Available online at www.sciencedirect.com

ScienceDirect

Computer Communications 31 (2008) 464–488

computer communications

www.elsevier.com/locate/comcom

Scalable multicasting with network-aware geometric overlay

Eng Keong Lua [a,b,*], Xiaoming Zhou [c,d], Jon Crowcroft [a], Piet Van Mieghem [c]

[a] *University of Cambridge, Computer Laboratory, William Gates Building, 15 JJ Thomson Avenue, Cambridge CB3 0FD, United Kingdom*
[b] *Nippon Telegraph and Telephone Corporation (NTT Corporation), NTT Service Integration Laboratories, 3-9-11 Midori-Cho, Musashino-Shi, Tokyo 180-8585, Japan*
[c] *Delft University of Technology, Faculty of Electrical Engineering, Mathematics, and Computer Science, 2600 GA Delft, The Netherlands*
[d] *Philips Research, High Tech Campus 34, 5656 AE Eindhoven, The Netherlands*

Available online 14 September 2007

Abstract

It is crucial to design an efficient network-aware overlay network to enable multicast service to adjust under the dynamic underlying network conditions and node churn in a scalable manner without extensive network measurements. We propose an accurate and scalable Internet subspace geometry to embed the nodes onto a geometric plane by measuring delay latencies between some nodes and assign geometric coordinates to all nodes in such a way that the geometric distances between node coordinates closely approximate their delay latencies. We exploit this Internet subspace geometry to design a network-aware SuperPeers–Peers geometric overlay hierarchy. This is maintained locally in a distributed manner allowing lightweight self-organization. We then create shortest-path overlay multicast tree based on shortest geometric distances between SuperPeers at the SuperPeers layer for overlay multicasting. This low-latency and high-bandwidth multicast backbone infrastructure will serve the Peers in the lower layer. We evaluate our proposals on the 10 massive scale networks each consisting of 100,000 nodes and in the PlanetLab. Our performance evaluation results show *high efficiency* and *good scalability*.

Keywords: Peer-to-Peer overlay networks; Overlay multicast networks; Application level multicast; Network embedding; Internet coordinate systems

1. Introduction

Overlay multicast provides an attractive alternative to IP multicast which has limited wide scale deployment. Overlay multicast is a service that implements multicast on an overlay network, and it offers the benefit of not requiring any network support except the network unicast capabilities. This allows service diversity and accelerated service deployment with a greater flexibility because we do not need to rely on the restriction or availability of the underlying IP multicast's network resources. That is, we are able to implement multicast service over the overlays even though we have limited IP multicast deployment in the underlying network.

The basic principle of building overlay multicast network is to organize nodes into multicast groups at a higher abstraction layer above the underlying network infrastructure. One-to-many or many-to-many communication is performed by building data delivery trees on the overlays. Building an overlay network for multicasting on a random relationship relative to the underlying physical network topology could result in a loosely coupled connectivity between overlay nodes. The impact on the multicast service is that the delay latency could be high as data may pass through many overlay nodes that might be slow and have long physical paths in the underlying physical network. For multicast applications to experience better quality of service characteristics, short network paths are desirable in addition to good load and data distribution. Thus, it is important to mitigate multiple overlay edges from

* Corresponding author. Address: Nippon Telegraph and Telephone Corporation (NTT Corporation), NTT Service Integration Laboratories, 3-9-11 Midori-Cho, Musashino-Shi, Tokyo 180-8585, Japan. Tel.: +81 422 59 3946; fax: +81 422 59 5671.

E-mail addresses: eng.keong-lua@cl.cam.ac.uk, lua.engkeong@lab.ntt.co.jp (E.K. Lua), x.zhou@ewi.tudelft.nl, xiaoming.zhou@philips.com (X. Zhou), jon.crowcroft@cl.cam.ac.uk (J. Crowcroft), p.vanmieghem@ewi.tudelft.nl (P.V. Mieghem).

0140-3664/$ - see front matter © 2007 Elsevier B.V. All rights reserved.
doi:10.1016/j.comcom.2007.08.046

traversing the same underlying network links and multiple communications across many end systems that will produce redundant traffic and increase latency.

To design *scalable* network-aware overlay network, we could initiate multiple network measurements to determine node-to-node latencies. However, such a exhaustive method will result in a large measurement overhead in overlay operation and when node churn is high. If we perform network measurements periodically, the resulting measurement is not related to the realistic usage of the overlay network and thus leads to stale and inconsistent information. In addition, to construct *efficient* overlay multicast streaming service, we cannot ignore the dynamic underlying network behavior and incur unnecessary extensive measurements affecting the quality of the connections between network nodes that cause overheads. The overlay multicast streaming network should not add extra complexity that results in scalability problem and complex maintenance operation.

With all the above criteria in mind, we propose a network-aware geometric overlay multicast streaming network that efficiently exploit the underlying network locality and proximity of nodes for overlay node placement strategy, routing and multicasting. In our approach, we exploit our accurate and scalable network subspace embedding (*Highways* [1]) of network latencies between nodes, such as Round-Trip-Times (RTTs), into a low-dimensional geometric space. This is done by measuring the network latencies between *some* nodes and assign geometric coordinates to *all* nodes in such a way that the geometric distances between node coordinates closely approximate their RTTs. The measurement overhead is reduced because non-measurements are estimated. The node coordinates can be maintained in a distributed manner with a small number of network latency measurements. The network embedding system adapts to dynamic network changes as the overlay nodes update their node coordinates iteratively. Hence, network embedding provides a feasible and promising approach to help in the scalable construction of network-aware geometric overlay network.

Of course, we require a reasonably good quality network embedding function to provide accurate mapping [2–4]. From the scalability meta-metric in [2], subspace embedding (Euclidean space) in small clusters [1] consisting of closer nodes will achieve better embedding accuracy. We design an overlay control plane service using this subspace embedding (*Highways*) to compute node local geometric coordinates in local geometric space and distances between the nodes within each cluster and perform basis transformation of local geometric coordinates to global geometric coordinates in global geometric space for computation of distances between nodes in different clusters. The distance ranking of nodes determine the node locality in the underlying network. Basically, the overlay network learns of the underlying network delay latencies (RTTs) between nodes through their coordinates and distances computed from their coordinates in the geometric plane. This information allows efficient and selective placement of nodes in the overlay topology.

A network-aware SuperPeers–Peers geometric overlay hierarchy is created to scale the overlay network communication and management – Lightweight SuperPeers Topologies (LST) [5–7]. The SuperPeers layer provides a low-latency and high-bandwidth backbone infrastructure for communications among all nodes in the network. The Peers in the Peers layer are connected to their closest SuperPeers in terms of their shortest geometric distances between them. We use *Yao-graph* [8] to build the SuperPeers' connectivity – every SuperPeer is connected to *six* closest SuperPeers (neighbors). The geometric structure of *Yao-graph* is maintained locally in a distributed manner allowing lightweight self-organization and recovery from node churn. We then create source routed overlay multicast tree from the resulting optimal geometric routing paths that are based on the shortest geometric distances (latencies) between SuperPeers at the SuperPeers layer. The optimal geometric routes are derived from the LST's geometric routing protocol (localized random compass routing algorithm) at the SuperPeers layer for multicasting. As every SuperPeer has only six other connections at the SuperPeers layer, this means that the bandwidth consumed by each SuperPeer for multicasting is being reduced to certain extent. This low-latency and high-bandwidth multicast backbone infrastructure will serve the Peers in the lower layer (*Bos*) [9,10].

In order to examine the performance of LST's and *Bos*, we conduct simulation experiments on the 10 massive scale networks each consisting of 100,000 nodes and carry out deployment experiments in the PlanetLab [11]. The performance evaluation results show that our proposals have *high efficiency* and *good scalability*.

Outline. The rest of the paper is organized as follows. Section 2 describes the design principles of our network-aware SuperPeers–Peers geometric overlay network (LST) using accurate and scalable Internet subspace geometry (*Highways*). We also discuss the overlay maintenance management and robustness during node churn. Section 3 presents our proposal of exploiting the network awareness and locality of LST overlay network for overlay multicasting (*Bos*). Section 4 explains the setup of our simulation experiments using the same realistic massive scale networks that were used by Scribe [12,13] and SplitStream [14] from the Microsoft Research Cambridge. We also discuss our deployment experiments in the PlanetLab's real-world Internet-scale environment. Section 5 discusses our performance evaluation results of LST and *Bos* on the 10 massive scale networks and in the PlanetLab. Section 6 describes related work and Section 7 concludes.

2. Network-aware geometric overlay network

The LST overlay network [5,6] is divided into two layers: SuperPeers and Peers. The upper SuperPeers layer acts as a reliable low-latency and high-bandwidth backbone

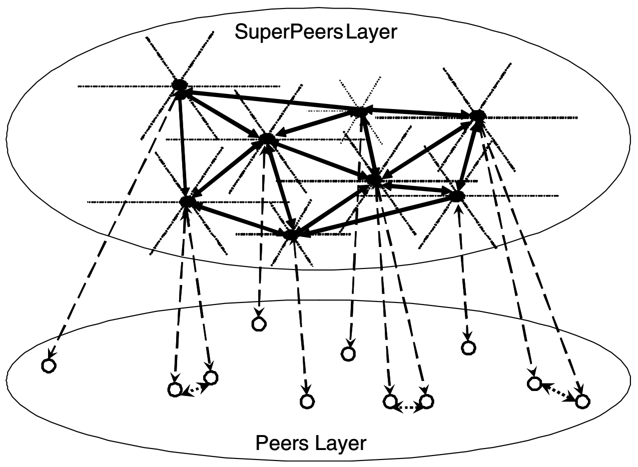


Fig. 1. Network-aware SuperPeers–Peers geometric overlay hierarchy.

network infrastructure for communications among all nodes. Every Peer in the Peers layer is connected to their closest SuperPeer in terms of the shortest geometric distance between them for end-to-end overlay geometric routing. Fig. 1 shows an example of the network-aware SuperPeers–Peers geometric overlay hierarchy of the LST overlay network.

There are three key steps in the construction of the LST overlay network which we will describe in the below subsections.

2.1. SuperPeer election

When a new node joins the LST overlay network either during bootstrap or normal overlay operation, the following criteria are evaluated to determine whether this node will be elected as a SuperPeer or normal Peer:

- The SuperPeer should have *sufficient* resources to serve other SuperPeers and Peers.
- The SuperPeer should be *reliable* and it is not joining and leaving the LST overlay network very frequently.

With the above criteria imposed, the SuperPeers layer consists of SuperPeers acting as low-latency and high-bandwidth backbone infrastructure for the Peers in the Peers layer. Trust, security and incentive schemes could be implemented in this election process, although this is out of scope for this work.

2.2. Internet geometry

Network embedding computes node coordinates and geometric distances between nodes to estimate their underlying network performance metrics such as latency in a scalable way. That is, nodes are mapped onto points in a geometric space and they are assigned geometric coordinates in such a way that the computed geometric distances between node coordinates closely approximates the latencies (i.e. RTTs) between nodes. These node coordinates also reflect their geometric position in the geometric space. RTT measurements from each node to some landmarks are performed for embedding into a geometric space. So, it does not require full mesh N^2 network measurements of N nodes that will cause extensive overheads to deduce the quality of the network connection metric (RTTs) between nodes.

Using these geometric distances, efficient and selective placement of nodes in the geometric overlay network is done. In other words, we are able to determine node locality from the node coordinates and geometric distances between nodes. Basically, the LST overlay network learns of the underlying network latencies (RTTs) between nodes through their coordinates and computed geometric distances. Our Internet geometry provides the geometric location information for the construction of network-aware geometric overlays. From the scalability meta-metric observations in [2], subspace embedding (Euclidean space) in small partitioned clusters achieves better embedding accuracy. The nodes have closer landmarks within each cluster.

We propose and develop *Highways* [1] as an overlay network control plane service providing geometric location information for the LST overlay network. In *Highways*, superspace embedding [2] embeds the whole set of overlay nodes in the system as one large set into *global* geometric space while subspace embedding embeds small partitioned clusters of overlay nodes into *local* geometric space. In this manner, both the global and local geometric spaces are established to derive the global and local geometric positions of all overlay nodes, respectively. We use Euclidean space as the embedded geometric space in our Internet geometry.

Fig. 2 shows the SuperPeers being classified as the list of landmarks for network superspace embedding to compute the node global geometric coordinates in the global geometric space. Fig. 3 illustrates the subspace embedding

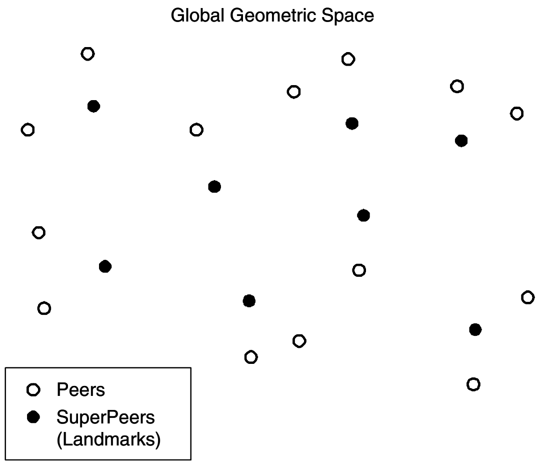


Fig. 2. Global geometric space for overlay nodes.

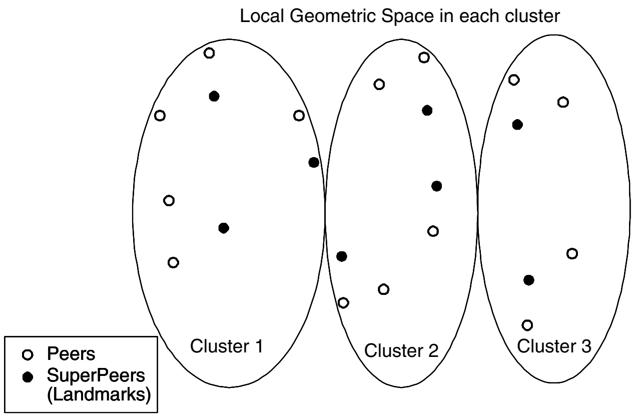


Fig. 3. Local geometric space for overlay nodes.

(Euclidean space) in each of the small cluster consisting of closer nodes.

The local geometric position information helps to provide an accurate geometric distance estimation between overlay nodes within each of the clusters while the global geometric position information estimates the inter-cluster geometric distances between overlay nodes in different clusters. The overlay nodes in each cluster compute their local geometric coordinates based on its local basis of the local geometric space. Once the node local geometric coordinates are computed through subspace embedding, we maintain a basis transformation matrix for the ease of converting the node local geometric coordinates from the local geometric space to the global geometric space to derive the node global geometric coordinates, without measuring any property between itself and the landmarks that spans such a space.

In our system, we first measure the RTTs among the SuperPeers and use this measured RTT matrix to partition the SuperPeers into smaller clusters by adopting a simple approach of the K-means method [15]. The algorithm separates and combines nodes into clusters in the LST overlay network by assigning each node to the cluster having the nearest centroid (mean) based on the geometric distances. We use $K = 3$ as the number of partitions due to the geographical continents of the world, which comprises generally of North/South America, Africa/Europe and Asia Pacific. After all the SuperPeers have been partitioned into smaller clusters, all Peers measure their RTTs to all the SuperPeers and every Peer uses this information to find the closest SuperPeer. The Peer then joins the cluster whereby the closest SuperPeer belongs. These measured RTTs between SuperPeer-to-SuperPeer and Peer-to-SuperPeer as a result of clustering are usable and required for the network embedding. Thus there is no additional overhead and redundancy in measuring these raw RTT measurements and they are less than N^2 measurements taken.

Subspace embedding into Euclidean space is performed strictly in each of the partitioned clusters to compute node local geometric coordinates in each cluster. To compute the inter-cluster geometric distances between all nodes residing in different clusters, we make use of the basis transition matrix. We would be able to transform the node local geometric coordinates from its local geometric space to the global geometric space to derive the node global geometric coordinates. Once the transformation is done, we are able to compute the inter-cluster geometric distances between these nodes residing in different clusters that spans the global geometric space. As a result, all overlay nodes will have both local and global geometric coordinates being assigned.

Here we describe the landmark-based embedding and singular value decomposition (SVD) technique that we use to map the nodes into points in a low-dimensional geometric space. In our LST overlay network, the list of elected SuperPeers are the landmarks for the subspace embedding.

To calculate and assign coordinates of k-dimensional geometric space for all N nodes in X, *at least* $k+1$ landmarks [16,17] (SuperPeers) are selected. This is to solve the possible problem that coordinate vectors of the landmarks could be linearly dependent in the geometric space which may cause the nodes to unable to differentiate their distinct geometric locations from these landmarks and hinder the computation of the node coordinates. That is, if the landmarks have their coordinate vectors as a multiple of the other i.e. the landmarks are in a straight vector line, then the nodes would not be able to compute their distinct geometric locations from these landmarks. As in [18], this framework relies on a set of landmarks from which the nodes may select any set consisting of at least $k+1$ landmarks out of a list of all landmarks for embedding into k-dimensional geometric space. This allows flexibility for a node to determine its geometric coordinates in choosing its set of landmarks without the need to use a fixed infrastructure of well-known landmarks. It solves the problems of communication bottlenecks and single points-of-failure caused by the use of well-known landmarks. However, note that the chosen set of landmarks must share at least one common landmark in their selection so that the vector basis constructed spans the embedded geometric space.

A symmetric measured network latency matrix D for the set of landmarks $L = \{l_1, l_2, \ldots, l_m\}$ is derived as $D = [d(i,j)]_{i,j=1,\ldots,m}$, where $m \geqslant k+1$ and $d(i,j)$ is the measured network latency (RTTs) between m landmarks. For $i \neq j$, $d(i,j) = d(j,i)$ and $d(i,i) = 0$. Dimensionality reduction to k is done using SVD

$$D = U \cdot W \cdot V^{\mathrm{T}} \tag{1}$$

where U and V^{T} are orthogonal matrices, and W is a diagonal matrix containing the singular values of D.

The RTT measurements of all overlay nodes $i \in X$ where $i = 1, \ldots, N$ to their sets of selected landmarks $L = \{l_1, l_2, \ldots, l_m\}$ are made. This can be written using column vector notation for a node i as below:

$$\phi(i) := \begin{pmatrix} d(i,l_1) \\ d(i,l_2) \\ \vdots \\ d(i,l_m) \end{pmatrix} \tag{2}$$

This is simply the Lipschitz embedding of X using set of landmarks L.

By using the first k columns of U denoted by Uk, we project the m-dimensional space into a new k-dimensional space: where $\phi'(i) = Uk^{\mathrm{T}} \cdot \phi(i)$ is the coordinates of node i after dimensionality reduction.

To minimize the discrepancy between the distance represented in the coordinates system and the measured distance between m landmarks, we defined and used a scaling factor: $\alpha_k = \frac{\sum_i^m \sum_j^m d(i,j).\delta_(i,j)}{\sum_i^m \sum_j^m \delta(i,j)^2}$ where $\delta(i,j) = L_2(Uk^{\mathrm{T}} \cdot \phi(i),\ \ Uk^{\mathrm{T}} \cdot \phi(j))$, where L_2 is the Euclidean norm since we use Euclidean space.

In order for a node to know about the global geometric space G and derive its coordinates in G, without measuring any property between itself and the nodes that form such a vector basis, a basis changing technique is adopted and a basis transition matrix T_G is maintained. That is, a basis transition matrix T_G is computed for converting the node local geometric coordinates between its local basis of the local geometric space C to global basis of the global geometric space G to derive its global geometric coordinates. The basic insight is that a randomly selected set of landmarks defines an embedding geometric space that can be easily (linearly) mapped into another embedding space derived from a different set of landmarks. We maintain a basis transformation matrix for the ease of converting node local geometric coordinates from its local geometric space to the global geometric space to derive its global geometric space, without measuring any property between itself and the landmarks that spans such a space.

If we want to change the local basis of $\mathbb{R}^k$ from local geometric space C to the global geometric space G, the basis transition matrix T_G is calculated by a selected arbitrary set of nodes. This selected set of nodes measure coordinates to two landmark sets in the global geometric space G and local geometric space C. Then, the following equation is solved using least squares to obtain T_G

$$\begin{aligned} T_G \cdot P_C &= P_G \\ T_G &= P_G \cdot P_{C^{-1}} \end{aligned} \tag{3}$$

where P_G is the selected set of node global geometric coordinates in global geometric space G and P_C is the selected set of node local geometric coordinates in original local geometric space C.

Once we have T_G, then we can calculate the global geometric coordinates of the node i in the global geometric space G from its local geometric coordinates in original local geometric space C:

$$\phi_G(i) = T_G \cdot Uk_C^{\mathrm{T}} \cdot \phi_C(i) \tag{4}$$

Therefore, the node global geometric coordinates in the global geometric space G can be obtained relative to the basis transition matrix T_G and its node local geometric coordinates in the original local geometric space C, with nothing more than the information it already has.

We expect nodes to recompute their coordinates from time to time due to node churn or topology changes. Such changes are captured by the network performance metrics (i.e. RTTs) between nodes. In this case, a node recomputes its coordinates following the above embedding steps. If for some reasons, a landmark becomes unavailable during this recomputation process, the node then chooses other alternative landmark to devise the basis transition matrix.

2.3. SuperPeers–Peers geometric overlay hierarchy

A brief description of *Yao-graph* [8] and its attractive geometric properties are in Appendix.

The basic idea is that we construct *Yao-graph* [8] at the SuperPeers layer by cutting the two-dimensional Euclidean space around each SuperPeer into *six* sectors, each with equal geometric angle of $\theta = \frac{\pi}{3}$. Every SuperPeer in the *Yao-graph* chooses the *closest* SuperPeer (neighbor) in terms of the shortest geometric distance to other SuperPeer in each of the *six* sectors. So, every SuperPeer is connected to *six* closest SuperPeer neighbors.

The *Yao-graph* is proven to exhibit the Euclidean minimum spanning tree (EMST) in [8]. From the Definition 6 of EMST and Theorem 7 of *Yao-graph* as shown in Appendix, we can see that *Yao-graph* is a type of Euclidean geometric graph with lightweight maintenance characteristic and efficient spanning tree property.

Previous work [19,20] use *Yao-graph* in mobile wireless networks and their good communication performance results motivate our usage of the *Yao-graph* geometric structure. Such graph structure is able to minimize overhead during overlay maintenance management. *Yao-graph* can be maintained locally in a distributed manner because each node is connected to other *six* closest neighbors based on the shortest geometric distances between nodes. That is, the local maintenance algorithm is confined to the affected node and its immediate *six* closest neighbors. In this manner, this geometric graph structure allows efficient and lightweight local recovery from node churn. In addition, *Yao-graph* was the first technique to break the $O(N^2)$ time complexity barrier in the computation of the EMST in a connected graph with N nodes [8].

Here we describe our self-stabilizing and distributed *Yao-graph* construction protocol as shown in Fig. 4. We consider a connected graph $G(V,E)$, where V corresponds to a set of points (nodes) in the Euclidean space $\mathbb{R}^2$, and E to the set of edges with weight corresponding to the Euclidean length of an edge. Suppose that every node $u \in V$ knows its neighborhood $N(u)$ and the current positions of the nodes in $N(u)$ in the Euclidean space. Every node aims at maintaining a connection to the closest node

```
YAO-GRAPH($V, E$)
1  while Node $u \in V$
2  do for Every Node $w \in E(u)$
3      do if Node $v \in N(u)$ in $w's$ sector with $\|uv\| < \|uw\|$
4             then Remove $w$ from $E(u)$
5      for Every Sector $S$ of $u$
6      do if $S$ has at least one node in $N(u)$ but no node in $E(u)$
7             then Add the node $w$ in $S$ of shortest distance to $u$ to $E(u)$
```

Fig. 4. A distributed and self-stabilizing *Yao-graph* topology construction protocol.

in every sector S (or cone). Let $E(u)$ be the current set of the connections of node u.

Theorem 1. *When the Yao-graph distributed protocol self-stabilizes in the stable state, the out-degree of every node is at most s where the two-dimensional Euclidean space around every node $v \in V$ is cut into s sectors with angle $\theta = \frac{2\pi}{s}$.*

Proof. Follows directly from the distributed and self-stabilizing protocol. In our two-dimensional Euclidean space, the out-degree of a node is 6 since every node connects to 6 closest neighbors in the 6 sectors with angle $\theta = \frac{\pi}{3}$. □

We use *Yao-graph* distributed protocol to build the overlay network connectivity among the SuperPeers based on their geometric coordinates and distances with other SuperPeers. These **SuperPeer–SuperPeer Yao-graph routes** serve as the reliable low-latency and high-bandwidth backbone network connectivity for the overlay network. This architecture is illustrated in Fig. 5 which gives the flat two-dimensional geometric view of the overlay network.

In the Peers layer, Peers are directly connected to the *closest* SuperPeers that are capable of serving an additional Peer and this connectivity is called the **Peer-SuperPeer 1-Hop route**. Among the Peers being served by their closest SuperPeer, direct connectivity between these Peers can be established if there exists a shortcut route between the Peers. That is, a **Peer–Peer Shortcut route** is established between two Peers belonging to a SuperPeer, if the direct connectivity between these two Peers is shorter than their Peer-SuperPeer 1-Hop routes. All these routes are easily established in the overlay network using the geometric location information. Examples of various overlay routes in the LST overlay network are illustrated in Fig. 6.

2.4. Geometric overlay routing

We use the *localized* geometric routing algorithm – random compass routing [21] – to route data from one SuperPeer to destination SuperPeer in the *Yao-graph* at the SuperPeers layer. For end-to-end routing, the Peer–Peer Shortcut route and Peer–SuperPeer 1-Hop route are utilized to complete the routing process. That is, for end-to-end routing between Peers, if there exists a Peer–Peer Shortcut route, then packet is delivered using this route. Otherwise, the Peer–SuperPeer 1-Hop route is used to route the packet from the source Peer to the source SuperPeer at the SuperPeers layer serving the source Peer and localized random compass routing protocol is activated to deliver the packet to the destination SuperPeer that serve the destination Peer which the Peer–SuperPeer 1-Hop route is used to complete the routing process. The motivation for using localized random compass routing algorithm at SuperPeer layer is that the results in [22] show that the delivery rate of random compass routing in *Yao-graph* is 100%.

Here we describe the localized random compass routing protocol at the SuperPeers layer.

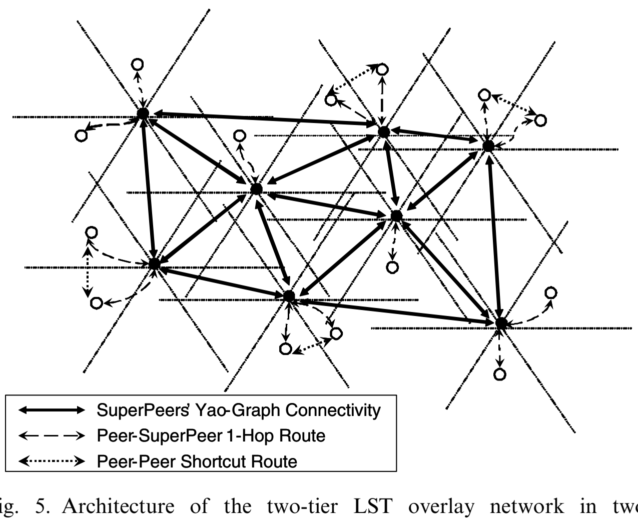

SuperPeers' Yao-Graph Connectivity
Peer-SuperPeer 1-Hop Route
Peer-Peer Shortcut Route

Fig. 5. Architecture of the two-tier LST overlay network in two-dimensional view.

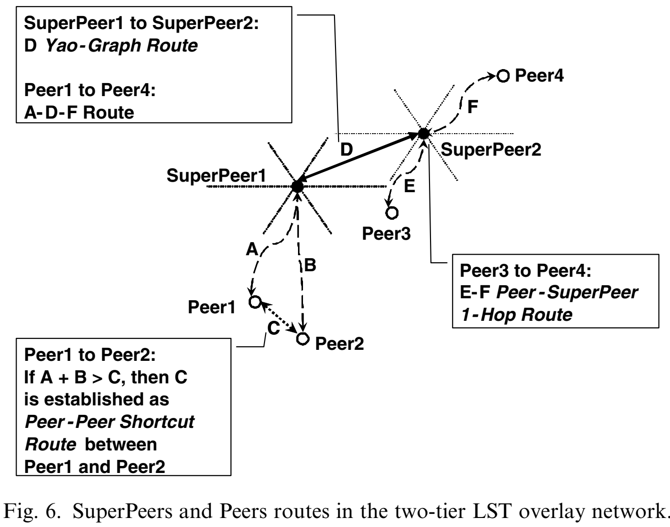

SuperPeer1 to SuperPeer2:
D Yao-Graph Route
Peer1 to Peer4:
A-D-F Route
Peer4
F
D
SuperPeer2
SuperPeer1
E
Peer3
A
B
Peer3 to Peer4:
E-F Peer-SuperPeer 1-Hop Route
Peer1
C
Peer2
Peer1 to Peer2:
If A + B > C, then C is established as Peer-Peer Shortcut Route between Peer1 and Peer2

Fig. 6. SuperPeers and Peers routes in the two-tier LST overlay network.

Let node s be the source node, u be the current node and t be the destination node. In *localized* routing, the next node is selected to forward the packets based on the information in the header of the packet and the position of its local neighbors. The localized routing algorithm routes a packet from a source node s to a destination node t whereby each relaying node decides to which node to forward the packet only based on the following information: the source node s, the destination node t, the current node u and all the k-local nodes of node u. We use the case for $k = 1$. That is, a node uses only the information of the 1-Hop neighbors.

Suppose a packet is currently at node u and the destination node is t. The basic localized compass routing uses the local information of node u to route packets. That is, it finds the next node v of node u. Basically, compass routing finds the next relay node v such that the $\angle vut$ is the smallest among all neighbors of node u in the *Yao-graph* topology. For localized random compass routing: Let v_1 be the node on the above of line ut such that $\angle v_1ut$ is the smallest among all such neighbors of node u. Also, we define node v_2 to be the node below line ut that minimizes the angle $\angle v_2ut$. Then node u randomly choose either node v_1 or node v_2 to forward the packet, as illustrated in Fig. 7.

2.5. Geometric overlay maintenance

Our two-tier LST overlay network uses stable overlay nodes as SuperPeers to handle most of the heavy system workloads and reduce the network maintenance overhead. When an overlay node leaves the system or a failure occurs, the information that is related to the leaving overlay node must be updated among the other affected overlay nodes. Similarly, if a new overlay node joins the system, information relating to the new overlay node will also have to be updated. For high node churn, network maintenance overhead can be heavy. The following cases describe the LST overlay network maintenance operations during node churn which will invoke the *local* topology repair algorithm.

2.5.1. New overlay nodes joining

A new overlay node will contact the *bootstrap service* operating at the SuperPeers layer for a standard overlay **JOIN** procedure. Once the new overlay node is elected as a SuperPeer or normal Peer, the following operations are executed.

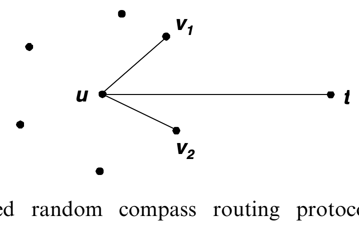


Fig. 7. Localized random compass routing protocol in *Yao-graph* topology at SuperPeers layer of LST geometric overlay network.

2.5.1.1. A new normal Peer is joining the LST overlay network. During the **JOIN** procedure, the new normal Peer measures the RTTs to all existing SuperPeers. It uses this information to join the cluster whereby the closest SuperPeer belongs. Then, the new normal Peer's geometric coordinates within the cluster are computed by the *Highways* overlay control plane service. Using the estimated geometric distances, the Peer–SuperPeer 1-Hop route to the closest SuperPeer is established and all possible Peer–Peer Shortcut routes are setup and updated with the neighboring Peers within the cluster.

2.5.1.2. A new SuperPeer is joining the LST overlay network. A new overlay node is elected as the new SuperPeer L and measures the RTTs to all existing SuperPeers. It uses this information to join the cluster whereby the closest SuperPeers belongs. Then, the new SuperPeer's geometric coordinates within the cluster are computed by the *Highways* overlay control plane service.

This new SuperPeer L starts to cut the space surrounding itself into *six* sectors with equal angle of $\theta = \frac{\pi}{3}$. Then this new SuperPeer builds the *Yao-graph* connectivity by connecting to other *six* closest SuperPeers in terms of the shortest geometric distance to other SuperPeers in its *six* sectors. It attempts to connect to the list of *six closest* neighboring SuperPeers in each of its *six* sectors. That is, the local topology repair algorithm will be invoked to reconstruct the SuperPeers' *Yao-graph* to include this new SuperPeer in the SuperPeers layer. The existing Peers that are associated with the neighboring SuperPeers also reorganize and reestablish the Peer–SuperPeer 1-Hop routes to this new SuperPeer, if there exists shortest 1-Hop routes.

The new SuperPeer L initializes its state by routing **Join** messages to the list of closest neighboring SuperPeers found in each of its *six* sectors. Once **Join** messages are routed to this list of *six* closest neighboring SuperPeers, the new SuperPeer L will establish *Yao-graph* overlay connectivity in the *six* sectors. The new SuperPeer L learns of the IP addresses of these closest neighboring SuperPeers in the *six* sectors. The neighboring SuperPeers also require to update their neighbor tables to eliminate those SuperPeers that are no longer neighbors as a result of this new inclusion of SuperPeer L. In addition, the existing Peers associated with the *six* closest neighboring SuperPeers reorganize and reestablish the Peer–SuperPeer 1-Hop routes to this new SuperPeer if there exists shortest Peer–SuperPeer 1-Hop routes. The connection relationships of the affected Peers that change their Peer–SuperPeer 1-Hop routes to this new SuperPeer L will be updated.

This update is done by broadcasting an **Update** message containing the new topological information to all affected SuperPeers and Peers, as illustrated in Fig. 8.

Once the *Yao-graph* overlay connectivity of this new SuperPeer L has been established, it will have a set of maximum *six* neighboring SuperPeers $N = \{N_1, N_2, \ldots, N_6\}$ in

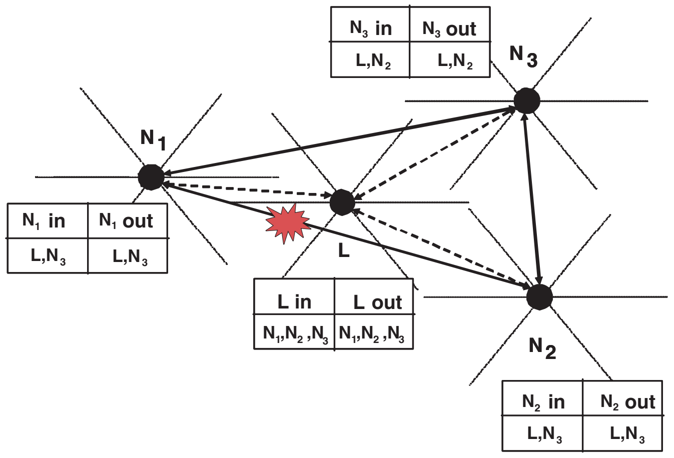


Fig. 8. Updates of neighbor tables in the *Yao-graph* at SuperPeers layer when a new SuperPeer joins.

the SuperPeers layer. In this example, before SuperPeer L joins the SuperPeers layer, SuperPeers N_1, N_2 and N_3 are connected directly to each other with *Yao-graph* overlay routes, as shown in dashed lines. After the SuperPeer L joins, the new SuperPeer L will connect to the neighboring SuperPeers N_1, N_2 and N_3 with their *Yao-graph* overlay routes, as shown in dotted lines. The SuperPeers N_1, N_2 and N_3 have to adjust their *Yao-graph* overlay connectivity by updating their neighbor tables. As a result of this update, the direct *Yao-graph* overlay route between SuperPeers N_1 and N_2 does not exist and new *Yao-graph* overlay routes are established to the new SuperPeer L. The **Update** message is sent with a limited range of time-to-live (TTL). The expected number of hops in the TTLs, $E[Hops_N]$ is $\log_6 N$ (where N is the number of SuperPeers in the SuperPeer layer, and a SuperPeer has *six* neighboring SuperPeers because of *six* sectors division in the *Yao-graph*). In our case where $N = 10{,}000$, this gives about five TTLs. This **Update** procedure ensures that the affected SuperPeers will quickly learn about the change and perform necessary update on their own neighbor tables accordingly.

2.5.2. Existing overlay nodes leaving

To be able to detect overlay nodes leaving the *LST* overlay network or overlay node failures, a *heartbeat* approach is used. Every overlay node sends small *alive* messages to each other periodically and maintenance operations are invoked when heartbeats are lost. The following operations are executed.

2.5.2.1. A normal Peer is leaving the LST overlay network. The missing heartbeat will be detected from this normal Peer. The associated SuperPeer and Peers who have their Peer–SuperPeer 1-Hop and Peer–Peer Shortcut routes with this normal Peer will attempt to free their connection resources. Only the affected SuperPeer and Peers are reorganized locally and this minimizes global overhead management.

2.5.2.2. A SuperPeer is leaving the LST overlay network. The *six* neighboring SuperPeers that are connected to this leaving SuperPeer will notice this failure through the missing heartbeats and trigger the local topology repair algorithm. It will reconstruct the *six* neighboring SuperPeers' *Yao-graph* relationships and reorganize the Peer–SuperPeer 1-Hop and Peer–Peer Shortcut routes of its connecting Peers.

The neighboring SuperPeers of this leaving SuperPeer are notified of this change and update their neighbor tables. This notification occurred because SuperPeers periodically exchange *alive* heartbeat messages. When this leaving SuperPeer leaves the overlay network and heartbeats stop, every neighboring SuperPeers will send a **Discovery** broadcast message with a limited five TTLs to other neighboring SuperPeers. Each neighboring SuperPeer receiving the **Discovery** broadcast message will respond with its geometric position information and its IP addresses.

The **Join** and **Update** procedures as described above, will help to adjust its current topology state for the affected SuperPeers and Peers as illustrated in Fig. 9.

This ensures that the SuperPeers' *Yao-graph* connectivity and Peers' Peer–SuperPeer 1-Hop routes can be quickly reconstructed locally as a result of this change. Due to the lightweight properties of *Yao-graph*, node churn causes little problem to the hierarchical layers of the LST overlay network, as long as a SuperPeer does not become disconnected by the loss of all its neighboring SuperPeers. Even for the drastic case of losing all neighboring SuperPeers, the affected SuperPeer can contact the bootstrap service to rejoin the network.

3. Network-aware geometric overlay multicasting

In *Bos*, we construct source rooted multicast tree whereby each sender (source or root) builds a separate tree

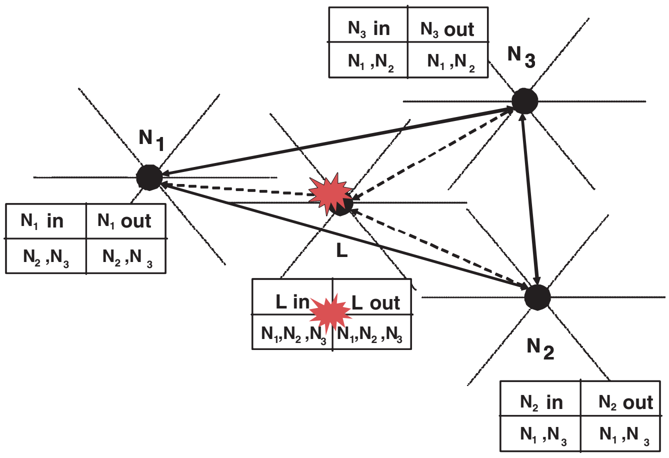


Fig. 9. Updates of neighbor tables in the *Yao-graph* at SuperPeers layer when a SuperPeer leaves.

rooted to itself at the LST's SuperPeers layer. We adapt the reverse path forwarding (RPF) algorithm [23] to construct *optimal reverse path tree* from the source to all receivers with optimal geometric routing routes. Our adapted version of RPF algorithm relies on the SuperPeers' knowledge of the optimal routing routes from it to the source. So we use the LST's localized random compass routing algorithm based on the shortest geometric distances between the SuperPeers to find this optimal geometric routing paths at the SuperPeers layer. These resulting geometric routing paths are in general different from the shortest path tree from the source to all the other receivers. In our case, since the geometric overlay routes between a root and another SuperPeer group member is symmetrical, the optimal-path tree and the optimal reverse path tree from the source are isomorphic.

Each SuperPeer group member has a simple forwarding behavior as depicted in Fig. 10. The adapted RPF algorithm allows a SuperPeer group member to accept a multicast data only on the connection interface from which the SuperPeer group member would send a unicast data to the root. The union of these source-receiver geometric routes forms the optimal reverse path tree for our overlay multicasting. From the root to all other SuperPeer group members, we build a multicast distribution tree rooted at the SuperPeer root (source) such that each SuperPeer group member has the optimal geometric overlay route back to the SuperPeer root. The result of this *grafting* process creates the multicast distribution tree at the SuperPeers layer. Since the *Yao-graph* topology is constructed at the LST's SuperPeers layer which exhibits EMST characteristic, the multicast tree being formed also follows similar property. The forwarding tree may contain subtrees with no multicast group members and data need not be forwarded down the subtree. That is, if a neighboring SuperPeer does not belong to the multicast group and there is no other group members downstream, it sends a *pruning* message upstream. Then the subsequent multicast data would not be forwarded to these group members who respond with the *pruning* messages.

The SuperPeers keep track of the multicast tree state using their *Children Tables*. These tables are used in our *localized* multicast group membership management protocol (as illustrated in Fig. 11) to reduce the maintenance overheads and the impact of reconstructing the multicast tree due to node churn. When node churn occurred, we rely on the local topology maintenance control in the LST overlay network to repair the overlay. We also invoke a **local ancestor-recovery protocol** to localize repair and self-healing of the multicast tree. In this manner, group membership operations are decentralized and managed efficiently. We describe our localized multicast group membership management as follows.

Scalable multicasting with network-aware geometric overlay

```
MulticastRPF($GroupMember, msg$)
1  if ($msg$ received on $GroupMember.IncomingLink$
2      on optimal routing path back to $Root$)
3     then Forward($msg, GroupMember.AllOutgoingLinks$)
4     else  return Ignore($msg$)
```

Fig. 10. Adapted reverse path forwarding algorithm.

```
GroupMembershipManagement($GroupMember, ActionType, groupID$)
1  switch
2    case $ActionType = CONNECT$ :
3         $NewMember \leftarrow GroupMember$
4         $NewMember.Parent \leftarrow$ FindClosestSuperPeer($NewMember$)
5         AddChildrenTable
6         ($NewMember.Parent.ChildrenTable, NewMember$)
7         UpdateAncestors($NewMember, groupID$)
8
9    case $ActionType = DISCONNECT$ :
10        $LeavingMember \leftarrow GroupMember$
11        RemoveChildrenTable
12        ($LeavingMemeber.Parent.ChildrenTable, LeavingMember$)
13        while ($LeavingMember.ChildrenTable$)
14        do
15            for $Child \leftarrow 1$ to $N \in LeavingMember.ChildrenTable$
16            do
17               $Child.Parent \leftarrow$ FindClosestParent
18               ($Child.ClosestAncestor, Child.ClosestAncestor.ChildrenTable$)
19               AddChildrenTable
20               ($Child.Parent.ChildrenTable, Child$)
21               UpdateAncestors($Child, groupID$)
22
23   case default  :
24        if ($GroupMember.Heartbeat > DefaultTimeout$)
25          then GroupMembershipManagement
26               ($GroupMember, DISCONNECT, groupID$)
```

Fig. 11. Algorithm for *Bos*'s multicast group membership management.

1. To join the multicast tree (lines 2–7), a new joining SuperPeer group member sends a **Connect** message to the multicast group with the groupID. This new joining SuperPeer group member will select the closest SuperPeer group member in the multicast tree to join. The closest SuperPeer group member will become the new parent and its *Children Table* is updated with this new SuperPeer group member's IP address which will be used for multicast routing. The new joining SuperPeer group member will obtain and maintain a **list of ancestors** (with their IP addresses), from the root to its parent. In this way, every group member will maintain this list of ancestors from the root to its parent and this helps to reduce service interruption probability. The geometric locality properties of the LST overlay network enable the joining group member to select a parent with smaller geometric distance cost. This ensures that the data reach all group members within relatively few optimal geometric hops and with reasonably small delay latency.
2. To leave a multicast group (lines 9–21), this leaving SuperPeer group member in the multicast tree sends a **Disconnect** message to its parent in the multicast tree. All the related multicast group membership information of this leaving SuperPeer group member will be removed from its parent's *Children Table*. This leaving SuperPeer group member may have children in the multicast tree. These *affected* children will respond to this change by making a check on its list of ancestors. They will find their closest neighboring ancestors as obtained from their lists of ancestors to rejoin the multicast tree. These closest neighboring ancestors will provide the information of their children in their *Children Table*. With the geometric position information of the ancestors and their children, the closest ones will be chosen as the new parents to serve those affected children. These newly identified parents will update their *Children Table* with these affected children in the multicast group and these affected children will update their lists of ancestors accordingly. Using this **local ancestor-recovery protocol**, the affected children will be able to find their respective new parents quickly and save recovery cost in a highly dynamic system with frequent node churn.
3. To detect node churn of the SuperPeer group member (lines 23–26), all parents in the multicast tree periodically send heartbeat's *alive* messages to their children. If a child does not respond to the heartbeat for a period of time (timeout), the parent assumes that the child has failed. Then the parent will invoke local repair of this failed child in the multicast tree with the **Disconnect** operation, and subsequently, the local ancestor-recovery protocol is executed.

The list of SuperPeer group members is periodically updated. When the root detects a change occurred, update is distributed across the SuperPeer group members along the tree. When fault detection (node churn) and recovery occurred, only a few **Connect** and **Disconnect** messages ever reach all the way to the root. However, we keep track of the *total* number of **Connect** and **Disconnect** messages. If this number exceeded a certain *threshold*, the multicast tree rooted at the source is reconstructed using our adapted RPF algorithm. This is because after some failure recoveries, the multicast tree may not be optimal and some nodes might stuck in unfair and unfavorable situations. We set the threshold $= \frac{N}{\log_{10} N}$ for N SuperPeers. Our intuition for setting this threshold is based on the fact that we want to ensure that the total maintenance overhead messages of **Connect** and **Disconnect** do not exceed the order of O(log$N$). So, we divide the number of N SuperPeers N with $\log_{10} N$. In our case, we have threshold $= \frac{10{,}000}{\log_{10} 10{,}000} = 2500$. This would give us an estimate of the amount of maintenance overheads incurred in the multicast tree due to node churn. This justifies that reconstruction of the multicast tree is needed. That is, whenever this *threshold* is reached, the multicast tree rooted at the source is reconstructed using our adapted RPF algorithm. New multicast tree can be constructed concurrently while existing tree are in use. When the reconstruction is completed, the new multicast tree replaces the existing tree for the multicast data delivery.

Bos builds a multicast tree per application group. Multiple multicast trees can be formed at the SuperPeers layer of the LST overlay network. The multicast tree constructed at the SuperPeers layer provides the reliable low-latency and high-bandwidth backbone multicast connectivity for the lower layer's Peers who basically connect to their SuperPeers using their direct Peer–SuperPeer 1-Hop routes during multicast streaming. The Peers in the Peers layer who have their own direct Peer–SuperPeer 1-Hop connectivities with their SuperPeers can join and become the group members of the multicast tree by sending unicast messages to their SuperPeers which will serve and provide overlay multicast service for them. The SuperPeers serving the Peers will be responsible to become the group members of the multicast tree at the SuperPeers layer. All multicast communications

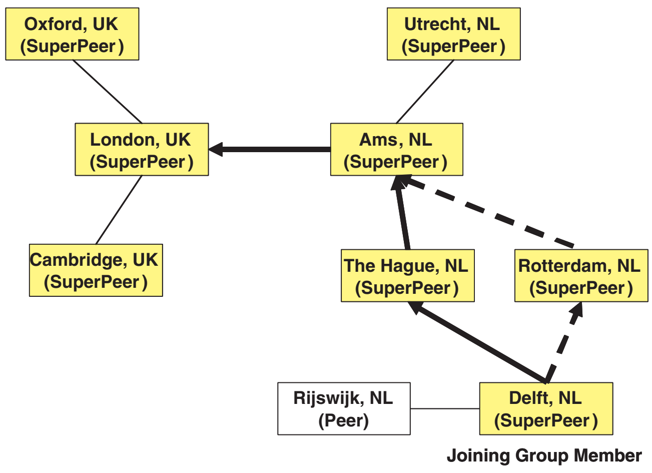


Fig. 12. A multicast group member is joining a multicast tree in *Bos*.

are done using standard transport control protocol (TCP) for the transport reliability.

Fig. 12 shows an example of the multicast group membership management operation by multicast group members (SuperPeers at SuperPeers layer) in The Netherlands (NL) and United Kingdom (UK). For simplicity, we make two assumptions: first, the RTTs are proportional to the geometric distances due to the network embedding on the RTTs between group members. The international links between group members located in Amsterdam and London have a high bandwidth. Second, the group member located in London acts as the root of the multicast tree and group members at The Hague and Rotterdam lie in the same Internet service provider (ISP) network and are elected as SuperPeers. We have a normal Peer, Rijswijk, with Peer–SuperPeer 1-Hop connectivity to the SuperPeer, Delft. Delft is joining the multicast group and it sends a **Connect** message to the multicast group. *Bos* selects the closest parent in the multicast tree and forwards the message to that parent (The Hague). The information of Delft (such as IP address and port number) is stored in its parent, The Hague's *Children Table*. And, the Delft's **ancestor list** is updated with the root to its parent, i.e. London (root) and Amsterdam (ancestor). The multicast links are shown by the **solid arrows**. In this way, the new group member, Delft, has joined the *Bos* multicast tree with London as its root. And, its normal Peer, Rijswijk, who requires multicast service is served by its SuperPeer, Delft, in the multicast group.

If Delft detects that its parent, The Hague, has failed due to missing heartbeats, then Delft initiates the **local ancestor-recovery protocol**. Delft finds its closest neighboring ancestor (Amsterdam) as obtained from its list of ancestor to rejoin the multicast tree. Amsterdam will provide the information of its children in its *Children Table* to Delft. With the geometric position information of Amsterdam and its child (Rotterdam), the closest Rotterdam is chosen as the new closest parent. The new closest parent, Rotterdam, will update its *Children Table* of this new SuperPeer child, Delft, in the multicast group. The new multicast links are shown by the *dashed* arrows. We have illustrated in this example the efficiency and scalability of the joining multicast group member to find its parent with rich multicast resources within the same country with close locality. Similarly, if Delft is leaving the multicast group, it sends a **Disconnect** message to its parent in the multicast tree, and its information will be removed from its parents' *Children Table*.

4. Implementation and experiments

This section describes the experimental setup and implementation required to evaluate the performance of LST's and *Bos*. We want to assess their scalability and effectiveness in both massive scale networks and real world Internet-scale environment. We use the same massive scale networks that were used by Scribe [12,13] and SplitStream [14] from the Microsoft Research Cambridge. The first part of the experiment aims to evaluate the network-aware geometric overlay network (LST) performance in massive scale networks. The second part evaluates the performance of *Bos* overlay multicasting in massive scale networks. Since we are using the same massive scale networks, we discuss the impact of our performance results in *Bos* with the results published by Scribe and SplitStream. Then, we implement the multicast using IP unicast scheme, IP multicast and multicast in Gnutella-like random graph, to make performance evaluation and comparison with *Bos* overlay multicasting. Finally, the third part of the experiment deploys the LST overlay network and *Bos* overlay multicast streaming network in PlanetLab [11]. We use the same experimental performance metrics we have used in the massive scale networks to evaluate the performance of LST overlay network and *Bos* overlay multicast streaming network in the PlanetLab's real world Internet-scale environment.

4.1. Simulation experiments in massive scale networks

The first and second parts of the simulation experiments are implemented using the massive scale networks that were used by the Scribe [12,13] and SplitStream [14] from Microsoft Research Cambridge. The massive scale networks are generated by Georgia Tech [24] random graph generator. The hierarchical transit-stub model contains 5050 routers. There are 10 transit domains at the top level with an average of five routers in each. Each transit router has an average of 10 stub domains attached, and each stub has an average of 10 routers. There are 100,000 end-system nodes that were randomly assigned to routers in the core with uniform probability. Each end-system node is directly attached by a local area network (LAN) link to its assigned router. There are 10 different networks using the same parameters but different random seeds – we have 10 massive scale networks with network model named as 0–9.

For such a massive scale network, it is more feasible to develop a simulator for our experiments. The well-known network simulator such as *ns*-2, would not be able to handle this large size of the networks involved and the dynamics of the overlay networks. The simulator models the propagation delay on the physical links as follows. The delay of each LAN link is set to 1 ms and the average delay of core links generated by the graph generator is 40.7 ms. Similar to the work of Scribe, the simulator does not model queuing delay, packet losses, or any cross network traffic because modeling of such parameters would prevent the simulation of massive networks.

To examine whether LST overlay network and *Bos* overlay multicasting are efficient overlays supporting multiple concurrent applications with varying requirements, we use the same environment as Scribe and SplitStream to run experiments using a large number of groups with a wide range of group sizes. As in Scribe [13], since there are no sources of real-world trace data to drive the

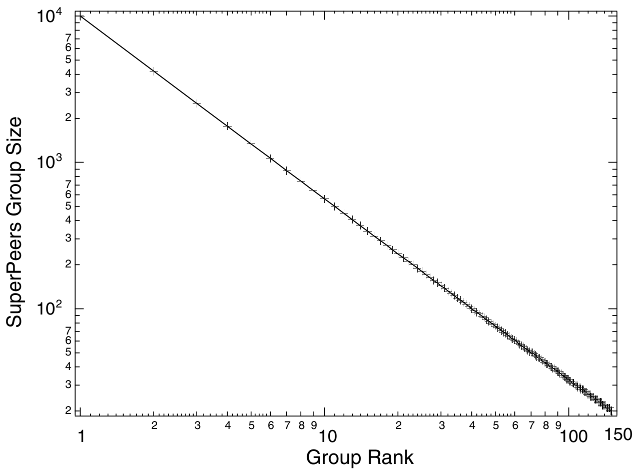


Fig. 13. Group size of SuperPeers versus group rank in massive scale networks.

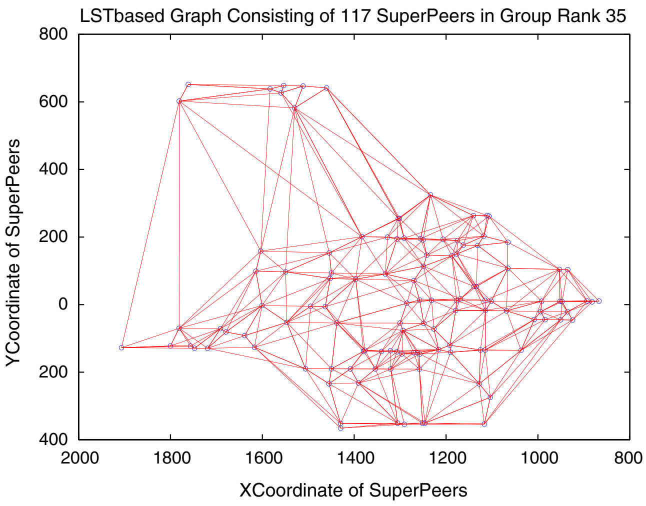


Fig. 14. *Yao-graph* structure of 117 SuperPeers in group rank of 35.

experiments, a Zipf-like distribution for the group sizes is adopted. Groups are ranked by its sizes. The size of a group with rank r is given by $gsize(r) = \lfloor Nr^{-1.25} + 0.5 \rfloor$, where N is the total number of overlay nodes. In each network model, the total number of group ranks is fixed at 150 (i.e. the total number of groups is 150 with group rank 1–150) and the number of overlay nodes (N) is fixed at 100,000, which are the numbers being simulated. Fig. 13 shows the group size of SuperPeers versus group rank in massive scale networks.

In each group, we choose 10% of the total number of overlay nodes to be the SuperPeers based on the election criteria described in subsection 2.1. The reason for the choice is derived from the recent study [25] which states that there are approximately 10% of the overlay nodes that have high capacity, and they exhibit stability and reliable connectivity in the overlay network. In each network model, the maximum number of SuperPeers is 10,000 in group rank 1 which consists of 100,000 nodes. We run our simulation system on these 10 massive scale networks and the total number of groups is 1500. We generate two-dimensional geometric coordinates for all the nodes in the system. Since our performance results in all 10 networks are similar, only the *average values* over the 10 massive scale networks are shown. To visualize the SuperPeers' *Yao-graph* structure being constructed in the overlay network, Fig. 14 shows an example of the two-dimensional geometric *Yao-graph* for group rank 35 containing a total of 117 SuperPeers.

For each group, we also implement the IP multicast, multicast using IP unicast scheme, and multicast in Gnutella-like random graph. This allows us to make performance comparison of *Bos* and these overlay multicast networks. Similar to Scribe [13], we use the policy routing link weights generated by Georgia Tech random graph generator to perform IP unicast routing. That is, all links of the same type such as intra-domain or inter-domain links, are assigned the same link weight. IP multicast routing uses a shortest path tree formed by merging the unicast routes from the source to each receiver, similar to what could be obtained using protocols like Distance Vector Multicast Routing Protocol (DVMRP) [26] or Protocol Independent Multicast (PIM) [27]. We ignore messages needed by these protocols to maintain the tress, so as to give a conservative comparison. To implement multicast in Gnutella-like random graph, each node in that overlay multicast network maintains a neighbor table with network addresses of its neighbors. Our design for multicast in Gnutella-like random graph bounds the number of neighbors to be at least 4 and not more than 12 of neighbors. We use a generic flooding mechanism to discover and respond to multicast queries in the random graph.

4.2. PlanetLab experiments

Our objective of this PlanetLab experimental setup is to evaluate the performance of LST overlay network and *Bos* overlay multicast streaming network with differing groups of elected SuperPeers in real world Internet-scale environment. We implement and deploy *Bos* in the PlanetLab [11], and run overlay multicasting experiments to evaluate the effectiveness and performance of the *Bos* overlay multicasting. In this experiment, we select one node in each site as a representation of the site because there may be many nodes (at least more than 5) that are very close by residing in each of these sites in the PlanetLab testbed. By using PlanetLab sites instead of nodes, this would enable a closer replica of Internet-scale environment – we have a total of 230 sites. We choose differing groups of PlanetLab sites (in groups of 20, 40, 60 and 81 sites) that possess the criteria of SuperPeers to become the elected SuperPeers – the distribution of the elected SuperPeers ranges from 9% to 35% of the PlanetLab sites. The rest of the PlanetLab sites are Peers in the Peers layer. We use `planetlab2.ewi.tudelft.nl` as the root (source) of the *Bos* multicast tree, generating a 560 Kbits/s (70 Kbytes/s)

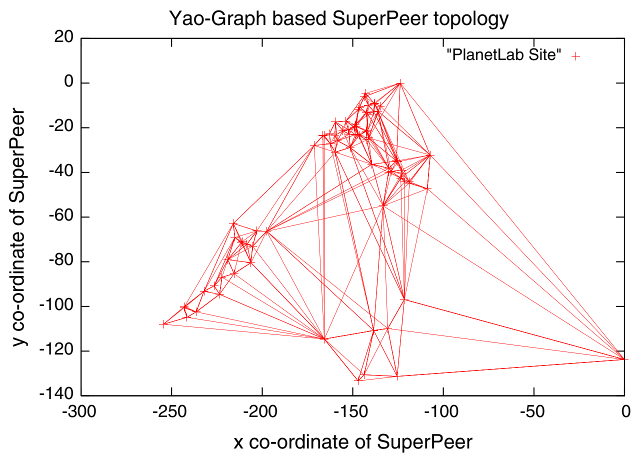

Yao-Graph based SuperPeer topology
"PlanetLab Site"
y co-ordinate of SuperPeer
x co-ordinate of SuperPeer

Fig. 15. *Yao-graph* structure of 81 PlanetLab sites (SuperPeers).

multicast stream in the multicast tree. The Round-Trip-Time (RTT) performance metrics are measured using Internet control management protocol (ICMP) ‘ping’ with 15-ping measurements between every two sites. We are interested in the network distances between nodes and ignore as much as possible any network-related conditions. Therefore, we take the minimum of the site-to-site RTT metric to avoid biases in the experimental results due to high variations in RTT measurements. We map the PlanetLab sites into two-dimensional Euclidean space for our Internet subspace geometry. Fig. 15 visualize the two-dimensional geometric *Yao-graph* structure for a group of 81 PlanetLab sites.

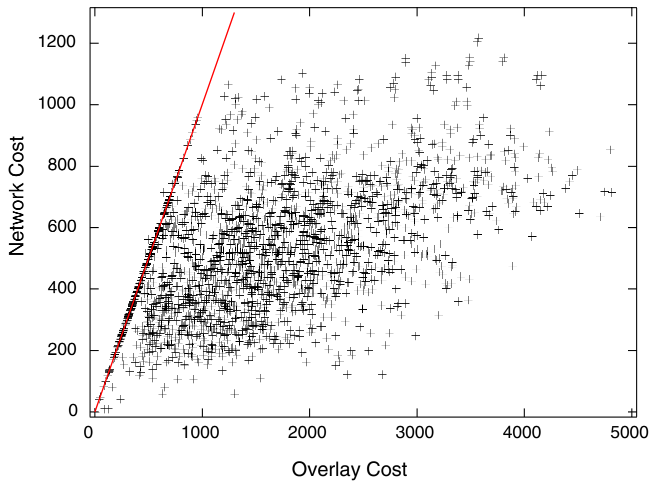
5. Performance evaluation and results

5.1. Geometric overlay (LST) performance in massive scale networks

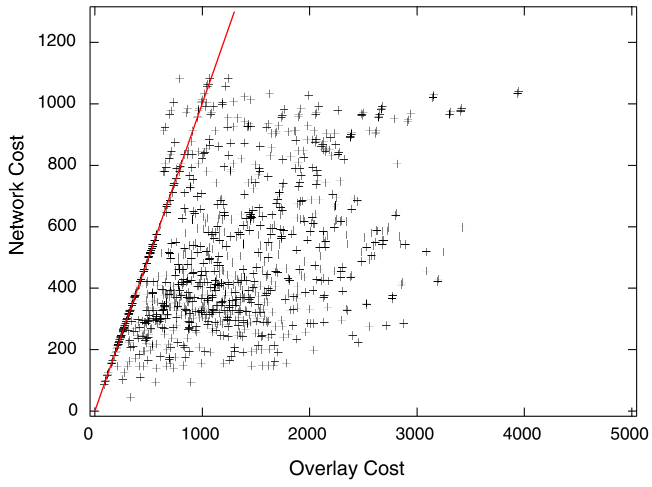
The first simulation experiment evaluates the LST overlay network performance in massive scale networks.

5.1.1. Overlay cost vs underlay cost

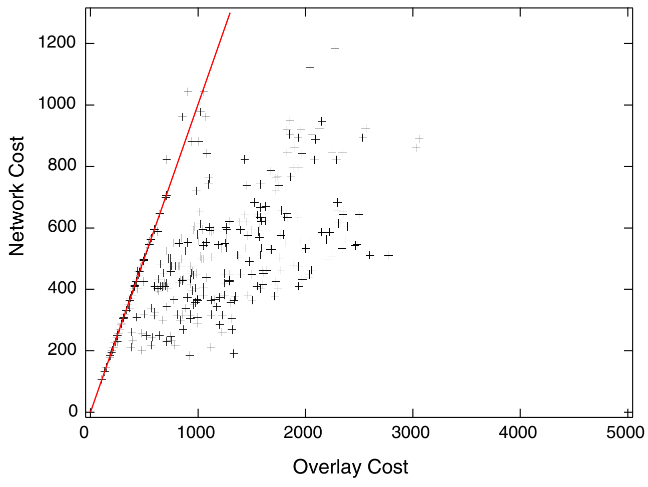
For each of the source-destination SuperPeers pair, we compare the LST overlay network cost and underlay network cost on a path-by-path basis. That is, we measure and compare the network cost between two nodes for direct IP communication utilizing the underlying network and the cost of using the overlay network. Fig. 16 shows the scatterplot performance of geometric SuperPeers-to-Peers overlay hierarchy for group rank 72, 96 and 148, respectively. The *X*-axis is the LST overlay network cost and the *Y*-axis is the underlay network cost. The solid linear line gives the indication of network cost being equal to the overlay cost and its purpose is to show this boundary. The results show that for some cases, using the LST overlay network for communications outperforms the direct IP-based communications in the underlying network. It also shows that the LST overlay network communications’ latencies are *reasonable* in delivering messages relative to



(a) Group Rank 72



(b) Group Rank 96



(c) Group Rank 148

Fig. 16. LST overlay network cost versus underlay network cost for group rank 72, 96 and 148 of the network model 0.

their direct underlay communications. These results are expected because all overlay communications usually suffer a slightly higher communication cost than the direct

Internet communications. This is due to the overlay network routing that is usually not as optimal as direct communications in the underlying network – these results are consistent with performance seen in Resilient Overlay Network (RON) [28].

5.1.2. In-degree/out-degree of a SuperPeer

The in-degree/out-degree of a SuperPeer denotes the number of In/Out neighbors that are maintained by that SuperPeer. For a SuperPeer to limit its outbound network bandwidth, it must limit its out-degree in the overlay network, otherwise, its forwarding capacity can be exceeded. The out-degree of a node in our *Yao-graph* is bounded at most s and its in-degree may be as high as $N-1$ where s is the number of sectors and N is the total number of nodes. In our LST overlay network, the out-degree of the SuperPeer is bounded at most 6, which is reasonably small. A SuperPeer with a high in-degree may easily become exhausted. It is interesting to find out the in-degree of the *Yao-graph* in our LST overlay network using the massive scale networks.

For each network model and group, we compute the in-degree/out-degree of each SuperPeer. The distributions of mean and maximum node degree for different group ranks are shown in Fig. 17. As expected, the maximum out-degree of a SuperPeer in our *Yao-graph* is 6. The experimental results show that the average in-degree is the **same** as that of average out-degree, i.e. the average in-degree is equal to the average out-degree in our experiments. It is evident that the average in-degree of each group is relatively small, with a maximum of 6. A small average in-degree in our LST overlay network suggests a low *link stress* (as defined in [29]) in the massive scale networks. The maximal in-degree of group rank 1 is 150, which is still realistic. The maximum out-degree is 6 due to the bounding characteristics of our *Yao-graph* structure in two-dimensional geometric space which has *six* sectors connecting *six* closest neighbors.

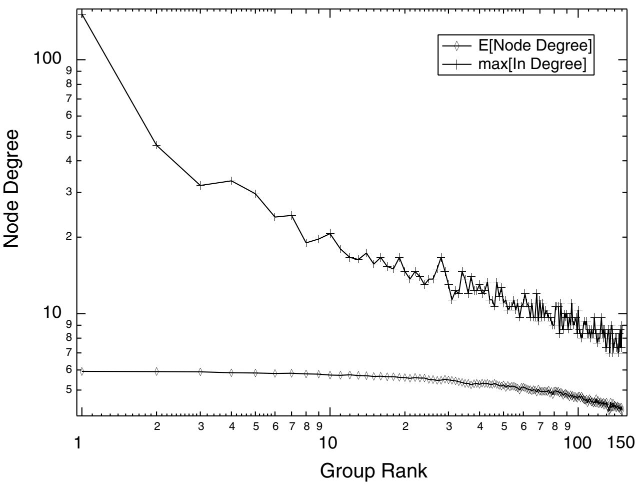


Fig. 17. Distribution of mean node degree and maximum in-degree in the LST SuperPeers overlay.

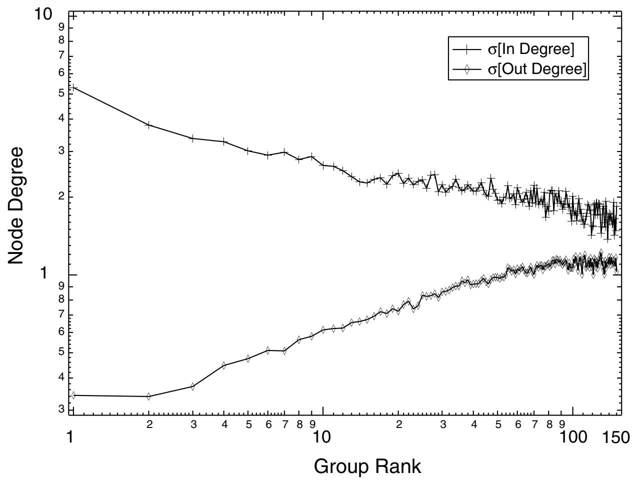


Fig. 18. Distribution of standard deviation of in-degree and out-degree in the LST SuperPeers overlay.

Fig. 18 illustrates the standard deviation of in-degree and out-degree for different group ranks. It is evident that the standard deviation of the SuperPeer's out-degree decreases as group rank decreases for 150–1. This indicates that the standard deviation of out-degree decreases as the group size increases. The standard deviation of out-degree is small: the minimum standard deviation of out-degree is 0.34 in group rank 2 and maximum standard deviation is 1.22 in group rank 130, giving an average of 0.98 over all groups. Again, this result is expected. For each group, all SuperPeers in our *Yao-graph* have their out-degree bounded at most 6.

However, the standard deviation of SuperPeer's in-degree increases with the group size. The minimum standard deviation of in-degree is 1.37 in group rank 138 and maximum standard deviation of in-degree is 5.3 in group rank 1, giving an average of 2.03 over all groups. This shows that the in-degree of the SuperPeer can be high. This is due to the possibility that this special neighbor is the only nearest neighbor to many other SuperPeers. To overcome the possibility of exhausting a SuperPeer's in-degree, undirected *sparsified Yao-graph* can be considered. Basically, a *sparsified Yao-graph* is a *Yao-graph* whereby only the shortest incoming edge is selected for incoming link if the in-degree of a sector exceeds one (see the Appendix).

5.2. Geometric overlay multicasting (*Bos*) performance in massive scale networks

In the previous section, we have shown the efficiency of the LST overlay network in massive scale networks. This second part of the experiment evaluates the performance of *Bos* overlay multicast streaming in massive scale networks. In each group, the elected SuperPeers send multicast messages to all other SuperPeers. We compare the results of *Bos* overlay multicasting with the results published in Scribe and SplitStream and discuss the implications. Then, we repeat the experiments with IP multicast,

multicast using IP unicast scheme, and multicast in Gnutella-like random graph. We then compare their multicast performance results with the overlay multicast performance of *Bos*.

5.2.1. Link stress

Link stress quantifies the load on the network which is equivalent to the number of packets sent over the link. Link stress [30] is measured as the number of duplicate packets carried by each network physical link during the overlay multicasting. A node with high link stress can be easily exhausted. Fig. 19 shows the histogram distribution of the mean link stress for the *Bos* overlay multicast network, when a message is multicast in each group of the total 1500 groups (10 networks, 150 groups in each network), i.e. it shows histogram distribution of the mean link stress for all the links in all groups.

Our results show that most links have low stress: the mean link stress is 4.4, with standard deviation 5.6. This means that the mean link stress induced by *Bos* is approximately 4.4 times higher than IP multicast on the same experiment. Our result is acceptable but *moderately* higher than the result in Scribe, which generates 3.4 times higher mean link stress than IP multicast. The mean link stress in SplitStream is 1.28 times higher than IP multicast – SplitStream has a lower link stress and this is because of multiple multicast trees created by SplitStream which distribute the multicast load. As shown in the tail of the plots in Fig. 19, only 1 link has a mean link stress of 100 and 52 links have mean link stress of 20. In total, 2% (523 links out of a total of 25,391 links) of the links have mean link stress more than or equal to 20. This is *relatively small* in number.

We examine the maximum link stress and notice that *Bos* has maximum link stress of 100 which is *much smaller* than the maximum link stress of 4031 published in Scribe. However, the Scribe results are generated for 100,000 nodes in the overlay network whereas our results are derived from 10,000 SuperPeers at the SuperPeers layer and the rest of 90,000 Peers are connected directly to the SuperPeers. In general, our comparisons show that *Bos* has a *lower* maximum link stress due to its network-aware SuperPeers–Peers geometric overlay hierarchy and it has acceptable mean link stress.

5.2.2. Node stress

Node stress quantifies the load on nodes which is equivalent to the number of messages that it receives. End nodes in the overlay multicast network are responsible for maintaining routing information and for forwarding and duplicating packets whereas routers perform these tasks in IP multicast. We measure the node stress by counting the number of nodes in each node's routing table and the number of messages received by each node when members join the groups. Fig. 20 shows the distribution of mean and standard deviation of node stress for each group in *Bos*. For all 1500 groups, the mean node stress are between 4.81 in group rank 134 to 7.75 in group rank 1, giving an average of 7.08 over all groups. This average result (7.08) is close to the average node stress published in Scribe which has an average node stress of 6.2. It is also lower than the configuration in SplitStream whereby the desired in-degree and node stress is typically set to 16. This suggests that *Bos* overlay multicast streaming is efficient in spreading data over all nodes. Our experimental results also show that the standard deviation of node stress are between 1.41 in group rank 109 to 2.76 in group rank 97, giving an average of 2.25 over all groups. Among all the 1500 groups, the maximum node stress is 110. These results suggest that in *Bos*, end nodes just need to forward multicast messages only to a **small** number of other nodes and this is helpful to achieve **scalability**.

Scribe and SplitStream make use of some mechanisms to limit a node's out-degree, thus they may add extra complexity. Scribe uses a built-in mechanism ('push-down'). When a node has reached its maximal out-degree and it receives a request for connectivity from a new prospective child, it provides that new prospective child with a list of

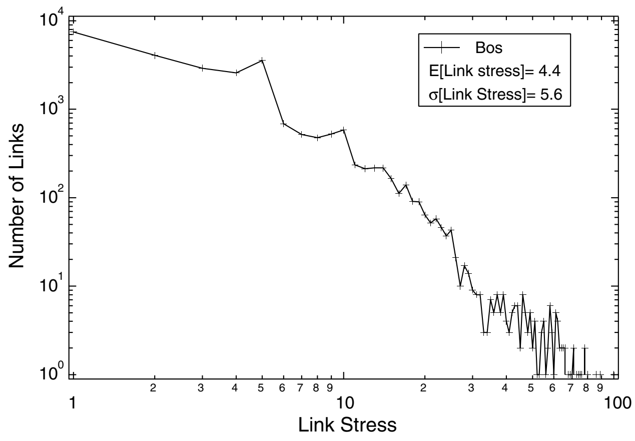


Fig. 19. Histogram distribution of mean link stress in *Bos* SuperPeers overlay multicasting.

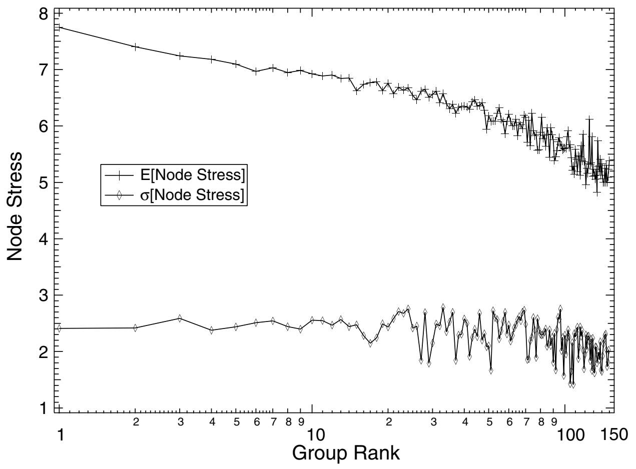


Fig. 20. Distribution of mean and standard deviation of node stress in *Bos* SuperPeers overlay multicasting.

its current children that are able to accept the request. This prospective child then seeks to be adopted by the child from the list with the lowest delay. This procedure continues recursively down the tree until a node is found that can take that prospective child. This is guaranteed to terminate successfully with a single Scribe tree provided each node is required to take on at least one child. However, this procedure is not guaranteed to work in SplitStream [14] which builds forest of multiple multicast trees. The reason is that a leaf node in one tree may be an interior node in another tree, and it may have already reached its out-degree limit with children in this other tree. SplitStream adds several procedures to limit its out-degree by introducing "Locating parents" and "Spare capacity group" schemes. In the "Location parents" scheme, the node first adopts the prospective child regardless of the out-degree limit. Then, it evaluates its new set of children to select a child to reject as an orphan. This selection is made in an attempt to maximize the efficiency of the SplitStream forest. If the orphan has not found a parent, it sends an anycast message to a special Scribe group ("Spare capacity group"). The node stress of each SplitStream node is published to be equal to its desired in-degree, and this enable nodes in SplitStream with sufficient bandwidth to participate in the system. Depending on the configuration of the SplitStream, the desired in-degree is typically set to 16 which means all nodes will have their in-degree and node stress of 16.

To evaluate the Bos' underlay connectivity, we examine the **underlay physical node degree** which measures the node's average number of underlay connectivity. The distribution of mean and standard deviation of underlay physical node degree for each group in *Bos* is shown in Fig. 21. It reveals that the mean underlay node degree is **relatively small**: between 2.8 in group rank 53–5.4 in group rank 144, giving an average of 3.17 over all groups. As shown, the standard deviation of the underlay node degree is between 2.05 in group rank 110–4.2 in group rank 2, giving an average of 2.74 over all groups. Over all the 1500 groups, the maximum underlay node degree is 40, minimum underlay node degree is 1, giving an average of 3.17. These results suggest that in the *Bos* overlay multicast service, end underlay nodes just need to forward multicast messages only to a **small** number of other nodes.

5.2.3. Relative Delay Penalty (RDP) and Relative Average Delay Penalty (RAD)

Overlay multicast adds some delays to messages and delay penalty helps to quantify the goodness of an overlay multicast. To evaluate this delay penalty in *Bos*, we measure the delays to deliver a message to each member of a group using both overlay and IP multicast. We evaluate the delay penalty with two metrics: Relative Delay Penalty (RDP) and Relative Average Delay Penalty (RAD). The average RDP computes the average of the ratio of overlay multicast network delays between any a pair of SuperPeer group members in *Bos* and their corresponding underlay network shortest path delays; while Relative Average Delay Penalty (RAD) computes a ratio between the average overlay delays in a group using *Bos* and average delays using IP multicast. The RAD measurements may show the relative paths of each packet when sent over a multicast tree.

Fig. 22 shows average RDP distribution versus group ranks. Our results show that the maximum average RDP is 3.8 in rank 1 and minimum average RDP is 1.64 in group rank 135, giving an overall mean of the average RDP is 2.08 over all groups. In addition, 50% of the groups have an average RDP of at most 2.18. For all group members in all the groups, we note that there are more than 57.05% of the group members have RDP less than 2.25, and more than 62.7% of the group members have RDP less than 4. These results are relatively close but slightly higher than the results published in Scribe: average RDP is 1.81 in group rank 1 consisting of 100,000 nodes and for all

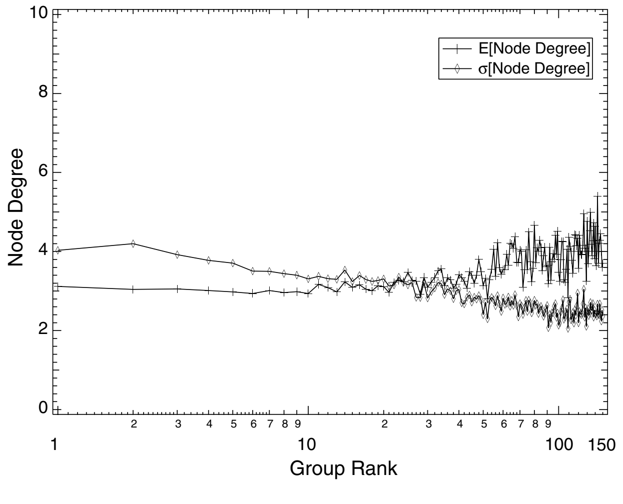


Fig. 21. Distribution of mean and standard deviation of underlay node degree in *Bos* SuperPeers overlay multicasting.

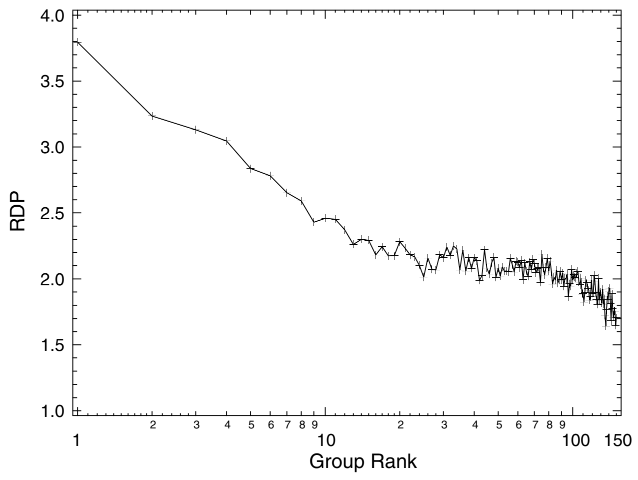


Fig. 22. Average Relative Delay Penalty (average RDP): average of the ratio of delays between any two SuperPeers in *Bos* SuperPeers overlay and underlay network.

groups, there are more than 80% of the group members have RDP less than 2.25, and more than 98% of the group members have RDP less than 4.

The results suggest that *Bos* overlay multicast streaming network delay is acceptable when compared with the corresponding underlay network delay. The average RDP decreases as the group size decreases. *Bos* has a *slightly higher* average RDP than Scribe because we use the localized random compass routing algorithm of LST for the calculation of the optimal geometric routing path (between a root and any other nodes) to construct the optimal-path overlay multicast tree. Although the use of this mechanism is efficient, fast and reliable in geometric overlay network [31], but with extra delays are incurred to guarantee packet delivery in *Yao-graph* topology of the multicast trees.

The RAD distribution versus group ranks is shown in Fig. 23. The maximum RAD (worst case) is 5.03 in group rank 1 and the minimum RAD is 2.03 in group rank 149, giving an average RAD of 2.8 over all group ranks. Significantly in *Bos*, smaller group size has better comparable performance and there are 50% of the groups having a RAD of at most 2.95. For Scribe, 50% of groups show good performance – RAD is at most 1.68 and in worst case, the maximum RAD is 2. Whereas for SplitStream, the average RAD is approximately 2 under some typical configurations (i.e. the desired in-degree for all nodes is 16, and the forwarding capacity of all nodes is 16). The RAD results of *Bos* are reasonable but the RAD results of Scribe and SplitStream are slightly better. This is because Scribe and SplitStream use Pastry's short routes property but there is tradeoff between the node degree and its delay penalty performance.

We proceed to evaluate and compare Bos's multicast performance with other multicast methods: multicast using IP unicast scheme, IP multicast, and multicast in Gnutella-like random graph using the same massive scale networks.

5.2.4. Transmission cost

We compare the total transmission cost (both delay and hop counts) using *Bos*, IP multicast, multicast using IP unicast scheme and multicast in Gnutella-like random graph. The total cost is calculated by summing the delay or hop cost of sending a packet from one group member to the rest of the group members. Fig. 24 shows the comparison of the total delay cost versus group ranks.

As expected, IP multicast is the best performer and multicast in Gnutella-like random graph is the worst. In statistical sense, the total delay cost in *Bos* is between minimum cost of 15,856.2 ms in group rank 148 to maximum cost of 460,487 ms in group rank 1, giving an average cost of 68,169.3 ms over all group ranks. For massive networks (group rank 1–20), the *Bos* overlay multicast streaming service outperforms the multicast using IP unicast scheme. However, *Bos* slacks in its total delay cost performance metric from group rank 20 onwards (i.e. network group size <223 nodes). As the network group size shrinks from 223 nodes, the multicast using IP unicast scheme performs slightly better. This is obviously clear that during overlay multicast, *Bos* is **more efficient** in massive network size than small network size. For smaller network group size, the multicast using IP unicast scheme is suitable because it does not have the complexities of multicast membership and routing. Significantly, in group rank 1, the total delay cost in *Bos* is 86% less than the multicast using IP unicast scheme, and approximately 30% higher than IP multicast scheme. This indicates strictly that the *Bos* overlay multicast network has a reasonably good control of the total delay cost in massive scale network size.

Fig. 25 shows the comparison results in total hop counts. The total hop counts in *Bos* is between minimum cost of 296.5 in group rank 148 to maximum cost of 8894.8 in group rank 1, giving an average cost of 1280.2 over all group ranks. The results show that IP multicast always outperforms except for group rank 1 whereby *Bos* performs better. That is, *Bos* is 38.8% **much cheaper** in

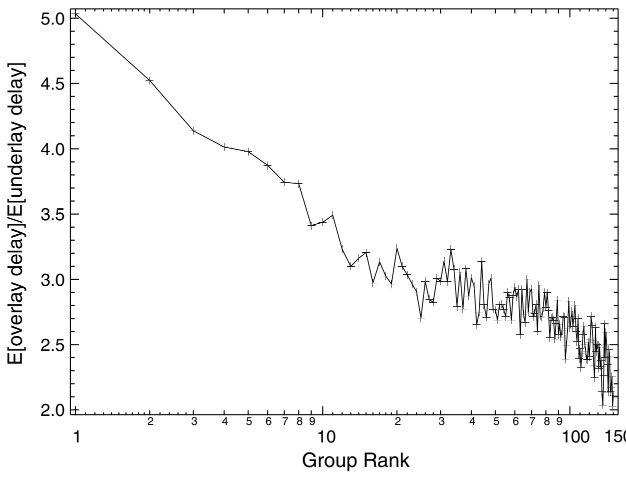


Fig. 23. Relative Average Delay Penalty (RAD): Ratio of average delays between any two SuperPeers in *Bos* SuperPeers overlay and average delays using IP Multicast in underlay network.

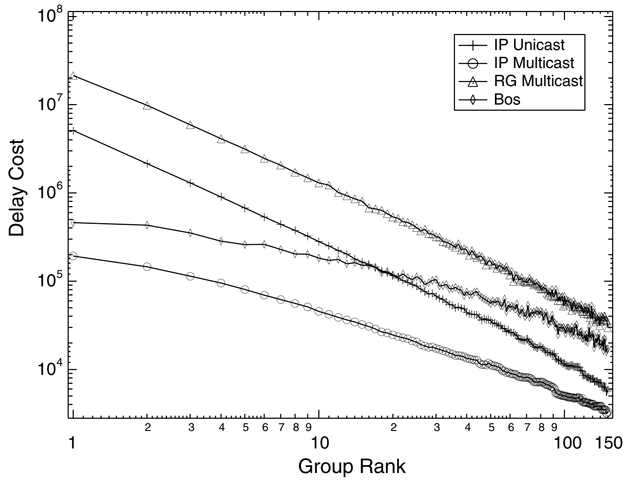


Fig. 24. Delay cost in *Bos* SuperPeers overlay multicasting.

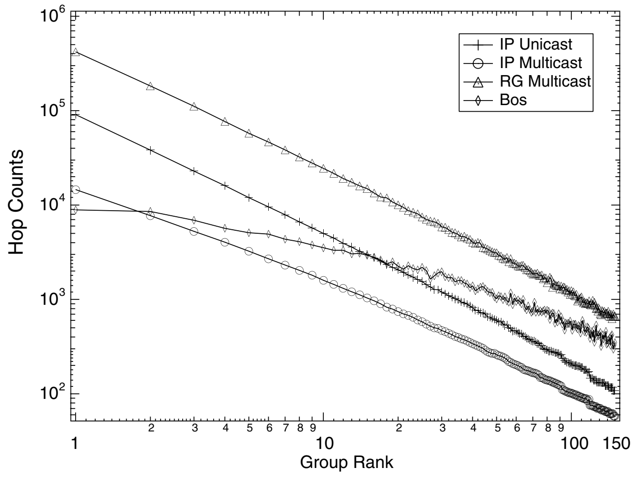


Fig. 25. Hop counts in *Bos* SuperPeers overlay multicasting.

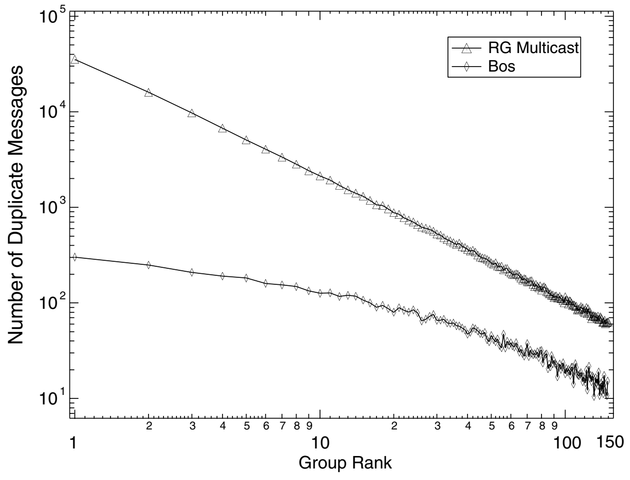


Fig. 26. Total duplicate messages overhead in *Bos* SuperPeers overlay multicasting.

terms of total hop counts than IP multicast for group rank 1 which consists of 100,000 nodes in the network. This confirms that *Bos* is more **cost-effective** than IP multicast in terms of hop counts during overlay multicasting in massive scale network. This is due to the fact that we have a two-tier architecture in *Bos* overlay multicast streaming network. This gives the flexibility of selecting efficient overlay paths through the low-latency and high-bandwidth backbone network connectivity of the SuperPeers at SuperPeers layer. In addition, this allows a smaller number of hop counts in the overlay multicast network as compared to the underlay network.

We observe that multicast in Gnutella-like random graph is the worst performer in terms of hop counts. This is because random graph uses flooding to distribute data, therefore it generates lots of duplicate messages. For large network size (i.e. group rank 1–17), the *Bos* overlay multicast streaming network outperforms the multicast using IP unicast scheme, whereby the cost reduction in terms of hop counts reaches about 90% in group rank 1. Similar to the results of the total delay cost metric observed in Fig. 24, the multicast using IP unicast scheme performs slightly better than *Bos* in small network group size. As shown in Fig. 25, *Bos* slightly slacks in its hop counts' performance metric from group rank 17 onwards (i.e. network group size < 271 nodes). As the network group size shrinks from 271 nodes, the multicast using IP unicast scheme performs slightly better. This is due to the lack of multicast membership and routing complexities in the multicast using IP unicast scheme for small network group size. In this aspect of comparison with multicast using IP unicast scheme, both the results observed in total delay costs and hop counts are similar. We conclude that *Bos* is **more efficient** to perform overlay multicasting in massive network size.

5.2.5. Duplicate messages

This second experiment measures the number of duplicate packets on a single group. Duplicate messages waste network resources and increase the additional load on a network link. Note that the duplicate message for any native IP multicast tree is zero. So in this experiment, measurements are made for the total number of duplicate messages received by end systems (IP layer) in *Bos* and multicast in Gnutella-like random graph, as we have illustrated in Fig. 26.

As expected, the number of duplicate increases with increasing network group size. For multicast in Gnutella-like random graph, the average maximum number of duplicate messages is 35,376 in group rank 1, and the minimum number of duplicate messages is 60.5 in group rank 147, giving an average value of 846.2 over all groups. In contrast, *Bos* gives much better results: the maximum number of duplicate messages is 404.3 in group rank 1 and minimum number of duplicate messages is 10.5 in group rank 132, giving an average maximum number of duplicate messages of 49.4 over all groups. *Bos* overlay multicast streaming **outperforms** multicast in Gnutella-like random graph in terms of the number of duplicate messages generated. The huge reduction in number of duplicate messages in comparison is as much as it reaches 98% in group rank 1. This is because random graph scheme uses flooding algorithm which produces many duplicate messages during multicasting.

We also find that *Bos* has a small average percentage of duplicate messages (less than 5%) received by end nodes over all the groups. Our result is **better** than Scribe which has published result of 16% if the overlay configuration parameter is 4. This results indicate that *Bos* gives a much better overlay multicast performance as compared to multicast in Gnutella-like random graph. This is because of the network awareness of *Bos* during overlay multicast streaming that helps to scale and boost the multicast efficiency and performance.

5.3. Geometric overlay (LST) performance in PlanetLab

We use the same performance metrics that we have used in the massive networks' experiments to quantify the performance of the LST overlay network and *Bos* overlay multicast streaming network in real-world Internet-scale environment (PlanetLab).

5.3.1. Overlay cost vs underlay cost

Similarly, we compare the LST overlay network and underlay network cost on a path-by-path basis for each of the source-destination PlanetLab sites (SuperPeers). Fig. 27 shows the scatterplot performance of the geometric SuperPeers-to-Peers overlay hierarchy for groups of 20, 40, 60 and 81 SuperPeers, respectively. The *X*-axis is the LST overlay network cost and the *Y*-axis is the underlay network cost. The solid linear line gives the indication of network cost being equal to the overlay cost and its purpose is to show this boundary. We confirm the observation of the LST overlay network performance in the massive scale networks. For some cases in the PlanetLab experimental results, using the LST overlay network for communication outperforms the direct Internet communications in the underlying network. The network awareness of the underlying network latencies between nodes helps the construction of the LST overlay network and contributes to the reasonably good overlay performance. Of course, in general, the LST overlay network suffer slightly higher communication cost than the direct Internet communications because it is not possible to achieve the same optimal routing performance as the underlying network.

5.3.2. In/Out-degree of a SuperPeer

The node degree in the overlay network represents the number of overlay network connections to other nodes that must be maintained by that node. A small average node degree results in better load balancing in the overlay network and thus a low underlying network load. That is, when we have small average node degree in the overlay network, it means that there is no network connection in the underlying network that will be significantly congested. Fig. 28 shows the average in-degree/out-degree of PlanetLab sites (SuperPeers) for groups of 20, 40, 60 and 81 SuperPeers. The *X*-axis is the node degree *d*, and the *Y*-axis is the probability density function (pdf) of the node degree in our PlanetLab experiments consisting of different groups of SuperPeers. As expected, the *Yao-graph* in our LST overlay network bounds the out-degree of the SuperPeer to at most 6 which is reasonably small. The average in-degree for every group of SuperPeers is equal to the average out-degree which ranges from 3.7 to 5.2 – this is similar to our experimental results in massive scale networks. Overall, the in-degree results show a wider variance and has lower pdf spread across the groups of SuperPeers. Sim-

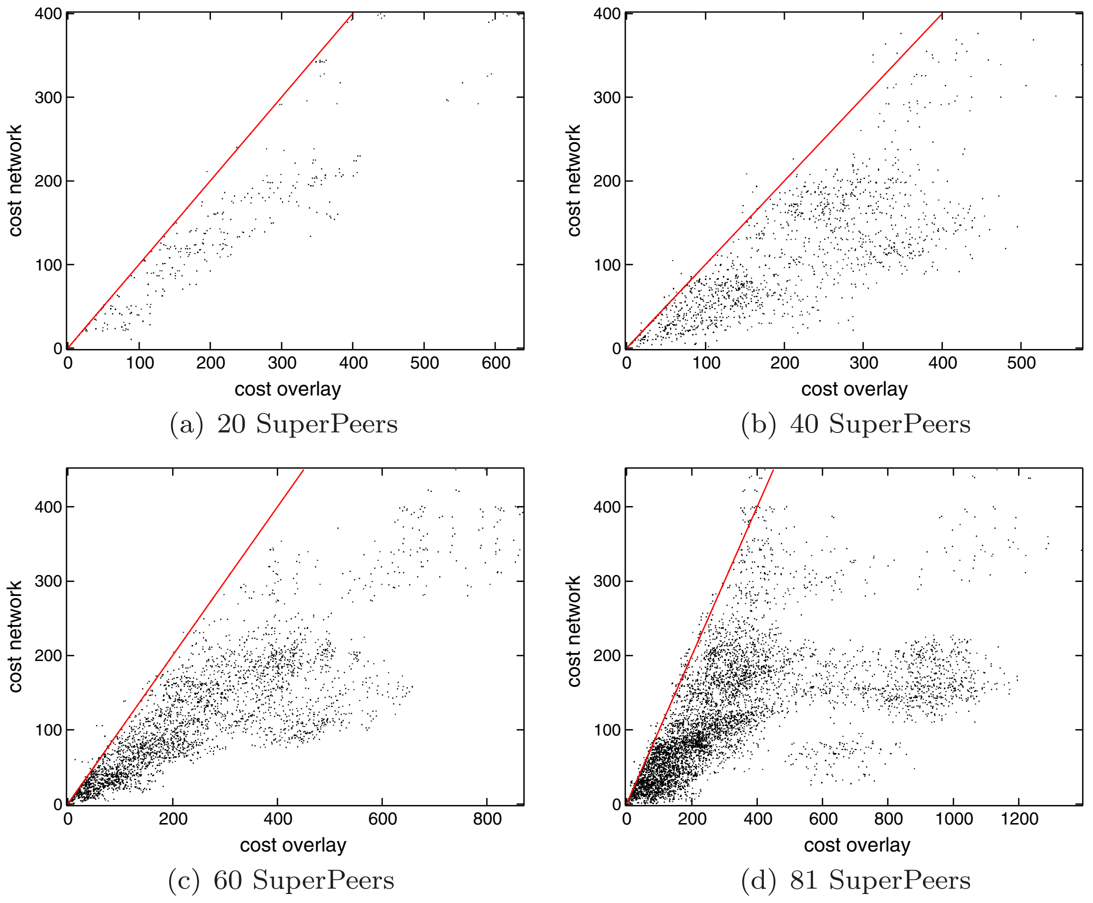


Fig. 27. LST overlay network cost versus underlay network cost in PlanetLab.

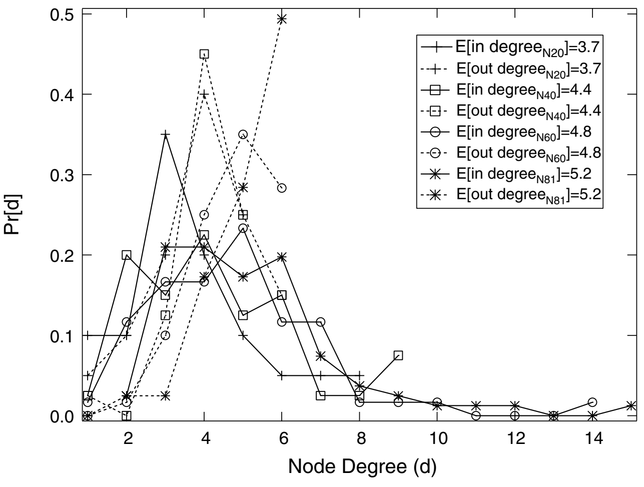


Fig. 28. Distribution of average in-degree/out-degree in the LST SuperPeers overlay.

ilar to the experimental results in massive scale networks, a small average in-degree in our LST overlay network suggests a low *link stress* in the PlanetLab.

5.4. Geometric overlay multicasting (Bos) performance in PlanetLab

We use the performance metrics of link stress, node (site) stress and Relative Delay Penalty (RDP) to quantify the network-aware geometric overlay multicasting (*Bos*) performance in the real-world Internet-scale environment (PlanetLab).

5.4.1. Link stress

We measure the link stress as the number of duplicate packets carried by each network physical link incurred during overlay multicasting in *Bos*. Fig. 29 shows the histogram distribution of the mean link stress for the *Bos* overlay multicast network in differing groups of PlanetLab sites as the elected SuperPeers in *Bos*. That is, it shows the histogram distribution of the mean link stress for all the links in all groups. The results show that most links have **low** stress in the *Bos* overlay multicast streaming network. The average link stress is 2.29, with standard deviation 3.14. This means that the average link stress induced by *Bos* is approximately 2.29 times higher than that for an IP multicast on the same experiment. There are 63.5% (125 links out of 197 links in total) of the links having their mean link stress of 1. The significance lies in the tail of the plots in Fig. 29. In total, there are 20 links having their mean link stress of 5 and above. That is, there are *only* 10% of the total number of links have their mean link stress more than or equal to 5. This is relatively small in number. In general, this results confirm that *Bos* overlay multicast streaming network has low average link stress and thus it has **high efficiency**.

5.4.2. Node (site) stress

We measure the node stress by counting the number of sites in each site's routing table and the number of messages received by each site. Fig. 30 shows the distribution of mean and standard deviation of node (site) stress for each group of PlanetLab sites as the elected as the SuperPeers in *Bos*. Table 1 shows the mean, standard deviation and maximum node stress for each group of PlanetLab sites.

All the results indicate that the mean node stress is small (<2). This suggests that *Bos* is able to spread data over all sites efficiently. In *Bos*, end sites just need to forward multicast messages only to a **small** number of other sites. This results confirm that *Bos* can achieve reasonably **good scalability** by reducing the load imposed on the sites in the *Bos* overlay multicast streaming network. We note that *Bos* performs well in terms of node (site) stress when small number of overlay sites (20 sites out of 230 sites) are elected as the SuperPeers.

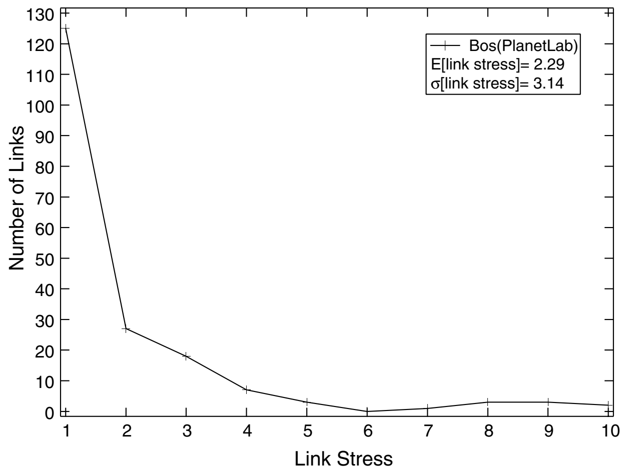


Fig. 29. Average link stress for different SuperPeers group sizes in PlanetLab.

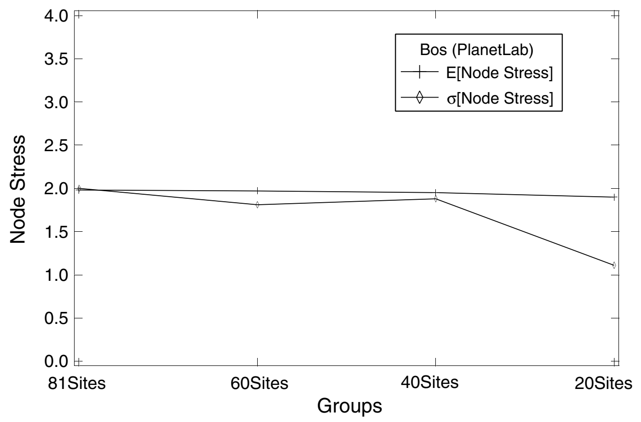


Fig. 30. Average node stress for different SuperPeers group sizes in PlanetLab.

Table 1
Distribution of node stress in PlanetLab

Groups	Mean	SD	Max
81 SuperPeer Sites	1.98	2.01	11
60 SuperPeer Sites	1.97	1.81	11
40 SuperPeer Sites	1.95	1.88	10
20 SuperPeer Sites	1.9	1.12	5

5.4.3. Relative Delay Penalty (RDP)

We evaluate the delay penalty using the Relative Delay Penalty (RDP) performance metric which computes the ratio of overlay multicast network delays between any pairs of SuperPeer group members in *Bos* and their corresponding underlay network delays. Fig. 31 and Table 2 show the average RDP distribution versus differing groups of PlanetLab sites elected as SuperPeers in *Bos*. The average RDPs for different groups are very small ($\leqslant 1.18$) and the overall mean of the average RDPs is 1.15. In addition, for all group members in all groups, more than 84% of the group members have RDP less than 2, and more than 98.5% of the group members have RDP less than 4. All these results further confirm that *Bos* is able to deliver data **efficiently** to the group members with reasonable overlay network delay when compared with its underlay network delay. Hence, we conclude that the overlay multicast streaming performance of *Bos* is acceptable when compared with direct IP-based communication in the underlying network. We also note that *Bos* performs well in terms of RDP when small number of overlay sites are elected as the SuperPeers.

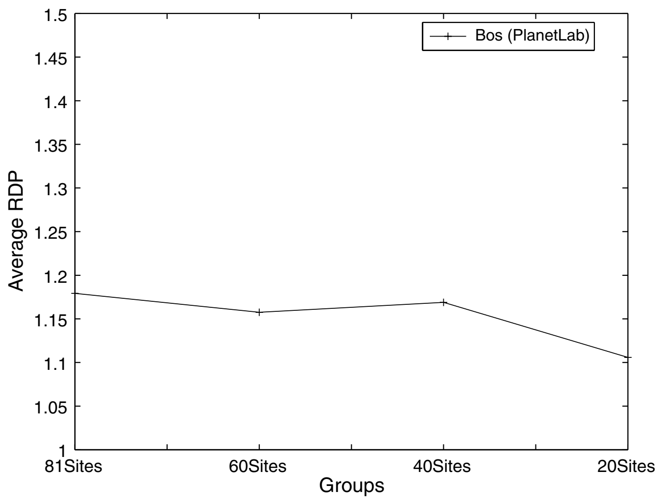


Fig. 31. Average Relative Delay Penalty (RDP) for different SuperPeers group sizes in PlanetLab.

Table 2
Distribution of Average Relative Delay Penalty (RDP) in PlanetLab

Groups	Mean	SD	Max
81 SuperPeer Sites	1.18	1.29	56.99
60 SuperPeer Sites	1.16	0.31	15.74
40 SuperPeer Sites	1.17	0.34	8.95
20 SuperPeer Sites	1.11	0.05	1.96

6. Related work

Scribe [12,13] is a scalable application-level multicast built on Pastry [32]. Multicast overlay maintenance and optimization can be handled locally. The Scribe tree optimization uses a bottleneck remover whereby a parent can request a child to switch its role to a sibling. SplitStream [14] is a high-bandwidth and low-latency multicast service for cooperative environments that is also built on Pastry. It splits the content into multiple stripes and multicast. Each stripe using a separate tree. In this way, the system load will be spread evenly across Peers and it increases resilience and robustness.

Bayeux [33] is a source-specific, application-level multicast scheme that is built on top of Tapestry (which has similar prefix-based routing as Pastry) routing infrastructure. Bayeux trees are source-specific, since the source acts as a centralized management function. Bayeux uses forward-path algorithm in building multicast trees, while Scribe uses reverse path forwarding scheme. Both schemes provides comparable routing delay behaviors. However, Scribe has a lower link stress performance than Bayeux. Bayeux proposes a scheme to scale to a larger numbers of receivers by restricting the service model to a single traffic source and are not well suited for applications such as multi-player games, massive scale distributed collaborative applications.

CAN-Multicast [34] is built on top of Content Addressable Network (CAN), by creating a separate CAN overlay for each multicast group, and then perform flooding of multicast messages to all nodes. CAN-Multicast is capable to scale large group size in a distributed way by exploiting the CAN topologies (i.e. virtual coordinate space) and eliminate the requirement of having multicast routing algorithms to construct distribution trees. Any group member can be a source, and only group members forward the multicast.

Bayeux and CAN-based multicast are similar in that they achieve scalability by leveraging the scalable routing infrastructure provided by CAN and Tapestry. Fundamentally, Bayeux supports only source-specific multicast whilst CAN-based multicast allows any group member to act as a traffic source. Bayeux uses an explicit protocol to set up and tear down a distribution tree from the source to the set of receiver nodes. On the other hand, CAN-based multicast exploits the CAN structure and does not require a routing protocol to explicitly construct distribution trees. Castro et al. [12] show that tree-based overlay multicast built on top of Pastry provides better performance than trees built on top of CAN by making a comparative study of building overlay multicast on these two overlay networks.

Our work (*Bos*) differs from previous research because we exploit accurate and scalable Internet geometry to design a network-aware SuperPeers–Peers geometric overlay hierarchy. We propose the use of *Yao-graph* at the SuperPeers layers to provide a low-latency and high-bandwidth backbone infrastructure to serve the Peers in Peers layer. Optimal-path overlay multicast tree is created at the SuperPeers layer for efficient and scalable overlay multicasting. *Yao-graph* has been used to build power efficient wireless ad hoc networks [19] because of its lightweight local topology maintenance structure.

There are several key advantages in *Bos* overlay multicast streaming network:

- The *Yao-graph* structure enables a new multicast group member to join the group and find the group member with the optimal geometric routes between the multicast members. This also means that the new group member would be able to locate the group member with the best performance metric (i.e. RTTs) between them using the node-to-node geometric distances that are computed in a scalable manner. So, the overlay multicast infrastructure is built with optimized network metric and the multicast group will achieve much better performance as the multicast group grows.
- The multicast tree is an optimal tree structure to ensure that there is no multicast routing loops formed – *Yao-graph* has EMST characteristic. On the other hand, overlay multicast service built on mesh overlay is more complex in eradicating multicast routing loops.
- The LST overlay network has efficient local overlay topology maintenance management and it has less overhead messages between the group members to manage the multicast overlay. This is because multicast membership management is restricted to the immediate *six* closest neighbors. This will help to reduce the overlay multicast service interruption probability from node churn.
- The *Yao-graph* created at the SuperPeers layer is able to bound their number of outgoing links to at most 6. This will help to limit the number of multicast forwarding messages in the outgoing links (i.e. out-degree) of every multicast group member in the multicast tree. So, the outbound network bandwidth of each multicast group member is being controlled to reduce overloading during overlay multicasting. In addition, the network-aware SuperPeers–Peers geometric overlay hierarchy helps to scale the overlay multicasting.

Yao-graph is also used to construct efficient network topology for small mobile ad hoc networks to minimize overhead traffic management. To the best of our knowledge, *Yao-graph* has not been previously studied for the design of network-aware geometric overlay network to provide Internet-scale multicasting.

7. Conclusion

Since we cannot ignore the underlying network metric such as delay latency (RTTs) between nodes to construct efficient overlay network for multicasting, we propose and design a *network-aware geometric* overlay network. We leverage on the observation in our scalability metametric [2] that subspace embedding (Euclidean space) in small partitioned clusters consisting of closer nodes will achieve *better* embedding accuracy. We exploit this result in our Internet geometry (*Highways*) to compute node coordinates in local and global geometric spaces and provide accurate geometric distances between nodes in Euclidean space. Once the locality of nodes in the underlying network has been determined in a scalable manner, the selective placement of nodes in the geometric overlay is done efficiently.

A network-aware SuperPeers–Peers geometric overlay hierarchy is proposed and designed to scale the overlay network communication and management – Lightweight SuperPeers Topology (LST). The SuperPeers layer provides a low-latency and high-bandwidth backbone infrastructure for the Peers in the lower Peers layer. *Yao-graph* is constructed at the SuperPeers layer whereby every SuperPeer is connected to *six* closest neighbors in two-dimensional Euclidean space. We propose to create geometric overlay multicast service (*Bos*) at the LST SuperPeers overlay. Optimal-path overlay multicast tree is built on the *Yao-graph* structure at the SuperPeers layer based on optimal geometric routing paths between SuperPeers derived from the LST's localized random compass routing algorithm.

To evaluate the effectiveness and scalability of our proposals, we conduct simulation experiments on 10 massive scale networks each consisting of 100,000 nodes, and carry out deployment experiments in PlanetLab's real world Internet-scale environment. Our performance evaluation results show that overlay multicasting using network-aware geometric overlay hierarchy approach achieves *high efficiency* and *good scalability* in massive scale and wide-area networks. This is due to the scalable design of network awareness and node locality properties in our geometric overlay multicast streaming network (*Bos*) that help to scale and boost the multicast efficiency, quality and performance.

Acknowledgments

The authors thank Miguel Castro and Manuel Costa (Microsoft Research, Cambridge, UK) for their discussion and providing the massive scale network models, Michael Kleis (Fraunhofer Institute FOKUS) for his contribution to the initial phase of this research project and Michael Smirnov (Fraunhofer Institute FOKUS) for his valuable comments. Xiaoming Zhou and Michael Kleis visited the University of Cambridge Computer Laboratory for 3 weeks with the support from the European Union under the E-NEXT NoE *FP*6 – 506869. Eng Keong Lua was sponsored by Microsoft Research Fellowship and EPSRC e-Science FutureGrid, and Xiaoming Zhou was supported by the NWO SAID Project, for their Ph.D. research studies.

Appendix. The *Yao-graph*

We consider a fully meshed connected graph $G(V,E)$, where V corresponds to a set of points in the Euclidean space $\mathbb{R}^2$, and E to the set of edges with weight corresponding to the Euclidean length of an edge.

Definition 2. Given any finite $V \subset \mathbb{R}^2$ and let $s \in \mathbb{N}$ is an integer. For any pair of nodes u, v, let $S_{u,v}$ denote the sector (or cone) of u containing node v. Suppose that the space around every node $v \in V$ is cut into s sectors with angle $\theta = \frac{2\pi}{s}$. Then the *Yao-graph* $YG_\theta(V)$ of V consists of the following set of links (see Fig. 32):

$$E = \left\{ \begin{array}{l} (u,v)|u,v \in V \text{and there is no} w \in V \\ \text{with } w \in S_{u,v} \text{ and } \|uw\| < \|uv\| \end{array} \right\}$$

Lemma 3. *Let $p \in \mathbb{R}^2$ be a point and S be a sector originating at p. Furthermore, let q and r be two points in S with $\|pq\| \leqslant \|pr\|$. Then $\|qr\| \leqslant \|pr\| - \left(1 - 2sin\left(\frac{\theta}{2}\right)\right)\|pq\|$.*

Proof. Consider Fig. 33. In this figure, q' represents the point on the line from p to r with the same distance to p as q. Applying the triangle inequality to q, q', and r, we get

$$\|qr\| \leqslant \|qq'\| + \|q'r\|$$

$\|qq'\|$ is certainly maximized if q and q' are on opposite sides of the sector. Hence,

$$\|qq'\| \leqslant 2\sin\left(\frac{\theta}{2}\right) \cdot \|pq\|$$

Moreover,

$$\|q'r\| = \|pr\| - \|pq'\| = \|pr\| - \|pq\|$$

Substituting above yields

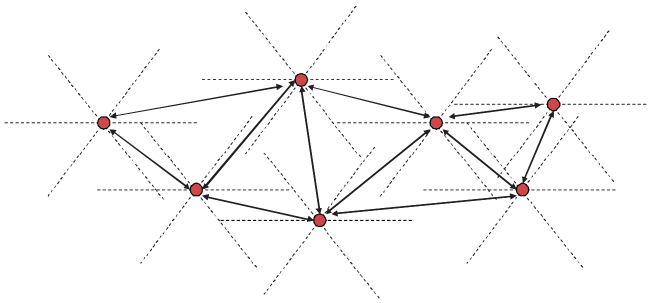


Fig. 32. An example of a *Yao-graph*.

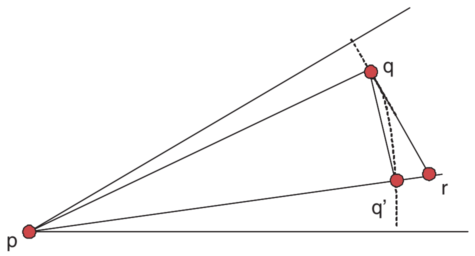


Fig. 33. The sector of p that contains r.

$$\begin{aligned} \|qr\| &\leqslant 2\sin\left(\frac{\theta}{2}\right) \cdot \|pq\| + \|pr\| - \|pq\| \\ &= \|pr\| - \left(1 - 2\sin\left(\frac{\theta}{2}\right)\right)\|pq\| \qquad \square \end{aligned}$$

The drawback of the *Yao-graph* is that, although its out-degree is at most s, its in-degree may be as high as $N-1$ for N nodes. Various subgraphs of the *Yao-graph* have been suggested to remove this drawback. We will present two of them here.

Definition 4. The sparsified *Yao-graph* $S_pYG_\theta(V)$ is a subgraph of $YG_\theta(V)$ with link set

$$E = \left\{ \begin{array}{l} [c]c(u,v) \in E(YG_\theta(V))| \text{ for all } w \in V \\ \text{with } (w,v) \in E(YG_\theta(V)) \\ \text{and } w \in S_{u,v} : \|vw\| > \|uv\| \end{array} \right\}$$

In words, for every sector of every node v, the sparsified *Yao-graph* only keeps the shortest of all links into v. Hence, the sparsified *Yao-graph* has an in-degree of at most s and an out-degree of at most s, and therefore a degree of at most 2s.

Definition 5. The symmetric *Yao-graph* $S_yYG_\theta(V)$ is a subgraph of $YG_\theta(V)$ with link set

$$E = \{(u,v) \in E(YG_\theta(V))|(u,v) \in E(YG_\theta(V))\}$$

In words, the symmetric *Yao-graph* only keeps a link (u,v) if not only v is the nearest neighbor of u in $S_{u,v}$ but also u is the nearest neighbor of v in $S_{v,u}$. Hence, the symmetric *Yao-graph* has a degree of at most s. Obviously,

$$S_yYG_\theta(V) \subseteq S_pYG_\theta(V) \subseteq YG_\theta(V)$$

Fig. 34 shows that there are cases in which the edge sets of the different graphs are proper subsets of each other. Thus, it suffices to prove connectivity for $S_yYG_\theta(V)$ in order to prove connectivity for both variants of the *Yao-graph*.

Definition 6. A Minimum Spanning Tree (MST) of a connected weighted graph with N nodes and W edges, is defined as the tree connecting all the nodes that minimizes the sum of the link weights. That is, a MST is a tree without any loops which connects all the nodes with the lowest cost (weights) among all the candidate trees. A Euclidean MST (EMST) is an MST where weight of a link is its Euclidean distance in the Euclidean space of $\mathbb{R}^2$. Using Kruskal's algorithm [35], the computation of the MST of the connected weighted graph requires $O(W \log N)$. In the case of a geometric graph in two-dimensional geometric space, the calculation of the EMST can be done in $O(N \log N)$[36].

Theorem 7. *Consider any finite $V \subset \mathbb{R}^2$, then the Euclidean Minimum Spanning Tree of V is a subgraph of the Yao-graph.*

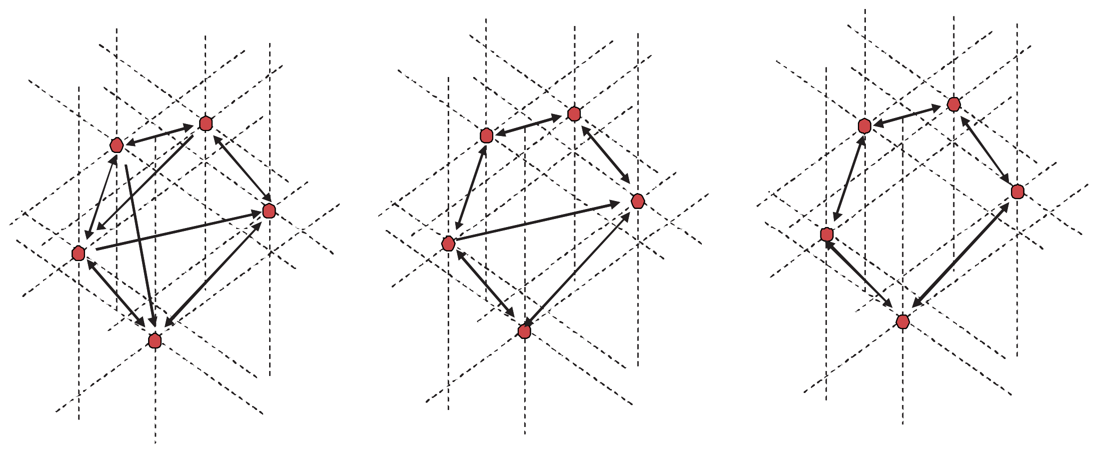


Fig. 34. The *Yao-graph*, the sparsified *Yao-graph*, and the symmetric *Yao-graph* of a point set.

Proof. See A. C.-C. Yao, On constructing minimum spanning trees in k-dimensional spaces and related problem. [8]. □

References

[1] E.K. Lua, J. Crowcroft, M. Pias, Highways: proximity clustering for scalable peer-to-peer network, in: Proceedings of the 4th IEEE International Conference on Peer-to-Peer Computing (IEEE P2P 2004), 2004, pp. 266–267.

[2] E.K. Lua, T. Griffin, M. Pias, H. Zheng, J. Crowcroft, On the accuracy of embeddings for internet coordinate systems, in: Proceedings of the ACM SIGCOMM-Usenix Internet Measurement Conference 2005 (IMC 2005), 2005.

[3] H. Zheng, E.K. Lua, M. Pias, T.G. Griffin, Internet routing policies and round-trip-times, in: Proceedings of the Passive Active Measurement Workshop 2005 (PAM 2005), 2005.

[4] E.K. Lua, T.G. Griffin, Embeddable overlay networks, in: 12th IEEE Symposium on Computers and Communications (IEEE ISCC 2007), Aveiro, Portugal, 2007.

[5] M. Kleis, E.K. Lua, X. Zhou, A case for lightweight SuperPeer topologies, in: KiVS Kurzbeiträge und Workshop, Germany, 2005, pp. 185–188.

[6] M. Kleis, E.K. Lua, X. Zhou, Hierarchical peer-to-peer networks using lightweight SuperPeer topologies, in: Proceedings of the 10th IEEE Symposium on Computers and Communications (IEEE ISCC 2005), La Manga del Mar Menor, Cartagena, Spain, 2005.

[7] E.K. Lua, X. Zhou, Network-aware SuperPeers-peers geometric overlay network, in: 16th IEEE ICCCN 2007 P2P and Grid Networking Track, Honolulu, Hawaii, USA, 2007.

[8] A.C.-C. Yao, On constructing minimum spanning trees in k-dimensional space and related problems, SIAM Journal on Computing 11 (1982) 721–736.

[9] E.K. Lua, X. Zhou, J. Crowcroft, P.V. Miegham, Hierachical geometric overlay multicast network, in: 25th IEEE INFOCOM Conference 2006 Posters and Demo, Barcelona, Spain, 2006.

[10] E.K. Lua, X. Zhou, Bos: massive scale network-aware geometric overlay multicast streaming network, in: IEEE Globecom 2007 P2P Networking Track, Washington, DC, USA, 2007.

[11] PlanetLab home page, Available from: <http://www.planetlab.org>, URL <http://www.planetlab.org>.

[12] M. Castro, M.B. Jones, A.-M. Kermarrec, A. Rowstron, M. Theimer, H. Wang, A. Wolman, An evaluation of scalable application-level multicast built using peer-to-peer overlays, in: IEEE INFOCOM 2003, 2003.

[13] M. Castro, P. Druschel, A.-M. Kermarrec, A. Rowstron, Scribe: a large-scale and decentralized application-level multicast infrastructure, IEEE Journal on Selected Areas in Communication (JSAC) 20 (8).

[14] M. Castro, P. Druschel, A.-M. Kermarrec, A. Nandi, A. Rowstron, A. Singh, Splitstream: High-bandwidth multicast in a cooperative environment, in: ACM Symposium on Operating Systems Principles (ACM SOSP 2003), 2003.

[15] J.B. MacQueen, Some methods for classification and analysis of multivariate observations, in: 5th Berkeley Symposium on Mathematical Statistics and Probability, vol. 1, 1967, pp. 281–297.

[16] E.K. Lua, The structure of internet latency, Ph.D. thesis, University of Cambridge Computer Laboratory, Cambridge, United Kingdom (August 2006).

[17] P.V. Mieghem, Data Communications Networking, Techne press, Amsterdam, The Netherlands, 2006.

[18] M. Pias, J. Crowcroft, S. Wilbur, T. Harris, S. Bhatti, Lighthouses for scalable distributed location, in: 2nd International Workshop on Peer-to-Peer Systems, 2003.

[19] X.-Y. Li, P.-J. Wan, Y. Wang, O. Frieder, Sparse power efficient topology for wireless networks, in: Hawaii International Conference on System Sciences (HICSS 2002), 2002, p. 296.

[20] A. Czumaj, F. Ergun, L. Fortnow, A. Magen, I. Newman, R. Rubinfeld, C. Sohler, Sublinear-time approximation of euclidean minimum spanning tree, in: 14th Annual ACM-SIAM Symposium on Discrete Algorithms, 2003, pp. 813–822.

[21] E. Kranakis, H. Singh, J. Urrutia, Compass routing on geometric networks, in: 11th Canadian Conference on Computational Geometry, Vancouver, Canada, 1999, pp. 51–54.

[22] X.-Y. Li, G. Calinescu, P.-J. Wan, Distributed construction of a planar spanner and routing for ad hoc wireless networks, in: Proceedings of IEEE INFOCOM 2002, vol. 3, New York, USA, 2002, pp. 1268–1277.

[23] Y.K. Dalal, R.M. Metcalfe, Reverse path forwarding of broadcast packets, Communication of the ACM 21 (12) (1978) 1040–1048.

[24] E.W. Zegura, K.L. Calvert, S. Bhattacharjee, How to model an internetwork, in: IEEE INFOCOM 1996, vol. 2, San Francisco, CA, USA, 1996, pp. 594–602.

[25] L. Plissonneau, J.-L. Costeux, P. Brown, Analysis of peer-to-peer traffic on adsl, in: Passive and Active Measurement Workshop 2005 (PAM 2005), 2005.

[26] S. Deering, D. Cheriton, Multicast routing in datagram internetworks and extended lans, ACM Transactions on Computer Systems 8 (2).

[27] S. Deering, D. Estrin, D. Farinacci, V. Jacobson, C. Liu, L. Wei, The pim architecture for wide-area multicast routing, IEEE/ACM Transactions on Networking 4 (2).

[28] D.G. Andersen, H. Balakrishnan, M.F. Kaashoek, R. Morris, Resilient overlay networks, in: ACM Symposium on Operating

Systems Principles (ACM SOSP 2001), Banff, Canada, 2001, pp. 131–145.

[29] Y.-H. Chu, S.G. Rao, H. Zhang, A case for end system multicast, in: ACM SIGMETRICS, 2000, pp. 1–12.

[30] Y. Chawathe, S. McCanne, E.A. Brewer, RMX: Reliable multicast for heterogeneous networks, in: IEEE INFOCOM 2000, Tel Aviv, Israel, 2000, pp. 795–804.

[31] J. Urrutia, Handbook of Wireless Networks and Mobile Computing, John Wiley & Sons, 2002, Ch. 18: Routing with guaranteed delivery in geometric and wireless networks, pp. 393–406.

[32] P. Druschel, A. Rowstron, Pastry: scalable distributed object location and routing for large-scale peer to peer systems, in: Proceedings of the 18th IFIP/ACM Middleware 2001, 2001.

[33] S.Q. Zhuang, B.Y. Zhao, A.D. Joseph, R.H. Katz, J.D. Kubiatowicz, Bayeux: an architecture for scalable and fault-tolerant wide-area data dissemination, in: ACM NOSSDAV, 2001, pp. 11–20.

[34] S. Ratnasamy, M. Handley, R.M. Karp, S. Shenker, Application-level multicast using content-addressable networks, in: Networked Group Communication, 2001, pp. 14–29.

[35] T.H. Cormen, C.E. Leiserson, R.L. Rivest, C. Stein, Introduction to Algorithms, second ed., MIT Press and McGraw-Hill, 2001.

[36] M.I. Shamos, D. Hoey, Closest-point problems, in: 16th IEEE Symposium on Foundations of Computer Science, 1975, pp. 151–162.

Eng Keong Lua is currently a Researcher in the NTT Service Integration Laboratories, Nippon Telegraph and Telephone Corporation at Tokyo, Japan, since November 2006. In 2006, he obtained his Ph.D. degree in Computer Science from the University of Cambridge Computer Laboratory, United Kingdom. From 2003 to 2006, his research studies were sponsored by Microsoft Research Fellowship and EPSRC e-Science FutureGrid. In 2004, he had also completed a 6-month research internship as a Researcher at the Intel Research Laboratory, Cambridge, UK. He was the inaugural recipient of the prestigious Australia-Asia Award in 2003, presented by the Australian Government to promising research scholars in Asia for support of research activities in Australia. He received his M.Sc. in Telecommunications with Distinction from the Department of Electronic and Electrical Engineering, University College London, United Kingdom, in 1997, and both Postgraduate Diploma of Teaching in Higher Education and B.A.Sc in Computer Engineering with Merit from the Nanyang Technological University (NTU), Singapore, in 1995 and 1993 respectively. His current research areas are telecommunications, networking and distributed systems. His main research interests include overlay networking, Peer-to-Peer (P2P) overlay systems, network-aware (locality-aware) overlay multicast streaming network, meta-routing framework (network routing algebra) for the design of network routing protocols in Internet and wireless networks, network security and ubiquitous networking. He is a Member of the IEEE Computer and Communications Societies, ACM, IET, and IES. He holds Cisco Certified Academic Instructor (CCAI), Cisco Certified Network Professional (CCNP) and Cisco Certified Network Associate (CCNA) professional certifications.

Xiaoming Zhou received M.Sc. degree in electrical engineering in Sep. 2002 from Delft University of Technology, the Netherlands. After that he did a research topic on Internet performance analysis funded by a DELTA scholarship. In January 2003 he began his Ph.D. work at Network Architectures and Services (NAS) group at TU Delft. His research project was mainly focused on Internet performance measurement and analysis at both application and IP layers. Such as Peer-to-Peer system, overlay multicast system, and the users perceived quality of Voice-over-IP applications. He obtained his PhD degree in November 2006. Since January 2007, he has been working as a research scientist in Philips Research, Eindhoven. His current research areas focus on the convergence of different networks (e.g. Internet, Mobile, TV).

Jon Crowcroft is the Marconi Professor of Networked Systems in the Computer Laboratory, University of Cambridge. Prior to that, he was Professor of networked systems at UCL in the Computer Science Department. He is a Fellow of the ACM, a Fellow of the British Computer Society and a Fellow of the IEE and a Fellow of the Royal Academy of Engineering, as well as a Fellow of the IEEE. He was a member of the IAB; was general chair for the ACM SIGCOMM 95–99. He has published 5 books–the latest is the Linux TCP/IP Implementation, published by Wiley in 2001. He is the Principle Investigator in the Computer Laboratory for the EU Haggle Project in DTN, the EPSRC TINA project on location sensors and wireless networking of airports, and for the ITA project in next generation wireless networks.

Piet F.A. Van Mieghem is professor at the Delft University of Technology with a chair in telecommunication networks and chairman of the basic unit Network Architectures and Services (NAS). His main research interests lie in new Internet-like architectures for future, broadband and QoS-aware networks and in the modelling and performance analysis of network behavior and complex infrastructures. Professor Van Mieghem received a Master's and Ph. D. in Electrical Engineering from the K.U.Leuven (Belgium) in 1987 and 1991, respectively. Before joining Delft, he worked at the Interuniversity Micro Electronic Center (IMEC) from 1987 to 1991. During 1993 to 1998, he was a member of the Alcatel Corporate Research Center in Antwerp where he was engaged in performance analysis of ATM systems and in network architectural concepts of both ATM networks (PNNI) and the Internet. He was a visiting scientist at MIT (Department of Electrical Engineering, 1992–1993) and, in 2005, he was visiting professor at ULCA (Department of Electrical Engineering). He was member of the editorial board of the journal Computer Networks from 2005–2006. He is the author of two books: Performance Analysis of Communications Networks and Systems, Cambridge University Press, 2006 and Data Communications Networking, Techne Press, Amsterdam, 2006.