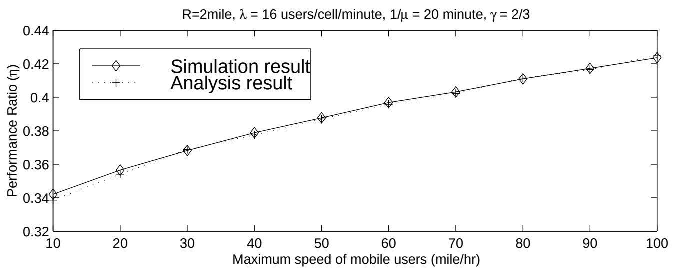


(b)

Figure 10: (a) The total message size as the function of the wireless weight. (b) Performance Ratio as a function of the wireless weight.



(a)



(b)

Figure 11: (a) Performance Ratio as a function of user join rate, (b) Performance Ratio as a function of users' maximum speed

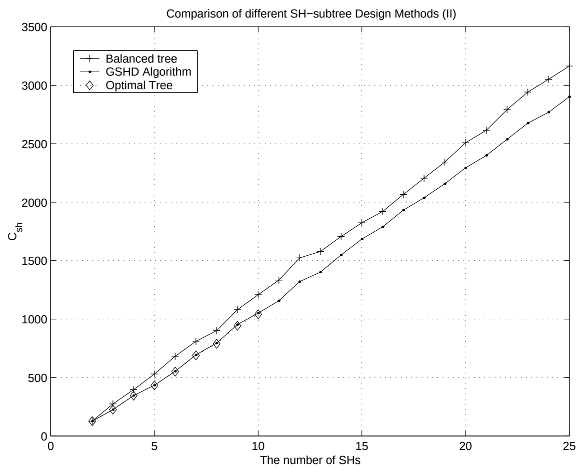


Figure 12: Comparison of Several SH subtree design methods

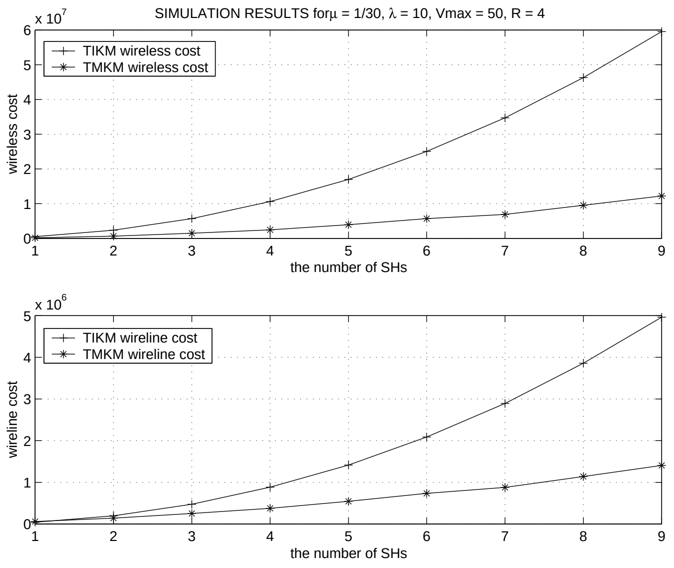


Figure 13: Performance comparison in multiple-SH systems with identical SHs

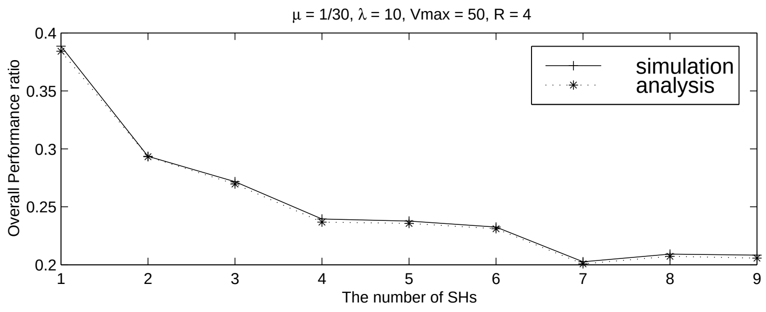


Figure 14: Performance comparison in multiple-SH systems with non-identical SHs

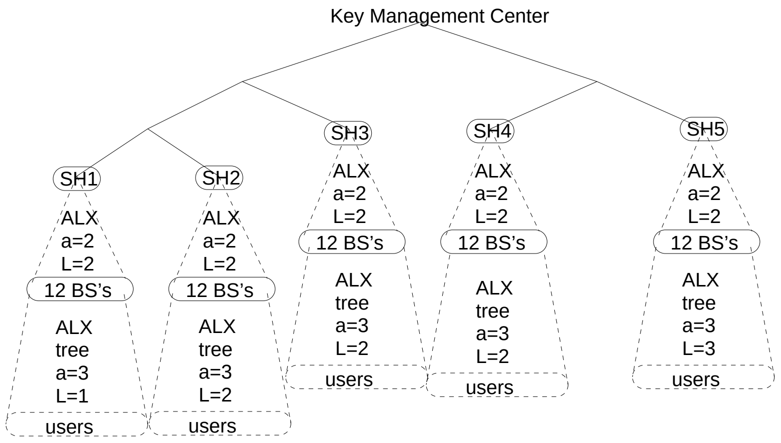


Figure 15: A TMKM tree containing 5 SHs