Distributed Servers Approach for Large-Scale Secure Multicast

Kin-Ching Chan and S.-H. Gary Chan

***Abstract*—In order to offer backward and forward secrecy for multicast applications (i.e., a new member cannot decrypt the multicast data sent before its joining and a former member cannot decrypt the data sent after its leaving), the data encryption key has to be changed whenever a user joins or leaves the system. Such a change has to be made known to all the current users. The bandwidth used for such re-key messaging can be high when the user pool is large. In this paper, we propose a distributed servers approach to minimize the overall system bandwidth (and complexity) by splitting the user pool into multiple groups each served by a (logical) server. After presenting an analytic model for the system based on a hierarchical key tree, we show that there is an optimal number of servers to achieve minimum system bandwidth. As the underlying user traffic fluctuates, we propose a simple dynamic scheme with low overhead where a physical server adaptively splits and merges its traffic into multiple groups each served by a logical server so as to minimize its total bandwidth. Our results show that a distributed servers approach is able to substantially reduce the total bandwidth required as compared with the traditional single-server approach, especially for those applications with a large user pool, short holding time, and relatively low bandwidth of a data stream, as in the Internet stock quote applications.**

***Index Terms*—Distributed servers approach, key tree, multicast security, re-key messaging, split-and-merge scheme.**

I. Introduction

MULTICAST is an efficient technique for delivering data to a large group of users in multimedia applications such as Internet stock quotes, Internet radio, audio/music delivery, video surveillance, etc., [1]. Many of these applications requires data security. The current multicast protocols such as distance vector multicast routing protocol (DVMRP), core-based tree (CBT), and protocol independent multicast–distance measuring (PIM-DM) [2]–[4], however, do not offer any security features in terms of confidentiality, authenticity and integrity. In this paper, we mainly study the data confidentiality issue in the multicast environment (i.e., unauthorized users should not be able to access the multicast data). In such a system, a new member should not be able to decrypt those multicast data sent *before* its joining (i.e., the so-called "backward secrecy") and a former member should not be able to decrypt those multicast data sent *after* its departure or eviction (i.e., the so-called "forward secrecy") [5].

Traditional data security is generally based on public key infrastructure (PKI) technology and applied in the unicast environment. Such a point-to-point approach is not applicable in the multicast environment with a large number of senders and receivers (the so-called "participants") and when the group is highly dynamic, i.e., the group members join and leave frequently and at random times. Therefore, whenever there is a membership change in a group, the data has to be reencrypted with a different key and the corresponding decryption key has to be made known to all members in the group. If not managed properly, these "re-key messages" which inform the key change would consume a large amount of network bandwidth and processing overhead. An efficient solution to address this issue of key management has been proposed independently by Wong *et al.* and Wallner *et al.* [6], [7] Both schemes introduce a hierarchical key tree structure in which the group members are arranged as a logical key tree. Each group member is at the leaf of the tree and belongs to more than one multicast subgroup. Using this approach, the number of re-key messages for each change of membership (in the form of "join" and "leave") is shown to be only $O(\log N)$, where N is the number of concurrent users in the system, i.e., the group size.

In each server of a secure multicast system, there is generally a data manager and a control manager. The data manager encrypts and transmits data while the control manager is responsible for key management such as generating, storing and distributing keys. One problem with a single server system serving the whole population is that its complexity increases as the number of user increases, mainly due to the large number of re-key messages and the size of the key database (there are generally many keys involved in a secure multicast group). Therefore, in order to reduce the complexity and manageability of the system when the multicast group is large, it may be beneficial to split the group into a number of smaller groups and serve them independently, thus forming a distributed servers network. We show in Fig. 1 such a system, in which the servers distributed in the network serve their respective pools of users. Note that these servers are not necessarily geographically distributed—they may be logical servers in a physical server. Therefore, a distributed servers network may consist of one or more physical servers, each of which may contain one or more logical servers. Data is multicast to the users from the

Manuscript received September 1, 2001; revised May 2002. This work was supported in part by the Areas of Excellence (AoE) on Information Technology funded by the University Grant Council in Hong Kong (AoE/E-01/99), and by the Sino Software Research Institute in the HKUST (SSRI00/01.EG04). This work was presented in part at the IEEE Globecom'01, San Antonio, TX. The work of K.-C. Chan was done while he was with the Department of Computer Science, The Hong Kong University of Science and Technology, Clear Water Bay, Kowloon, Hong Kong.

K.-C. Chan was with the Department of Computer Science, The Hong Kong University of Science and Technology, Clear Water Bay, Kowloon, Hong Kong and is now with Roctec Technology Ltd., Hong Kong.

S.-H. G. Chan is with the Department of Computer Science, The Hong Kong University of Science and Technology, Clear Water Bay, Kowloon, Hong Kong and also with Roctec Technology Ltd., Hong Kong (e-mail: gchan@cs.ust.hk).

Digital Object Identifier 10.1109/JSAC.2002.803966.

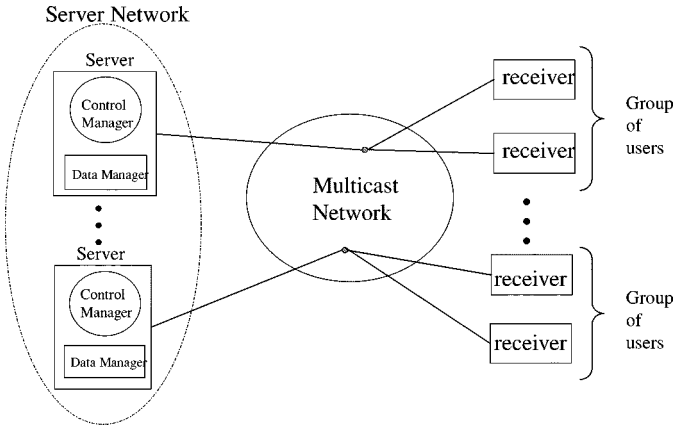

Fig. 1. Example of multicast system.

respective data manager, while the control manager notifies its users of the decryption keys.

Note that in such a server network, the total amount of multicast data traffic is proportional to the number of servers. On the other hand, as the number of servers increases, the overhead and, hence, the system complexity, in re-key messaging decreases (due to a decrease in the number of users served by each server). Therefore, there is a tradeoff between re-key messaging and data traffic and a corresponding optimal number of servers such that the total bandwidth requirement of the servers is minimized given a certain user traffic. Clearly, when the data rate is high and the users are less dynamic (as in some video applications), splitting the pool of users into many groups may not be beneficial; on the other hand, if the data rate is low and the user pool is large and highly dynamic (as in stock quote applications), the pool of users is more likely to be split into many groups.

Note that user traffic to a physical server may not be stationary. For example, in an Internet stock quote system, user traffic is expected to be higher at the start of the day than at the end of the day. It is, therefore, important for a server to split and merge its traffic dynamically into multiple groups each served by a logical server in order to minimize the total bandwidth. In this way, our view of the distributed servers network is thus hierarchical in nature: Given a certain target user traffic, the total pool of users is split and served by multiple independent physical servers to minimize the total bandwidth. When the underlying traffic changes, each physical server may further split and merge dynamically into multiple logical servers to serve its user traffic to minimize its own bandwidth. We propose an efficient scheme for such splitting and merging, and study the conditions under which "splits" and "merges" should be executed.

There are three contributions of this paper: 1) we present a simple and yet accurate model validated by extensive simulation of a distributed servers network for secure multicast; 2) we determine the optimal number of servers in order to minimize the total bandwidth in the system given a certain target user traffic; and 3) we propose a dynamic split-and-merge scheme to reduce bandwidth requirement as the underlying user traffic fluctuates, and study the conditions under which this can be done efficiently.

Our result shows that the distributed servers approach can achieve a substantial reduction in bandwidth (more than 30% in our examples) as compared with a single server system for secure multicast. This is especially true for some applications characterized by low data rates and fairly large groups of concurrent users (e.g., 100 000) (such as Internet stock quote applications). With our dynamic split-and-merge scheme, a further reduction in bandwidth (15% in our example) can be achieved when the underlying traffic fluctuates as compared with a static scheme. Given an application, there is an optimal number of users a server should serve; if the underlying traffic deviates from the number, our split-and-merge scheme should be used to regroup the users to reduce the cost.

Much of the previous work on secure multicast focuses on the key tree scheme. This body of work includes reducing the number of re-key messages and the number of keys stored in the server [8]–[10], maintaining the key tree by means of growing and pruning [11], [12]. Our work is based on the scheme given in [6] and [7], and we model and analyze it. Previous work addresses mainly reducing re-key messages and has considered neither a distributed servers network nor the tradeoff between multicast data and re-key messages. It mainly focuses on a specific number of users in the system and provides no analysis on the re-keying cost when the users join and leave dynamically. Some other work focuses on non-tree-based schemes. We will not discuss the schemes here; readers interested in them are referred to [13]–[15] and references therein.

This paper is organized as follows. In Section II, we first review the key tree scheme, present a model for the distributed servers network using this scheme, and provide illustrative numerical examples and results. In Section III, we discuss and analyze the split-and-merge scheme. We finally conclude in Section IV.

II. Scheme, Analytic Model, and Numerical Results

In [6] and [7], an effective hierarchical key tree approach is proposed to facilitate the distribution of re-key messages. They show that, whenever a member joins or leaves the system, the number of re-key messages is $O(\log N)$, where N is the number of concurrent users. In this section, we first review the scheme in Section II-A, then we present the analysis of the scheme in Section II-B, and provide some illustrative numerical results in Section II-C.

A. Scheme Description

A hierarchical key tree is a logical tree structure corresponding to each multicast group stored in the control manager. In the tree representation, group members are arranged at the leaves and the internal nodes store keys (see Fig. 2 for a k-ary tree with depth d). There are three types of keys. The first one is a group key K_1 used to encrypt/decrypt the multicast data; the second one is a subgroup key (such as K_{d-1} and K_d) used to encrypt/decrypt other keys instead of actual data, and the last one is the individual (public) key I. Each member holds the keys along the path from its leaf to the root. Therefore, for the case of member u, u holds $K_1, \ldots, K_{d-1}, K_d$. Each subtree in

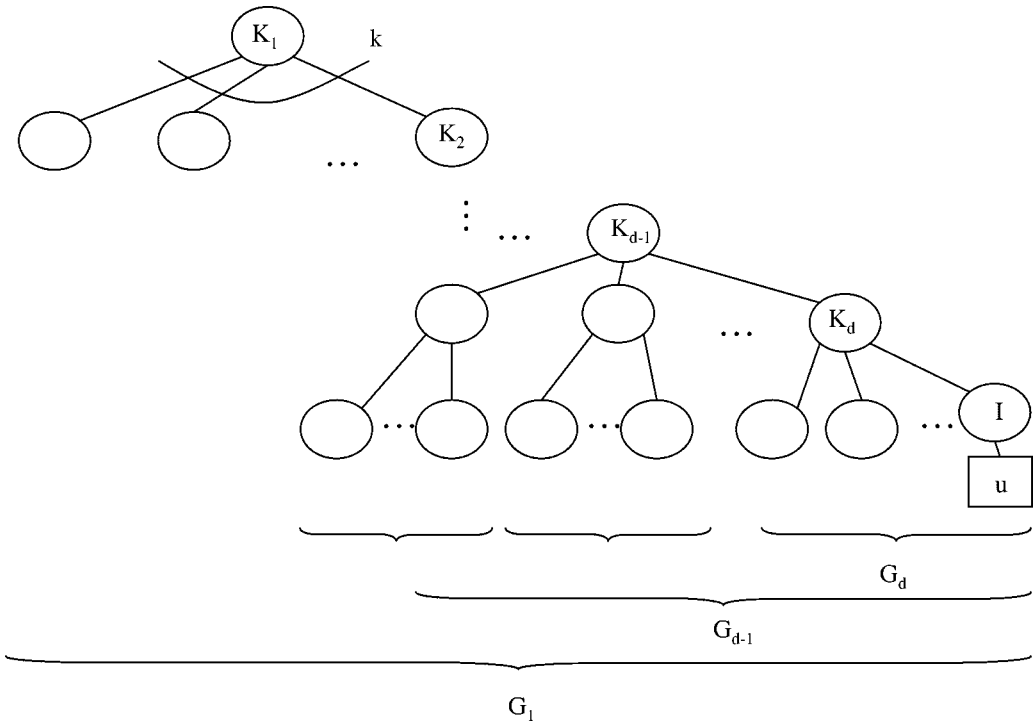

Fig. 2. k-ary key tree.

the entire key tree is a subgroup and each member is assigned to more than one subgroup. For example, member u belongs to group $G_d, G_{d-1}, \ldots, G_1$.

Whenever there is a membership change, apart from the group key, all keys held by the new or former member have to be changed in a bottom-up manner. For example, if u leaves the group, first K_d is changed to a new subgroup key, say K'_d, which is sent to all members who shared K_d with u (i.e., u's sibling in the tree). Since K_d is known by u, the control manager has to encrypt K'_d using each members' individual key and sends it to them by unicast. After sending K'_d, the process can be propagated one level up. Now, K_{d-1} has to be changed. Since K_d is changed to K'_d which is unknown to u, the control manager can encrypt the new K_{d-1} with all subgroup keys including K'_d and send it to all subgroups in the $d-1$th level. The process is repeated upwards one level at a time until it reaches the root where K_1 is changed. Then all keys, including the group key, held by u are changed.

If u is a new member joining the group, in order to guarantee backward secrecy, all the keys from K_d to K_1 have to be changed. Since u knows nothing about the keys in the group, when the control manager changes K_d to the new key K'_d, K'_d can be encrypted by K_d and multicast to u's siblings and unicast to u. Similarly, this process can be propagated upwards one level at a time, with the control manager multicasting the new keys to the subgroups under the key and unicasting the key to u.

If we assume that the key tree is a k-ary full tree, after each membership change, the number of re-key messages per leave or join is proportional to the depth of the key tree, $\log_k N$, where N is the group size. For each leave, each component of the key at each level has to be sent k times (one for each branch). For each join, each component of the key at each level has to be sent twice (one for multicasting to the old members while the other one for unicasting to the new member). Therefore, the number of re-key messages per leave and join is $k \log_k N$ and $2 \log_k N$, respectively.

From the above discussion, we see that if we split the user pool into m equal parts, the number of re-key messages for joins and leaves is reduced to $k(\log_k N - \log_k m)$ and $2(\log_k N - \log_k m)$, respectively. If such a saving offsets the penalty of an increase in the data rate, it is beneficial to split the pool. Indeed, we will see later in the numerical results that the number of splits can be quite large (e.g., $m \geq 10$) and the corresponding bandwidth reduction can be in excess of 20%.

B. Analytic Model

In this section, we analyze the system for the case in which users in a multicast group arrive according to a certain stochastic process with (target) rate λ (req/s). Each user stays in the system with mean duration of $1/\mu$ seconds. Define ρ as the average number of concurrent users in the system given by λ/μ. Let R bits/s be the data rate for a stream.

We consider that the pool of users is equally likely to access the m logical servers and, hence, the average number of concurrent users in a server is $\rho/m = \lambda/(m\mu)$. Denote S as the packet size of a re-key message, and $E[C_{\lambda/m,\mu}]$ as the cost of each server given by the expected number of re-key messages per second. Denote Γ bits/s as the total bandwidth used in the network, which is the sum of the re-key message data rate and multicast data rate. Clearly, Γ is given by

$$\Gamma = mSE[C_{\lambda/m,\mu}] + mR. \tag{1}$$

We are interested in minimizing Γ by adjusting m. To achieve this, we perform the following analysis of the system.

TABLE I
NOMENCLATURE USED IN THE PAPER

Symbol	Definition
ρ	Total average number of concurrent users. $\triangleq \lambda/\mu$ (req./s)
λ	Arrival rate for the users (req./s)
μ	Average service rate
R	Date rate (bits/s)
S	Packet size of a re-key message (assumed constant) (bits)
β	$\triangleq S\mu/R$
m	Number of multicast groups in the system
$J_i, E[J_i]$	Random variable for the number of re-key messages and its expected value, respectively, for a new member joining the system with i concurrent users (msgs)
$L_i, E[L_i]$	Random variable for the number of re-key messages and its expected value, respectively, for a member leaving the system with i concurrent users (msgs)
$E[C_{\lambda,\mu}]$	Expected number of re-key message per second, given arrival rate, λ, and average service rate, μ (msg/s)
Γ	Total traffic or network bandwidth used (bits/s)
σ	$\triangleq \Gamma/R$
k	Branching factor of a key tree

To analyze our system, we consider that the requests arrive according to a Poisson process and the holding time is exponentially distributed. The system can, therefore, be modeled by a Markov process. Let $Q \in \{0, 1, 2, \ldots\}$ denote the system state corresponding to the number of concurrent users at a server. Let π_i be the steady state probability that $Q = i$. It is well known that

$$\pi_i = \frac{\left(\frac{\rho}{m}\right)^i}{i!} e^{-(\rho/m)}. \tag{2}$$

Note that a state change corresponds to a membership change which incurs some re-key exchange overheads (costs) in bits. Let J_i and L_i be the costs for a user joining and leaving the server in state i, respectively. Note that J_i and L_i are random variables depending on where the user joins or leaves the tree and are independent of ρ. Let $E[J_i]$ and $E[L_i]$ be the expected values of J_i and L_i, respectively. We show in Table I the nomenclature used in this paper.

By the steady-state properties of the Markov chain, $E[C_{\lambda/m,\mu}]$ can then be expressed by

$$E[C_{\lambda/m,\mu}] = \frac{\lambda}{m} \sum_{i=0}^{\infty} \pi_i (E[J_i] + E[L_i]) \tag{3}$$

i.e.,

$$\frac{E[C_{\lambda/m,\mu}]}{\mu} = \frac{\rho}{m} \sum_{i=0}^{\infty} \pi_i \left(E[J_i] + E[L_i]\right) \tag{4}$$

$$\triangleq f(m). \tag{5}$$

Therefore, Γ in (1) can be rewritten as

$$\Gamma = mS\mu f(m) + mR \tag{6}$$

or, equivalently

$$\frac{\Gamma}{R} \triangleq \sigma(m) \tag{7}$$

$$= m\beta f(m) + m \tag{8}$$

where β is a dimensionless parameter defined as

$$\beta \triangleq \frac{S\mu}{R}. \tag{9}$$

Equation (8) says that the total network bandwidth is known once $E[J_i]$ and $E[L_i]$ are obtained.

The closed-form expressions for $E[L_i]$ and $E[J_i]$ are intractable. Therefore, we approximate the key tree as a full tree at any time. By considering the interesting case where ρ/m is large, we therefore have $E[L_i] = 2\log_k(i+1) \approx 2\log_k i$ and $E[J_i] = k\log_k(i-1) \approx k\log_k i$, where k is the branching factor of the key tree. We show in Fig. 3 $E[J_i] + E[L_i]$ versus i for $k = 4$. The discrete points represent simulation results while the solid line is the analytical result. Clearly, simulation matches well with our analysis, showing that our approximation is valid. This is true even when the number of users is not large ($i \geq 15$).

Given ρ, $\sigma(m)$ (and hence Γ) in general first decreases and then increases as m increases. The expression of $f(m)$ is still quite complex and does not allow us to derive a closed form for the optimal m^*. Hence, we make further approximations as follows. Observing that the Poisson distribution peaks at its mean, we approximate the distribution as a δ-function at its mean, i.e.,

$$\pi_i \approx \delta\left(i - \frac{\rho}{m}\right) \tag{10}$$

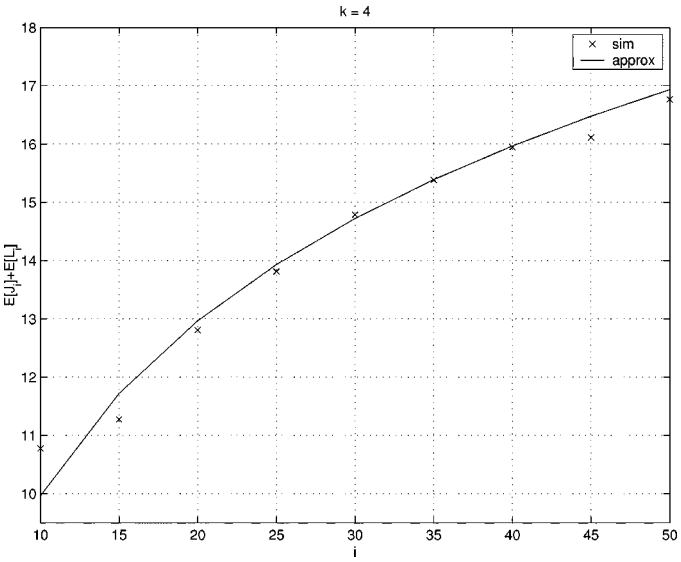

Fig. 3. $E[J_i] + E[L_i]$ versus i $(k = 4)$.

where

$$\delta\left(i - \frac{\rho}{m}\right) = \begin{cases} 1, & \text{if } i = \frac{\rho}{m} \\ 0, & \text{otherwise.} \end{cases} \tag{11}$$

Therefore

$$f(m) \approx \frac{\rho}{m} \sum_{i=0}^{\infty} \delta\left(i - \frac{\rho}{m}\right) (E[J_i] + E[L_i]) \tag{12}$$

$$= \frac{\rho}{m} \left(E[J_{\rho/m}] + E[L_{\rho/m}]\right) \tag{13}$$

$$\approx \frac{\rho}{m}(2+k)\log_k \frac{\rho}{m}. \tag{14}$$

Using it, we have

$$\sigma(m) \approx m\beta\left(\frac{\rho}{m}\right)(2+k)\log_k \frac{\rho}{m} + m \tag{15}$$

$$\triangleq \hat{\sigma}(m) \tag{16}$$

and m^* can be obtained by setting $d\hat{\sigma}(m)/dm = 0$, i.e.,

$$m^* = \frac{\beta\rho(2+k)}{\ln k}. \tag{17}$$

Note that if ρ is greater than $\ln k/((2+k)\beta)$, then $m^* \geq 1$; otherwise, we should use a single server. Furthermore, since $\rho/m^* = \ln k/(\beta(2+k))$, the optimal group size is a constant.

C. Illustrative Numerical Examples and Results

In this section, we present some illustrative numerical results of the secure server network studied. We first estimate the size of S. According to the IETF standard [16]–[18], a re-key message consists of a header (HDR), a sequence number (SEQ), a security association (SA), a key download payload (KD), a signature (SIG), and an optional certificate (CERT). It has been suggested HDR of size 28 bytes, SEQ of size 8 bytes, SA of at least 48 bytes, KD of at least 11 bytes in addition to the size of a key, and SIG of at least 4 bytes. Summing them up, S is at least 99 bytes. Including the key and certificate fields, we hence choose S to be 2 kb in our study.

Since the system parameter β has a determinant effect on the system performance (in terms of total bandwidth consumed), we first show its representative value for some multimedia applications given average user holding time $(1/\mu)$ and data rate of a stream (R) in Table II. We see that β in reality is likely to range quite widely from 10^{-3} (stock quote systems) to 10^{-6} (video applications). In [6], it has been found that for a single server the optimal branching factor k for the key tree is around four independent of the number of users, which is also validated by us using analysis or simulation (results not shown here). Therefore, we will use $k = 4$ in our following study. We consider a baseline system with $\rho = 10^5$ and $\beta = 10^{-4}$, and vary them one at a time in our sensitivity study. We hasten to note that the values of R and μ in the table are examples only. As shown in Section II-B, what affects the system performance is their ratio, i.e., the value of the parameter β. The choice of our baseline β by no means refers to applications with data rate of 5 kb/s. Indeed, some stock quote applications may have a higher data rate due to the presentation of charts or graphs. Our study shows that so long as β remains unchanged, the resulting number of optimal servers remains the same.

We first show the cost advantage in using a server network by plotting in Fig. 4 $\hat{\sigma}(m)/\hat{\sigma}(1)$ (i.e., the ratio of total traffic for a server network with m servers to a single server system) versus m given β ($\rho = 10^5$ and $k = 4$). The horizontal line corresponds to the single server case. For some certain values of β (e.g., $\beta = 10^{-4}$), as m increases, $\hat{\sigma}(m)/\hat{\sigma}(1)$ first decreases gradually to reach a minimum (mainly due to the decrease in re-key messaging), and then increases steadily (mainly due to the increase in total data bandwidth required). There is, hence, an optimal m^* to minimize the total network bandwidth. From the figure, we see that "splitting" the server in an intelligent manner can substantially reduce the bandwidth requirement of the system. On the other hand, for some low value of β (e.g., $\beta = 10^{-6}$ in this case), $\hat{\sigma}(m)/\hat{\sigma}(1)$ monotonically increases, showing that serving the group of users with a single server is optimal. This is mainly because the data rate is too high as compared with the re-key overhead to merit splitting. Therefore, the bandwidth saved for re-key messaging cannot mitigate the increase in the extra multicast data traffic. There is, hence, a "break even" β at which splitting should be done. This is in fact given by (17) when $m^* = 1$, i.e., the break even β is at $\ln k/(\rho(2+k))$, which is equal to 2.3×10^{-6} for our baseline.

We have also compared our analysis with simulation in this figure. The discrete points represent our simulation results while the solid line represents our analysis. Clearly, our analysis matches very well with the simulation, showing the validity of our model. In the remainder of this section, we will use analysis in presenting our results.

We show in Fig. 5 m^* versus ρ using (17). Clearly it is a straight line. Given certain values of β and k, as the external arrival rate (and thereof the number of concurrent users) increases, the user pool should be split into more groups. The group number may range widely from a few to several hundreds. What is worth noting from this result is that, as the underlying arrival rate changes, the number of users served by each server given by ρ/m^* should be kept constant $(= \ln k/(\beta(2+k))$, which is roughly equal to 2300 for the baseline) in order to min-

TABLE II
SYSTEM PARAMETERS FOR DIFFERENT APPLICATIONS

Application	Stock quote system			Internet radio	Near-CD quality MP3 audio	Internet Video
Average holding time	5 mins	30 mins (for part-time day trader)	4 hours (for investor or agent)	2 hours (~ 1 to 2 programs)	30 mins (>7 songs)	30 mins (CNN news)
μ (req./s)	1/300	1/1800	1/14400	1/7200	1/1800	1/1800
R (kbit/s)	5kb/s			16kb/s	96kb/s	256kb/s
β	1.33×10^{-3}	2.22×10^{-4}	2.78×10^{-5}	1.74×10^{-5}	1.16×10^{-5}	4.34×10^{-6}

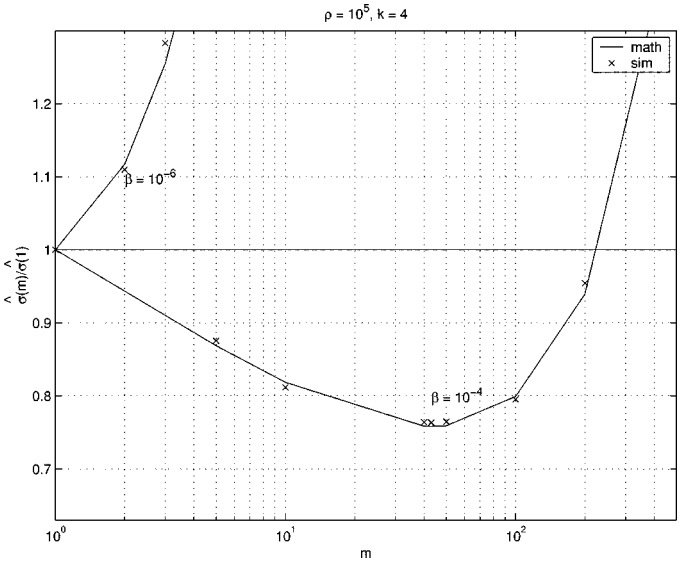

Fig. 4. $\hat{\sigma}(m)/\hat{\sigma}(1)$ versus m, given β ($\rho = 10^5$, $k = 4$).

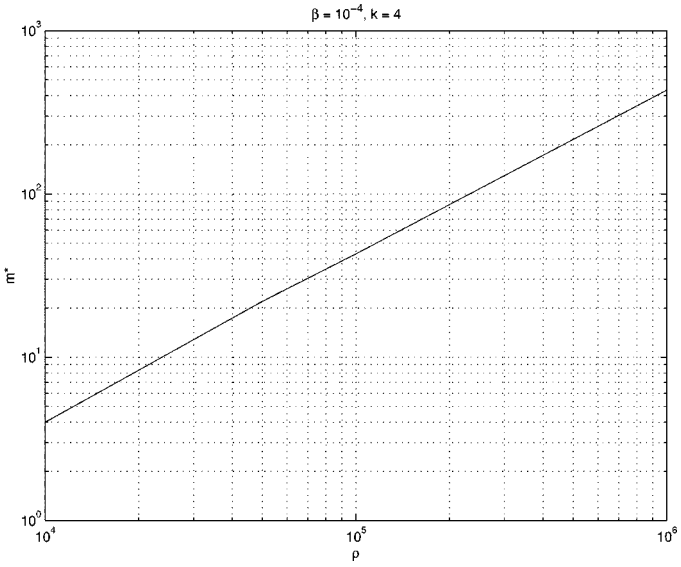

Fig. 5. m^* versus ρ ($\beta = 10^{-4}$, $k = 4$).

imize the overall network bandwidth. Therefore, the server network should execute some split-and-merge mechanism to dynamically adjust m to achieve such optimum, which we will discuss in Section III.

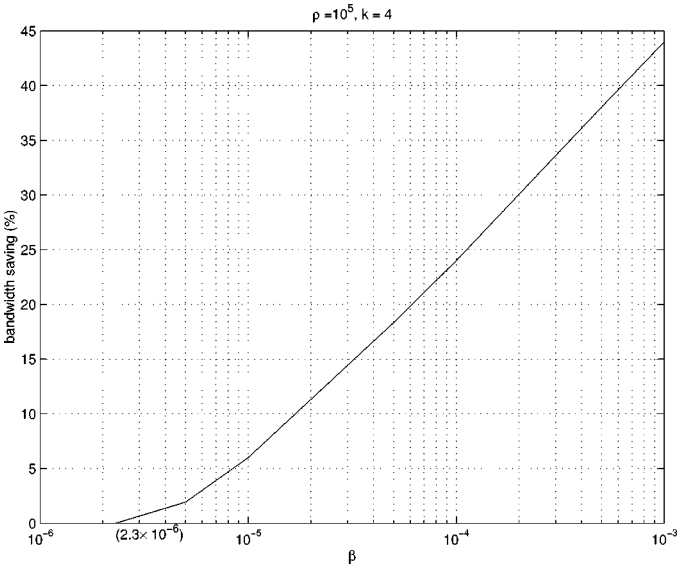

Fig. 6. Maximum saving in bandwidth versus β for a server network as compared with the single server case ($\rho = 10^5$, $k = 4$).

We explore the maximum reduction in network bandwidth of the server network as compared with the single server case. This is shown in Fig. 6 as the maximum bandwidth saving, defined as $1 - \hat{\sigma}(m^*)/\hat{\sigma}(1)$, plotted against β. As β increases, we tend to split the user pool into more groups and the saving, therefore, increases. The saving first increases slowly and then increases somewhat logarithmically with β. In other words, the saving increases with β with a decreasing rate. We see from the figure that with server networks, the network bandwidth requirement can be greatly reduced.

We finally show the optimal ratio of data bandwidth to the total bandwidth of a server (i.e., $1/(\hat{\sigma}(m^*)/m^*) = m^*/\hat{\sigma}(m^*)$) versus β for the distributed servers in Fig. 7 (solid line). We call this ratio "bandwidth utilization." Also shown in dotted line is the single-server case. For the single-server case, the ratio decreases continuously. This is mainly because, as β increases, either data rate and/or user holding time decreases, resulting in a relative increase in the re-key overhead. Note that the single-server approach attains a very low utilization, due to its high re-key overhead. On the other hand, the distributed servers approach increases such utilization tremendously. Its utilization first decreases as with the single-server case when β is less than the break even point of 2.3×10^{-6} (when $m^* = 1$). This is because when β is small, a single server is optimal. However, when

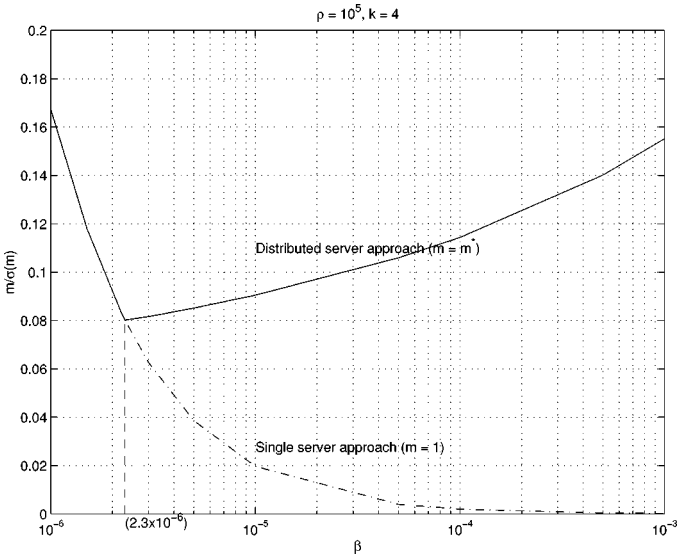

Fig. 7. Ratio of data bandwidth with respect to the total bandwidth of a server at its optimum versus β ($\rho = 10^5$, $k = 4$).

β increases, utilization for the distributed servers approach increases. It is due to the fact that when β is greater than the break even β, splitting reduces the total system bandwidth. Since each server now serves a small user pool, the re-key overhead reduces, which increases the bandwidth utilization.

III. Dynamic Split-and-Merge Scheme

Recall that we have shown in Section II that, in order to minimize the overall network traffic in the system, the number of users served by each server (and hence the group size) should remain constant. As the underlying arrival rate fluctuates, a server should, therefore, dynamically split and merge groups to achieve such optimum. This is the so-called "dynamic split-and-merge" scheme, which we propose and study in this section. The split-and-merge scheme is implemented in a single physical server to achieve load balancing as the underlying traffic fluctuates. Therefore, the scheme incurs little overhead (there is no membership exchange across physical servers) and fits well in a hierarchical distributed servers system. Essentially in this scheme, there is a threshold $\phi_{\max}$ at a server. If the number of users served by the server increases beyond $\phi_{\max}$, we split the user pool into two or more logical servers. By the same token, there is another lower threshold $\phi_{\min}$; if the total number of users in two or more (logical) servers is lower than $\phi_{\min}$, then we merge these groups together. In order to better explain the scheme, we first introduce a simplified version in Section III-A, and then discuss the scheme in details in Section III-A2. We present its analysis in Section III-B and some illustrative results in Section III-C.

A. Scheme Description

1) k-Split-and-Merge Scheme: In this subsection, we introduce a simplified version of our "dynamic split-and-merge scheme," termed "k-split-and-merge scheme." In this scheme, a server is always "split" into k servers, and only k servers can be "merged" together, where k is the branching factor of the key tree.

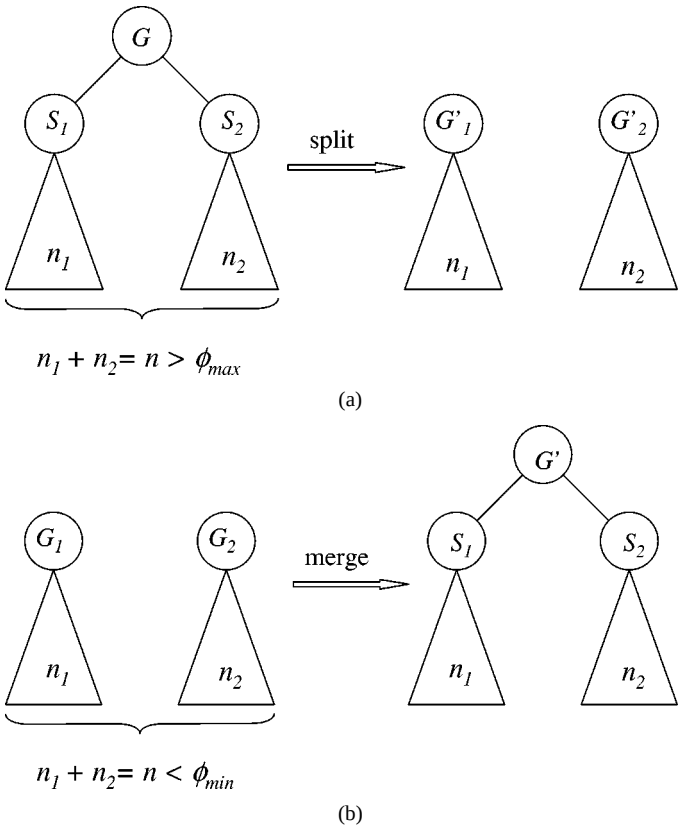

Fig. 8. Examples of merge and split using simple k-split-and-merge scheme for $k = 2$.

The "split" portion of the scheme is described as follows. If the number of users in a server is greater than $\phi_{\max}$, the user pool is split into k groups served by k logical servers. To achieve this, we simply remove the root node from the hierarchical key tree and form k new groups which are subgroups of the original key tree. In this way, the multicast data sent to the new k groups can be encrypted by their subgroup keys from their own servers. The delivery of data is secure because these keys are only known by all members within their respective groups. Since no new key is generated and sent, the re-key cost of splitting is zero.

We next describe the "merge" operation. If there are k servers in which the total number of users is less than $\phi_{\min}$, we merge the groups in these servers into an aggregated group served by only one logical server. To achieve this, we add a new root node and the former root nodes of these groups become the second level internal nodes. Since there is a new group key which needs to be known by all the users, we can encrypt the new group key with each group's original group key and then send it to its corresponding group. Therefore, each time k groups are merged, there are k overhead re-key messages.

Fig. 8(a) and (b) show an example of splitting and merging, respectively, for a binary key tree. If there is a group in which the total number of users N is greater than $\phi_{\max}$, we split the group into two and the original subgroup keys S_1 and S_2 become the new group keys, G_1' and G_2' for the new groups. On the other hand, if there are two groups in which the total N is less than $\phi_{\min}$, we merge them and generate a new group key. The original group keys, G_1 and G_2, become subgroup keys, S_1 and S_2, which can be used to encrypt the new group key G' which is sent to these two groups.

2) Variable-Split-and-Merge Scheme: Now we present the algorithm for the general case in which variable number of servers can dynamically merge and split depending on the number of users they serve. If the number of users in a server is greater than $\phi_{\max}$, we split the server into several servers for which the number of users in a server is as close as possible to the optimal group size ρ/m^*. If there are some servers in which the total number of users is less than $\phi_{\min}$, we merge the groups into a single logical server with the goal of getting as close as possible to ρ/m^*. The problem of finding proper groups to be merged is NP-hard. In order to tackle this problem efficiently, we can use the first-fit decreasing (FFD) bin-packing algorithm as proposed in [19], which is a 11/9-approximation algorithm (Baker gives a proof of this in [20]).

In the FFD algorithm, items are first sorted in decreasing order of size. There are a number of empty bins of size b with increasing index. We place the sorted items into the bins one by one, placing each item in the first bin in which it will fit (i.e., the total size of items in the bin does not exceed b) in a round-robin manner. The time complexity of FFD algorithm is shown to be $O(n \log n)$, where n is the number of items.

We can apply the FFD algorithm for merging servers. We consider each server is an item with its group size as the item size. Imagine that there are many bins with size of $\phi_{\min}$. If we can pack the items into the bins compactly, the number of nonempty bins is very close to the optimal number of servers. Therefore, we should try filling each bin as much as possible. After packing the groups into the bins, we can merge the groups in a bin into a new larger group served by a single logical server.

Regarding servers which need to be merged in a bin, if the number of servers is less than k, then only a new root node is created with all the trees attached to the new root (as in the k-split-and-merge scheme). On the other hand, if there are more than k trees, say n trees, to be merged, we consider the following simple algorithm to merge the trees. In order to keep the new key tree as short and as balanced as possible, we add the taller trees (i.e., a tree with greater depth) into higher level nearer to the root (i.e., level $\lfloor \log_k n \rfloor$) while the shorter ones into the lower level (i.e., level $\lceil \log_k n \rceil$). Fig. 9(a) illustrates a case of merging five servers with a branching factor of $k = 4$. If T_1 and T_2 are the shortest two trees, we should add these two trees into the second level and the taller trees into the first level. The shadowed nodes represent the new nodes created after merging.

We next discuss the splitting of a large group into multiple groups so that the sizes of each of the groups are as close as possible to the optimal size. We first keep splitting the key tree until every split tree is less than $\phi_{\min}$. Then, we use FFD algorithm to merge these trees together such that the sizes of the new trees are close to the optimal group size. Fig. 9(b) shows an example of splitting a key tree with different subtree sizes in the second level. Let n_i be the size of T_i, where $1 \le i \le 4$. We have shown the case where $\sum_{i=1}^{4} n_i > \phi_{\min}$, $\sum_{i=1}^{3} n_i \le \phi_{\min}$ and $n_i \le \phi_{\min}$, where $1 \le i \le 4$. First of all, we split the key tree into four trees. Since $\sum_{i=1}^{3} n_i \le \phi_{\min}$, T_1, T_2, and T_3 are merged together after the FFD process.

Clearly, the variable-split-and-merge scheme is a more flexible scheme than the k-split-and-merge scheme because it allows different combinations of groups to merge together. Therefore, it is more efficient to achieve a group size close to the optimum.

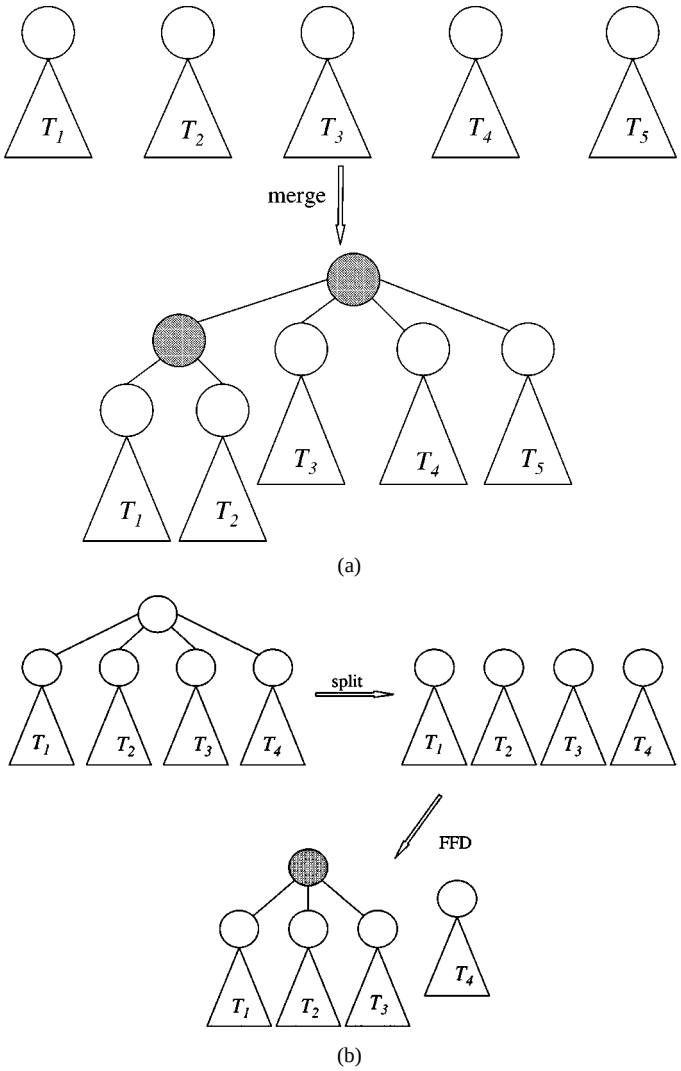

Fig. 9. Merge and split examples using the variable-split-and-merge scheme.

B. Analysis

The analysis of the dynamic split-and-merge scheme is difficult, especially pertaining to the optimization of $\phi_{\min}$ and $\phi_{\max}$; therefore, we have used simulation to study its performance. However, we do obtain some good analytic results on its bandwidth requirement, which we present in this section.

We consider the underlying user traffic as a pseudostationary Poisson process characterized by a series of upward and downward transitions. Essentially, such a function is characterized by a series of step functions as shown in Fig. 10. In this figure, we plot user traffic against time. At time zero, the user traffic is given by ρ. At time t_1, it steps to $\alpha\rho$ (if $\alpha > 1$, then the user traffic is a step-up function; otherwise, it is a step-down function) which maintains till t_2. We further define $\tau = (t_2 - t_1)/t_2$ as the ratio of the duration of traffic staying at $\alpha\rho$ with respect to the total time t_2. In the following, we are mainly interested in the steady state performance of the system given this step function. We assume that the duration of the traffic is sufficiently long so that the transitional overheads of the step is negligible as compared with the steady state. The extension to the general case with multiple step transitions (which is simply a sum of

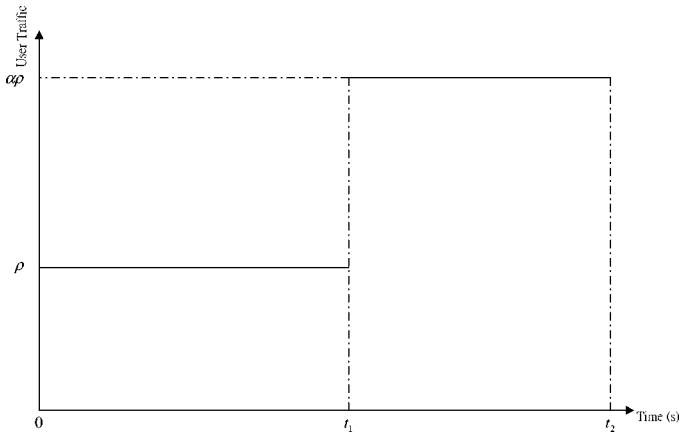

Fig. 10. Step function of variation of user traffic.

step functions) is straightforward and would not be discussed here.

If there were no split-and-merge scheme, the number of servers is kept at m_ρ^* for the whole duration t_2. Therefore, the network traffic of the "static" system, $\tilde{\sigma}_{\text{static}}$, is given by

$$\tilde{\sigma}_{\text{static}} = (1-\tau)\hat{\sigma}_\rho\left(m_\rho^*\right) + \tau\hat{\sigma}_{\alpha\rho}\left(m_\rho^*\right) \tag{18}$$

where $\hat{\sigma}_\alpha$ is given in (16) when the user traffic intensity is α.

With the dynamic split-and-merge scheme, the optimal number of servers after the step is $m_{\alpha\rho}^*$. Hence, the total network traffic for this scheme $\tilde{\sigma}_{\text{dynamic}}$ is given by

$$\tilde{\sigma}_{\text{dynamic}} = (1-\tau)\hat{\sigma}_\rho\left(m_\rho^*\right) + \tau\hat{\sigma}_{\alpha\rho}\left(m_{\alpha\rho}^*\right). \tag{19}$$

We finally define the bandwidth saving using our dynamic scheme as compared with without as $(\tilde{\sigma}_{\text{static}} - \tilde{\sigma}_{\text{dynamic}})/\tilde{\sigma}_{\text{static}}$.

C. Illustrative Numerical Results

We study the variable-split-and-merge scheme with the step function as given above mainly via simulation. Regarding the static case, we use the number of servers optimized according to the traffic at time zero, i.e., ρ. For the dynamic scheme, we adjust the number of servers by using the variable-split-and-merge scheme. The split-and-merge decision is made after a certain number of new users join the system (500 in this study). The baseline parameters are $\rho = 10^4$, $\beta = 10^{-4}$, $\tau = 0.5$, and $k = 4$. We first investigate the influence of the thresholds $\phi_{\max}$ and $\phi_{\min}$ on bandwidth savings. We study the thresholds as ratios to the optimal group size of the current traffic, ρ/m^*.

We plot in Fig. 11 the bandwidth saving versus the ratio $\phi_{\max}(\rho/m^*)$ given α and $\phi_{\min} = 1.2\rho/m^*$. We see that as the ratio increases, the bandwidth saving first increases to a maximum and then decreases. This is because if $\phi_{\max}$ is set to be too low, there are too many split-and-merge overheads which defeats the saving resulting from such operation; on the other hand, if the threshold is set too high, the system is not adaptive to the changes in traffic and, hence, is not of much use. There is an optimal $\phi_{\max}^*$ which leads to maximum bandwidth saving. Such saving increases as α increases (i.e., as the step size increases). Note that $\phi_{\max}^*$ is quite insensitive to α, which should be set approximately 3.6 times of the optimal group size of the current traffic.

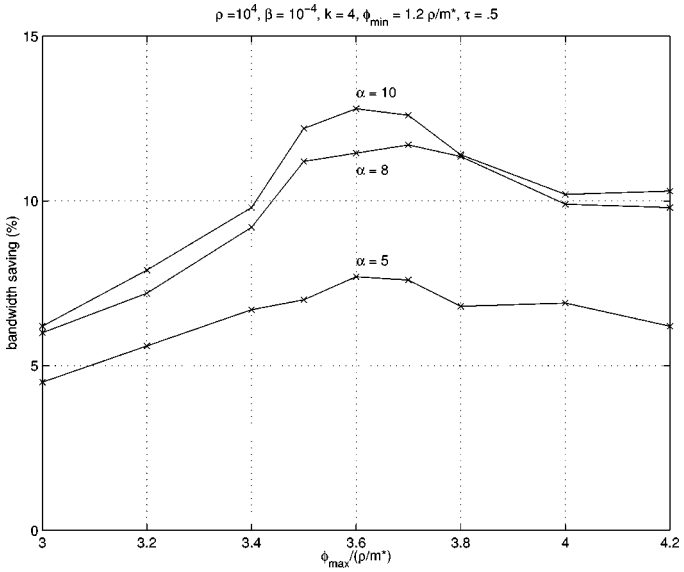

Fig. 11. Bandwidth saving versus $\phi_{\max}/(\rho/m^*)$, given α ($\rho = 10^4$, $\beta = 10^{-4}$, $k = 4\phi_{\min} = 1.2\rho/m^*$).

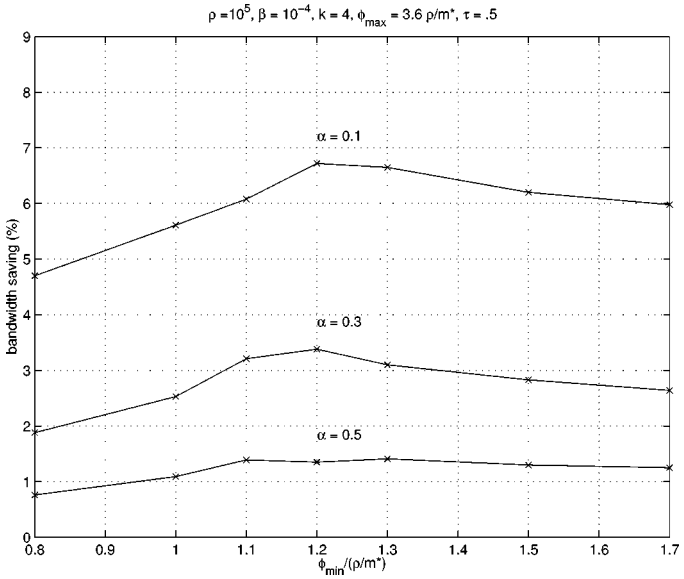

Fig. 12. Bandwidth saving versus $\phi_{\min}/(\rho/m^*)$, given α ($\rho = 10^5$, $\beta = 10^{-4}$, $k = 4$. $\phi_{\max} = 3.6\rho/m^*$).

We next show in Fig. 12, the bandwidth saving versus $\phi_{\min}/(\rho/m^*)$ given α, with the optimal $\phi_{\max}$. The same general trend is observed. The optimal $\phi_{\min}$ is quite independent of the step size, which should be set to be about 1.2 times to the optimal group size.

In Fig. 13, we show the bandwidth saving with the optimized parameters versus α given τ. The discrete points represent our simulation results while the solid lines represent our analytic results, from which we see close match with each other. This not only validates our analysis, but also shows that the overheads in split and merge is low. Recall that $\alpha < 1$ corresponds to a step-down function while $\alpha > 1$ corresponds to a step-up function (The case $\alpha = 1$ corresponds to static user traffic). The saving of the dynamic scheme increases with the step size. For a given α, when τ increases (i.e., the system stays longer

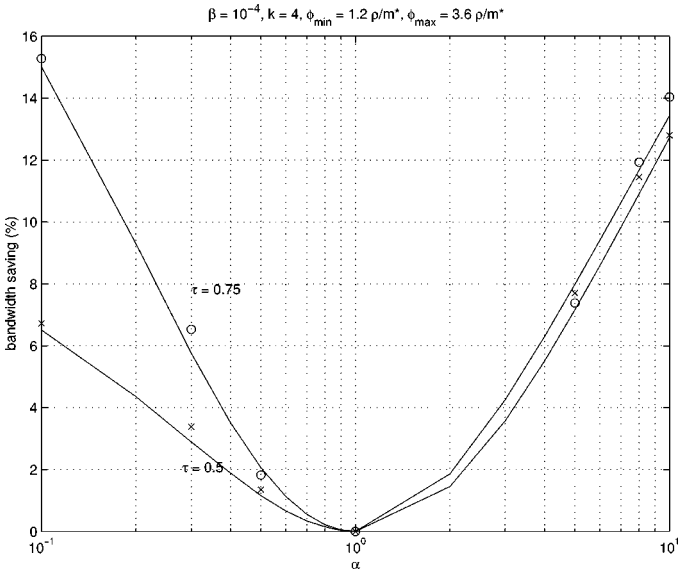

Fig. 13. Bandwidth saving versus α, given τ ($\beta = 10^{-4}$, $k = 4$, $\phi_{\min} = 1.2\rho/m^*$, $\phi_{\max} = 3.6\rho/m^*$).

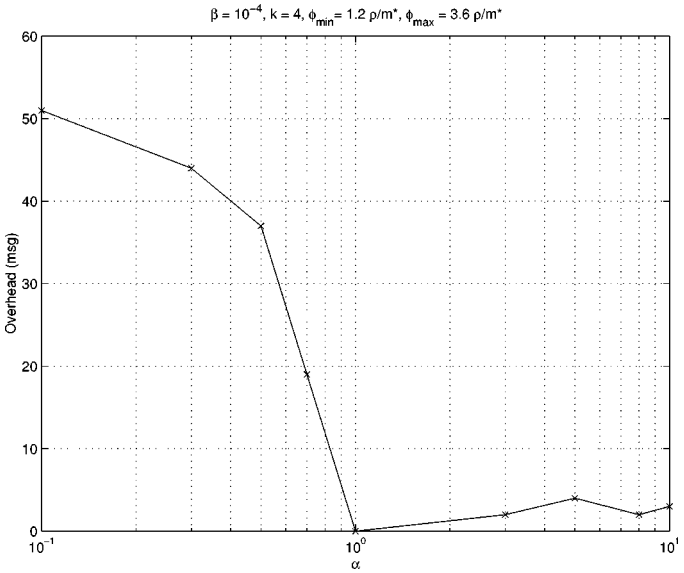

Fig. 14. Overhead versus α ($\beta = 10^{-4}$, $k = 4$, $\phi_{\min} = 1.2\rho/m^*$, $\phi_{\max} = 3.6\rho/m^*$).

at a level different from that was optimized), the saving of the dynamic scheme is higher as it can adapt to the changing traffic. This figure shows that when the underlying user traffic is more dynamic, our scheme achieves lower bandwidth requirement by adaptively splitting and merging serving groups.

We show in Fig. 14, the total overhead defined as the number of extra re-key messages to perform merging and splitting, versus α. As the step size increases, the overheads also increases. The case $\alpha < 1$ consists mainly of merging overhead, while $\alpha > 1$ consists mainly of splitting overhead. We see that merging overhead is much higher than splitting overhead. In any case, both overheads are quite low. These fixed overheads is incurred each time when the underlying traffic changes. We see that when the traffic fluctuates slowly, such overhead can be negligible as compared with the on-going total network bandwidth requirement for re-key messaging.

IV. Conclusion

In order to provide backward and forward secrecy in secure multicast systems, encryption keys have to be changed whenever a user joins or leaves the system. These re-key messages have to be communicated with the existing users in an efficient manner to minimize the overheads. An efficient solution for such key management is a hierarchical key tree approach. However, such an approach still suffers high re-key bandwidth requirement for a large number of users when a single server is used. In order to reduce the bandwidth, complexity and manageability of the system, we propose a distributed servers approach which trades off re-key messaging with some data traffic to achieve minimum total traffic requirement.

We present a simple model for the system and show that there is an optimal number of servers to minimize the bandwidth requirement. Such a number is proportional to the number of users in the system. The bandwidth saving of the system as compared with the single-server case can be quite significant ($\geq 30\%$), especially for the case when the data stream is of lower bandwidth, the average user holding time is short, and the average number of concurrent users is high (e.g., the Internet stock quote applications).

When the underlying user traffic fluctuates, we propose an adaptive scheme to dynamically split and merge the (logical) servers in order to achieve the minimum bandwidth. Our scheme incurs low split-and-merge overhead and achieves further reduction in bandwidth as compared with the static case. Such saving increases when the user traffic is more dynamic.

References

[1] S. E. Deering, "Multicast routing in internetworks and extended LAN's," in *Proc. ACM SIGCOMM*, Aug. 1988, pp. 55–64.

[2] ——, "Multicast Routing in a Datagram Internetwork," Ph.D. dissertation, Stanford Univ., Palo Alto, CA, 1991.

[3] A. Ballardie, P. Francis, and J. Crowcroft, "Core based trees (CBT): An architecture for scalable inter-domain multicast routing," in *Proc. ACM SIGCOMM*, Oct. 1993, pp. 85–95.

[4] S. E. Deering, D. Estrin, D. Farinacci, V. Jacobson, C. Liu, and L. Wei, "An arcitecture for wide-area multicast routing," in *Proc. ACM SIC COMM*, London, U.K., Sept. 1994, pp. 126–135.

[5] W. Diffie, "Authenticated key exchange and secure interactive communication," in *Proc. SECURICOM*, 1990, pp. 300–306.

[6] C. K. Wong, M. Gouda, and S. S. Lam, "Secure group communication using key graphs," *IEEE/ACM Trans. Networking*, vol. 8, pp. 16–29, Feb. 2000.

[7] D. M. Wallner, E. J. Harder, and R. C. Agee, "Key management for multicast: Issues and architectures," in RFC 2627, June 1999.

[8] R. Canetti, J. Garay, C. Itkis, D. Micciancio, M. Naor, and B. Pinkas, "Multicast security: A taxonomy and some efficient constructions," in *Proc. IEEE INFOCOM '99–18th Annual Joint Conf. IEEE Computer and Communications Societies*, vol. 2, 1999, pp. 708–716.

[9] I. Chang, R. Engel, D. Kandlur, D. Pendarakis, and D. Saha, "Key management for secure lnternet multicast using boolean function minimization techniques," in *Proc. IEEE INFOCOM '99–Conf. Computer Communications*, vol. 2, 1999, pp. 689–698.

[10] R. Canetti, T. Malkin, and K. Nissim, "Efficient communication-storage tradeoffs for multicast encryption," in *Proc. Advances in Cryptology—EUROCRYPT '99–Int. Conf. Theory and Applications of Cryptographic Techniques*, 1999, pp. 459–474.

[11] M. J. Moyer, J. R. Rao, and P. Rohatgi, "Maintaining Balanced Key Trees for Secure Multicast, Internet Draft," IETF, draft-irtf-smug-key-tree-balance-00.txt, 1999.

[12] A. Selcuk, A. McCubbin, and D. Sidhu, "Probabilistic Optimization of LKH-Based Mulitcast Key Distribution Schemes, Internet Draft," IETF, draft-selcuk-probabilistic-lkh-00.txt, 2000.

[13] M. Steiner, G. Tsudik, and M. Waidner, "Diffie-hellman key distribution extended to group communication," in *Proc. 3rd ACM Conf. Computer and Communications Security*, Mar. 1996.
[14] ——, "Cliques: A new approach to group key agreement," in *Proc. 18th Int. Conf. Distributed Computing Systems*, Amsterdam, Netherlands, May 1998, pp. 380–387.
[15] C. H. Chiou and W. T. Chen, "Secure broadcasting using the secure lock," *IEEE Trans. Software Eng.*, vol. 15, pp. 929–934, Aug. 1989.
[16] M. Baugher, T. Hardjono, H. Harney, and B. Weis, "The Group Domain of Interpretation, Internet Draft," IETF, draft-ietf-msec-gdoi-04.txt, 2002.
[17] D. Maughan, M. Schertler, M. Schneider, and J. Turner, "Internet security association and key management protocol," in RFC 2048, 1998.
[18] D. Harkins and D. Carrel, "The internet key exchange," in RFC 2049, 1998.
[19] D. S. Johnson, "Near-Optimal Bin Packing Algorithms," Ph.D. dissertation, MIT, Cambridge, MA, 1973.
[20] B. S. Baker, "A new proof for the first-fit decreasing bin-packing algorithm," *J. Algorithms*, vol. 6, pp. 49–70, 1985.

Kin-Ching Chan received the B.Sc. and the M.Phil. degrees in computer science from the Hong Kong University of Science and Technology, Hong Kong, in 1999 and 2001, respectively.

He is currently with Roctec Technology Ltd., Hong Kong, developing networking solutions for the company. During 1998–1999, he was a visiting student of Computer Science Department, University of Minnesota, Minneapolis-St. Paul, MN. His research interest includes multimedia networking, multicast protocols, computer security, cryptography, and coding theory.

S.-H. Gary Chan received the Ph.D. degree in electrical engineering with a minor in business administration from Stanford University, Stanford, CA, in 1999, and the B.S.E. degree (highest honor) in electrical engineering from Princeton University, Princeton, NJ, in 1993.

He is currently an Assistant Professor with the Department of Computer Science, Hong Kong University of Science and Technology, Hong Kong, and an Adjunct Researcher with the Microsoft Research Asia in Beijing. He was a Visiting Assistant Professor in networking at the University of California, Davis, from September 1998 to June 1999. During 1992–1993, he was a Research Intern at the NEC Research Institute, Princeton, NJ. His research interest includes multimedia networking, high-speed and wireless communications networks, and Internet technologies and protocols.

Dr. Chan was a William and Leila Fellow at Stanford University during 1993–1994. At Princeton, he was the recipient of the Charles Ira Young Memorial Tablet and Medal, and the POEM Newport Award of Excellence in 1993. He is a member of Tau Beta Pi, Sigma Xi, and Phi Beta Kappa.