SOT: Secure Overlay Tree for Application Layer Multicast

W.-P. Ken Yiu S.-H. Gary Chan
Department of Computer Science
Hong Kong University of Science and Technology
Clear Water Bay, Kowloon
Hong Kong
Email: {kenyiu, gchan}@cs.ust.hk

Abstract— **Application layer multicast (ALM) has been proposed to overcome current limitations in IP multicast. We address, for the first time, offering data confidentiality in ALM. To achieve data confidentiality, data encryption keys are shared among the multicast group members. Observe that in this system, a node may need to continuously *re-encrypt* packets before forwarding them downstream. Furthermore, keys have to be changed whenever there is a membership change, leading to *re-key* processing overhead at the nodes. For a large and dynamic group, these re-encryption and re-keying operations incur high processing overhead at the nodes. We introduce a scalable scheme called Secure Overlay Tree (SOT) which clusters ALM peers so as to localize re-keying within a cluster and to limit re-encryption at cluster boundaries, thereby minimizing the total nodal processing overhead.**

We describe the operations of SOT and compare its nodal processing overhead with two other basic approaches, namely, host-to-host encryption and whole group encryption. We show that there exists an optimal cluster size to minimize the total nodal processing overhead. SOT achieves substantial reduction in nodal processing overhead with little cost in network performance in terms of network stress and delay.

I. Introduction

Networked multimedia applications such as real-time stock quotes, Internet radio, video conferencing, etc., often require multicast support to efficiently disseminate information to a large group of distributed users. Many of these applications also require data confidentiality for information protection and for charging purpose. In such a secure multicast system, data should be encrypted by the source and decrypted by the group members. A new member should not be able to decrypt any data multicast *before* its joining (the so-called backward secrecy). Similarly, a former member should not be able to decrypt any data multicast *after* its departure (the so-called forward secrecy). In order to offer both backward and forward secrecy, data encryption key has to be changed (i.e., re-keyed) whenever there is a membership change, and the corresponding decryption key has to be made known to all the current members.

Providing confidentiality by network layer multicast has been extensively studied (see, for examples, [1], [2]). However, network layer multicast still has not been widely deployed. Therefore, recent research has been focusing on enabling multicast at the application layer, the so-called application layer multicast (ALM). In ALM, application peers first self-organize into a logical overlay tree. Multicast is achieved by transmitting data from one peer to another along the tree edges using unicast connections. Therefore, ALM does not require multicast-capable routers and hence potentially speeds up the deployment of large-scale multicast-based services. Since data in ALM may be read by a non-member using a network sniffer, the data confidentiality issue is important in ALM.

This work was supported, in part, by the Competitive Earmarked Research Grant (HKUST6156/03E) and Direct Allocation Grant (DAG02/03.EG41) from the Research Grant Council in Hong Kong.

To offer data confidentiality, one may think of two straightforward basic approaches:

- Host-to-host encryption: Each overlay connection on the data delivery tree shares a unique data encryption key. In forwarding packets from one host to another, each host has to first decrypt the packets received from its parent, and then re-encrypt the packets before forwarding them to each of its children using the corresponding encryption key of the connection. Clearly, we see that a node in this approach needs to continuously decrypt and re-encrypt packets. This leads to continuous *decryption/re-encryption* processing overhead depending on packet arrival rate.
- Whole group encryption: All group members share a universal group key, and hence decryption/re-encryption processing is not needed between peers. In this case, whenever there is a membership change, a new group key is generated which has to be communicated and made known to all the members. Such re-key messages have to be processed by all the peers in the network so as to agree on a common new group key. This leads to *re-key* processing overhead depending on how often group membership changes.

We see from above that the two basic approaches represent two extremes in nodal processing overhead to offer data confidentiality: one having high decryption/re-encryption overhead (host-to-host encryption) while the other having high re-key messaging overhead (whole group encryption). Therefore, for a given data rate and group dynamics, either of the approaches would not perform well in terms of processing overhead. A more efficient way is to use a hybrid scheme where the group members are divided into clusters. Whole group encryption is used within each cluster while host-to-host encryption is used between clusters. This strikes a balance between the two processing overheads, thereof achieving lower overhead at each node. Such an overlay tree provides a simple and yet efficient way to offer data confidentiality and, we term it Secure Overlay Tree (SOT).

In this paper, we present a framework on how to build SOT among application peers and compare its performance with the two aforementioned basic approaches. Our results show that, SOT achieves substantially lower overhead (by many factors) with little cost in network performance (in terms of physical link stress and relative delay penalty).

We briefly discuss previous related work here. Traditional multicast protocols such as DVMRP, CBT and PIM-DM, requires the use of multicast-capable routers, making the global deployment of multicast service difficult. Moving the multicast functionality to the application layer, the so-called Application Layer Multicast (ALM), eases the deployment. Many ALM protocols have been proposed in recent years in this regard, [3]–[8]. However, their main concerns are

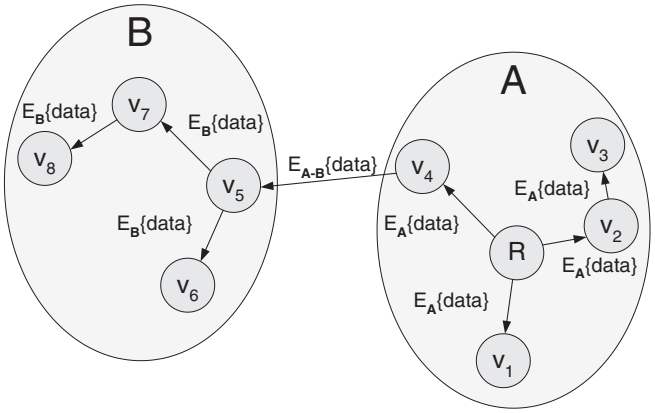

Fig. 1. Only members in the same cluster share the same cluster key. Re-encryption is only necessary when transferring data across different clusters.

connectivity, failure recovery, scalability, etc. None of them addresses the data confidentiality issue.

Logical key hierarchy (LKH) is often used to offer data confidentiality in IP multicast [1]. However, LKH cannot be directly applied in ALM, mainly due to the fundamental difference in overhead accounting. LKH assumes a multicast-capable network, and hence sending one re-key message to all members accounts for only one overhead. Therefore, the re-key message overhead for member joining and leaving is $O(\log_k N)$ and $O(k \log_k N)$, respectively, where k is the degree of the key tree and N is the group size. However, in ALM, the total overhead of sending one re-key message to all members is $O(N)$ which is clearly much larger than LKH for large group. Directly applying the key tree structure of LKH in ALM is hence not efficient in terms of network bandwidth usage and this calls for a different design.

Another approach to offer confidentiality in IP multicast is Iolus [2]. Iolus divides the members into subgroups, each of which is managed by a Group Security Agent (GSA). The subgroup has its own subgroup key; hence, re-keying needs to be performed only where a member joins or leaves. SOT is based on a similar idea. However, Iolus has not considered the impact of subgroup size and nodal processing overhead, which we consider and study here.

This paper is organized as follows: In Section II, we describe the two basic approaches and SOT in details. Then, we show in Section III-A that there exists an optimal cluster size in SOT. In Section III, we first describe our simulation, then we present our results and comparisons. Finally, we conclude in Section IV.

II. Scheme Description

In this section, we first describe in detail the two basic schemes, followed by our proposed SOT scheme.

A. Basic Schemes

We model the topology of the overlay tree of N nodes (hosts) as $T = (V, E)$, where the data source is located at root R. Each node $v_i \in V$ represents a user in the group and each $e_i \in E$ represents a unicast connection between the two end-point users.[1] We use the notation $E_k\{data\}$ to denote the encryption of "$data$" using key "k". As mentioned before, one may conceive the following two basic approaches to offer data confidentiality:

Host-to-host encryption: Two connected members v_i and v_j agree upon a symmetric key called *neighbor key* k_{ij} using key-encrypted-key mechanism (i.e., v_i generates k_{ij} and makes the key known to v_j by encrypting it using v_j's public key). Clearly, a neighbor key has only local significance, i.e., it is associated between a pair of peers only. In this case, data is encrypted as $E_d\{data\}||E_k\{d\}$ where d is a symmetric key randomly generated by the data source and k is any neighbor key between two peers on the data path. Thus, re-encryption is done by decrypting and re-encrypting d rather than the whole data packet. Upon receiving a packet from a member v_i, a member v_j first decrypts $E_{k_{ij}}\{d\}$ using k_{ij} and decrypts $E_d\{data\}$ for its use. It then re-encrypts d using k_{jk} and forwards $E_d\{data\}||E_{k_{jk}}\{d\}$ to the downstream child v_k. By this approach, whenever a member joins or leaves the system, only its parent and children are required to re-key. However, this approach requires per-packet processing on every node for re-encryption. Therefore, the nodal processing overhead is expected to be high for high-bandwidth applications.

Whole group encryption: In contrast to host-to-host encryption, only the source does data encryption using a universal group key g. When a member receives a data packet, it simply relays it to its children without any re-encryption (it certainly needs to decrypt the packet with g for its own consumption). However, whenever one of the group members joins or leaves, the group key has to be changed. This incurs $O(N)$ re-key messages to all the existing N members, who are required to process the re-key messages. Clearly, the nodal processing overhead is expected to be high for dynamic group.

B. Secure Overlay Tree (SOT)

As clear from above, either host-to-host encryption or whole group encryption may not perform satisfactorily given a certain data rate and group dynamics. A more efficient way is to group members into non-overlapping clusters of size m a shown in Fig. 1. We call the tree formed Secure Overlay Tree (SOT). Instead of sharing a group key among all members, members in a cluster share a "cluster key." Whenever a member joins or leaves the group, it actually joins or leaves a cluster. Hence, re-key messages are only delivered within a cluster. Therefore, only $O(m)$ re-key messages are processed for each join/leave. SOT loosely maintains its cluster size by splitting and merging. In order not to incur too much splitting and merging overhead while keeping a rather uniform cluster size, SOT bounds the size of each cluster between $m/2$ and $2m$. Packets are re-encrypted only when they cross the boundary of clusters, and only takes place at the ingress and egress nodes of a cluster. In other words, SOT uses "whole cluster encryption" within clusters and "host-to-host encryption" between clusters.

Members are logically organized into two layers in SOT as shown in Fig. 2: 1) Leader layer and 2) Member layer. Every cluster has a cluster leader, which constitute an overlay network for control messaging, and for coordinating operations such as joining, merging and splitting. Cluster leaders themselves also belong to the Member layer. Note that the protocol suggested here is a framework, hence, the overlay network formed by leaders and those within clusters can be implemented by any existing ALM protocols.[2] The protocol details are described as follows:

1) Internet Coordinate System: We suggest applying an Internet coordinate system like GNP, VL [9], [10], etc. to map the Internet locations of peers into a coordinate system, so that mutual distances between peers can be calculated more easily. Furthermore, cluster

[1] In this paper, "host," "node," "member" and "peer" are used interchangeably.

[2] Since SOT is a flexible framework, we intentionally leave out some implementation details here.

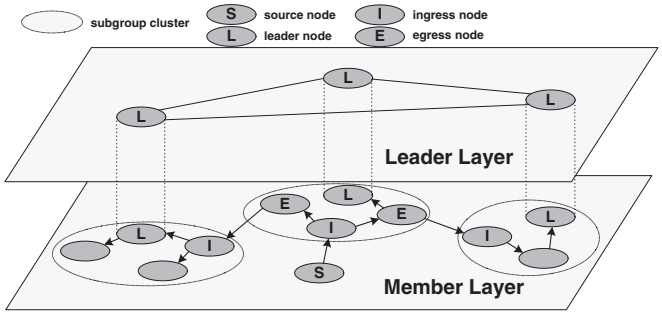

Fig. 2. Two layers are formed in SOT: Leader layer and Member layer. S is the source of data stream. Intra-cluster trees are formed within the clusters while an inter-cluster tree is formed via the connections between ingress and egress nodes.

leaders can summarize the cluster location by calculating the centroid of all member coordinates within its cluster. (If an Internet coordinate system is not available, peers can still use ping to determine their network distances.) Accurate measurements of network distance can improve the efficiency of the data path in terms of latency and bandwidth consumption as well as other maintenance operations such as joining, splitting and merging, etc.

2) Member Joining: A new member can contact any cluster leader to join the group. This can be bootstrapped by a server or Rendezvous Point (RP) keeping the location information of some cluster leaders. When a cluster leader receives a *JoinRequest* which contains the new member coordinates, it first finds out whether any other neighboring leader is closer to the new member. If so, it forwards the request to that closer leader. The process continues until the closest leader to the joining member is found. This determines which cluster the new member belongs. Then, the *JoinRequest* is passed down to the Member layer where the new member forms an ALM tree with the cluster members. The joining process, hence, can be viewed as two independent joining processes, first on the Leader layer and then on the Member layer.

Once the new member x attaches itself on the overlay tree, it forms a secure channel with each of its neighbors using asymmetric encryption. It sends its public key p_x in the *JoinRequest* to its neighbor y. y then generates a symmetric key k_{xy} and sends back a *JoinReply* which contains $E_{p_x}\{k_{xy}\}$. Then, x and y can share the same secret key k_{xy} which is used for updating the cluster key. At the same time, the cluster leader generates a new cluster key k'. It multicasts the re-key message $E_k\{k'\}$ in the cluster where k is the old cluster key. When a peer finds that its downstream peer is a new member, it re-encrypts k' using the secret key shared with the downstream peer. So that, the new member can also obtain k'.

3) Member Departure: The departure of a member can be known by either 1) a *Goodbye* message from the member to its neighbors (i.e., a graceful leave); or 2) an absence of several *HeartBeat* messages to its neighbors (i.e., an ungraceful leave). Whenever a peer detects the leaving of its neighbor, it informs its cluster leader to trigger the re-keying mechanism. After the cluster leader generates the new cluster key k, it multicasts k within the cluster along the overlay tree using host-to-host encryption, i.e., members re-encrypt k using neighbor keys before forwarding it.

4) Cluster Split and Merge: A cluster leader periodically exchanges with its neighboring leaders in the Leader layer *ClusterInfo*, which contains cluster size and member information. SOT keeps the size of each cluster within the range from $m/2$ to $2m$. When a cluster C becomes too large (i.e., greater than $2m$), the leader x triggers the spitting mechanism. As x stores all member locations, it can perform any centralized clustering algorithm based on these locations to split the original cluster C into two parts, C_1 and C_2, each of which containing m members. Suppose x is in C_1 after clustering. It then randomly picks a member y from C_2 as the leader of that cluster and sends to y a *LeaderTransfer* message containing information about C_2 members, so that y can promptly possess all its member information for management.

On the other hand, when a cluster C_1 shrinks and becomes too small (i.e., less than $m/2$), it has to merge with some suitable neighboring clusters. A cluster is considered to be suitable if, after merging, 1) the resultant cluster size is within the size range; or 2) the resultant cluster can then be split into smaller clusters whose sizes are within the range. Once the target cluster C_2 is identified, x sends a *MergeRequest* containing its member information, to the leader y of C_2. y then selects the closest pair: a member s from the member list in the *MergeRequest* and a member t in its cluster, and informs t to connect to s for member discovery according to the underlying ALM implementation. x then becomes the leader of the combined cluster.

5) Leader Election: Cluster leaders also periodically sends *HeartBeat* messages to its neighbors, indicating its presence. Any member finding that its leader has left the group declares itself as the new leader by broadcasting a *NewLeader* message within the cluster. It is possible that more than one member declare themselves as the new leader. In this case, tie is broken by their network addresses.

6) Data Path and Ingress/Egress Selection: An inter-cluster overlay connection is formed by connecting egress and ingress nodes, which are selected by the cluster leaders according to the member locations. Inside each cluster, tree is built and the ingress node is treated as the source of the data stream. The underlying ALM protocol implementation determines how such inter- and intra-cluster trees are built. For example, a DT protocol implementation may make use of compass routing [5]. The inter-cluster tree, together with the intra-cluster trees, form a Secure Overlay Tree (SOT).

In the inter-cluster tree, i.e., the tree connecting all clusters. The leader x of a parent cluster periodically look for suitable members to serve as egress nodes in its cluster for each of its children clusters. According to member locations, the member reasonably close to a child cluster is chosen as the egress node. x then sends an *EgressList* message containing the locations of several good candidates to the leader y of its child cluster. Upon the receipt of *EgressList*, y then finds a good egress-ingress pair and call the ingress node to make a connection with the egress node. After that, the egress-ingress pair establishes an inter-cluster key for re-encryption.

III. Simulation and Illustrative Results

In our simulation, we compare the performance of SOT with the two basic schemes and a recently proposed ALM protocol, namely Delaunay Triangulation (DT) [5], [6]. In our simulation, we use DT for our implementation of SOT in both Leader layer and Member layer.

In our experiments, we first generate a network topology by GT-ITM according to the Transit-Stub model with 1024 routers [11]. Members arrive according to Poisson process with rate λ req./s and stay in the system with exponential holding time of mean $\overline{T}$ seconds. (Clearly, the average number of users in the system is $\rho = \lambda\overline{T}$.) The hosts are randomly assigned to any one of the routers. The source is randomly chosen among the hosts. Each packet is of constant size p bits and the data stream is of rate R bits/s.

A. Optimal Cluster Size

In SOT, the members are divided into clusters. Clearly, each re-keying in SOT requires $O(m)$ re-keying messaging overhead while each multicast packet requires $O(N/m)$ re-encryption overhead, where m is the cluster size and N is the group size. Therefore, to minimize processing overhead, m should be as small as possible for dynamic groups. On the other hand, m should be as large as possible for high-bandwidth applications. Intuitively, there exists a balance point in cluster size ($1 \leq m^* \leq N$) such that the total processing overhead for an application can be minimized. Hence, we conduct experiments that estimating the nodal processing overhead for SOT using different cluster sizes.

We estimate the nodal processing overhead for encryption and decryption used in re-keying and re-encryption in a node. In estimating processing overhead, we normalize each symmetric encryption/decryption as one unit.[3] As symmetric cryptographic operations are often three to five orders of magnitude faster than their asymmetric counterparts, we introduce two factors α and β defined as follows ($\alpha, \beta > 0$):

$$\alpha = \frac{\text{Time for asymmetric } \textit{encryption}}{\text{Time for symmetric } \textit{encryption}};$$

and

$$\beta = \frac{\text{Time for asymmetric } \textit{decryption}}{\text{Time for symmetric } \textit{decryption}}.$$

We measure the speed of commonly-used cryptographic operations on a 800MHz Pentium III Linux workstation using the optimized implementations of the OpenSSL utility program [12]. We find that, e.g., for Rijndael-256 and RSA-1024, $\alpha = 1000$ and $\beta = 17285$.

B. Performance Metrics

In our simulation, we study the following three performance metrics:

- *Average nodal processing overhead*: We estimate the average nodal processing overhead $\overline{H}$ for re-keying and re-encryption in the systems. This metric is defined as:

$$\overline{H} = \frac{\sum_{i \in G} H_i}{\rho},$$

  where H_i is the processing overhead per second incurred in node i, which is a member of group G. The nodal processing overhead is incremented by one when a node performs a symmetric encryption/decryption, by α for each asymmetric encryption and β for each asymmetric decryption. We use this value to measure the processing overhead required per second for a node.
- *Physical link stress (PLS)*: This metric is commonly used in the literature for comparing the network performance of different ALM schemes. PLS is defined as the number of identical packets transmitted over a physical link when a packet is multicast.
- *Relative delay penalty (RDP)*: RDP is another commonly used metric and is defined as the ratio of the delay in the overlay path between a host and the source to the delay in the unicast path.

C. Results

We have chosen a video conferencing application as our baseline system, with parameters R = 256 kbps, p = 1000 bytes, $\overline{T}$ = 30 min., ρ = 10^5, m = 100, α = 1000 and β = 17285.

[3]Encryption and decryption are the same operation in symmetric cryptography.

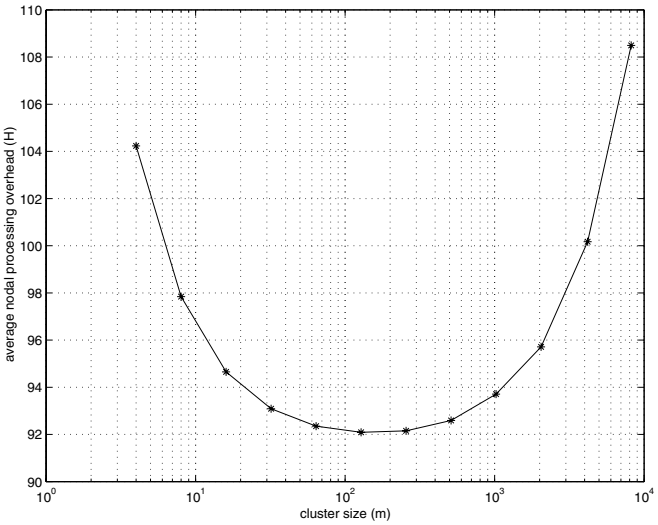

Fig. 3. Nodal processing overhead against cluster size for high-quality video streaming application.

TABLE I
OPTIMAL CLUSTER SIZE OF DIFFERENT APPLICATIONS.
(p = 1000 bytes).

Application	Average holding time ($\overline{T}$)	Data rate (R)	Optimal cluster size (m^*)
stock quote system	30 min.	5 kbps	30
Internet radio	2 hrs.	16 kbps	50
video conferencing	30 min.	256 kbps	100
high-quality video	2 hrs.	1 Mbps	500

In Fig. 3, we plot $\overline{H}$ against m in our system. Clearly, it first decreases (due to a reduction in re-encryption overhead) and then increases again (due to an increase in rekeying messaging overhead). There is an optimal cluster size m^* to minimize $\overline{H}$. Thus, in our baseline system, m should be set around 100.

We find in our study that m^* in fact depends on specific application parameters, i.e., R, p and $\overline{T}$. In Table I, we summarize m^* for different applications. We see that m^* can range from as few as tens of nodes to as many as hundreds of nodes.

We next explore $\overline{H}$ with respect to ρ, given m (m = 100). We compare SOT with the two basic schemes (host-to-host and whole group encryption) in Fig. 4. The overhead for host-to-host encryption is independent on ρ while that of whole group encryption increases with ρ. SOT achieves the minimum overhead and remains at a substantially lower level when ρ increases even to a large value. In our experiments, the gain in average nodal processing overhead can be as much as one to two order of magnitude depending on the values of α and β.

On the other hand, in Fig. 5 and 6, we show that the network performance of SOT in terms of PLS and RDP is comparable to that of the improved version of DT. When the group size is smaller than the cluster size, SOT in fact maintains the whole group as one cluster. Therefore, the network performance is the same as DT. When the group size increases, more overlay paths share the physical links, hence the PLS increases slowly. Also, the paths from the source to

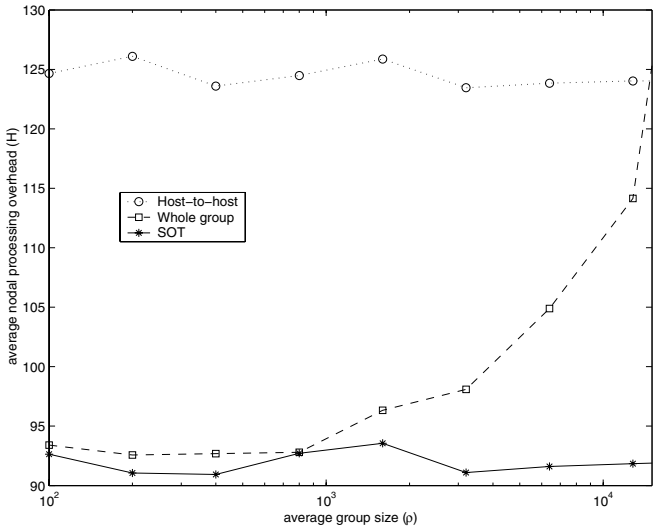

Fig. 4. Average nodal processing overhead against average group size.

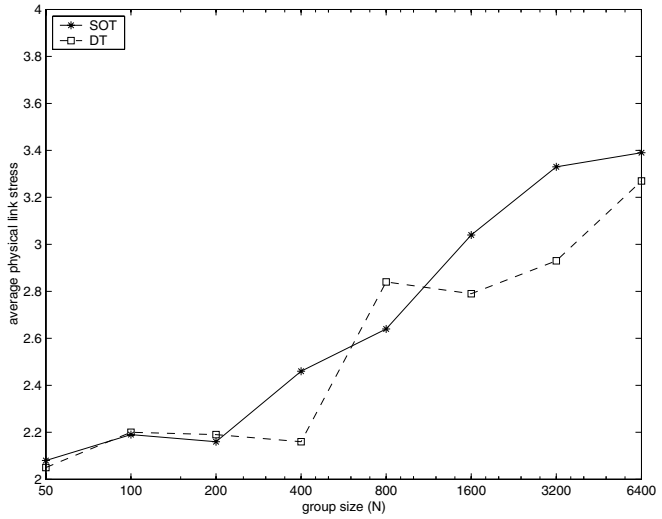

Fig. 5. Average physical link stress against group size.

the peers become longer, hence the RDP increases slowly too. Note that though the physical link stress appears higher than the values as presented in other literature, it is reasonable because we are studying a much larger system (of group size in the order of 1000 rather than 100).

IV. Conclusion

In this paper, we address the data confidentiality issue in application layer multicast (ALM) for the first time. We design a protocol called Secure Overlay Tree (SOT) to provide data security in ALM. SOT is based on clustering members into subgroups of optimal cluster size, which strikes a balance between the re-keying and re-encryption overhead. Compared with the two basic schemes, SOT can achieve much better performance in terms of average nodal processing overhead with comparable network performance (in terms of physical link stress (PLS) and relative delay penalty (RDP)).

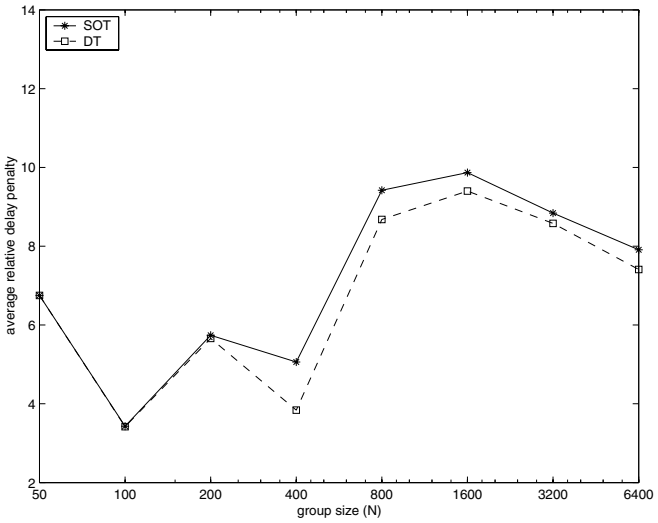

Fig. 6. Average relative delay penalty against group size.

References

[1] C. K. Wong, M. Gouda, and S. S. Lam, "Secure Group Commmunications Using Key Graphs," *IEEE/ACM Transactions on Networking*, vol. 8, pp. 16–29, Feb. 2000.

[2] S. Mittra, "Iolus: A Framework for Scalable Secure Multicasting," in *Proceedings of ACM Sigcomm*, 1997, pp. 277–288.

[3] Y.-H. Chu, S. G. Rao, S. Seshan, and H. Zhang, "A Case for End System Multicast," *IEEE Journal on Selected Areas in Communications*, vol. 20, no. 8, pp. 1456–1471, Oct. 2002.

[4] S. Banerjee, B. Bhattacharjee, and C. Kommareddy, "Scalable Application Layer Multicast," in *Proceedings of ACM Sigcomm*, Aug. 2002, pp. 205–217.

[5] J. Liebeherr, M. Nahas, and W. Si, "Application-Layer Multicasting With Delaunay Triangulation Overlays," *IEEE Journal on Selected Areas in Communications*, vol. 20, no. 8, pp. 1472–1488, Oct. 2002.

[6] W.-C. Wong and S.-H. Chan, "Improving Delaunay Triangulation for Application-level Multicast," in *Proceedings of IEEE Globecom'03*, Dec. 2003.

[7] M. Castro, P. Druschel, A.-M. Kermarrec, and A. I. T. Rowstron, "Scribe: A Large-Scale and Decentralized Application-Level Multicast Infrastructure," *IEEE Journal on Selected Areas in Communications*, vol. 20, no. 8, pp. '1489–1499, Oct. 2002.

[8] K.-F. Wong, S.-H. Chan, W.-C. Wong, Q. Zhang, W.-W. Zhu, and Y.-Q. Zhang, "Lateral Error Recovery for Application-Level Multicast," in *Proceedings of IEEE Infocom 2004*, Mar. 2004.

[9] T. S. E. Ng and H. Zhang, "Predicting Internet Network Distance with Coordinates-Based Approaches," in *Proceedings of IEEE Infocom*, vol. 1, June 2002, pp. 170–179.

[10] L. Tang and M. Crovella, "Virtual Landmarks for the Internet," in *Proceedings of ACM Internet Measurement Conference*, Oct. 2003, pp. 143–152.

[11] K. Calvert, J. Eagan, S. Merugu, A. Namjoshi, J. Stasko, and E. Zegura, "Extending and Enhancing GT-ITM," in *Proceedings of the ACM SIGCOMM Workshop on Models, Methods and Tools for Reproducible Network Research (MoMeTools)*, Aug. 2003, pp. 23–27.

[12] OpenSSL. [Online]. Available: http://www.openssl.org/