On Maximizing Tree Bandwidth for Topology-Aware Peer-to-Peer Streaming

Xing Jin, *Student Member, IEEE*, W.-P. Ken Yiu, *Student Member, IEEE*, S.-H. Gary Chan, *Senior Member, IEEE*, and Yajun Wang

***Abstract*—In recent years, there has been an increasing interest in peer-to-peer (P2P) multimedia streaming. In this paper, we consider constructing a high-bandwidth overlay tree for streaming services. We observe that underlay information such as link connectivity and link bandwidth is important in tree construction, because two seemingly disjoint overlay paths may share common links on the underlay. We hence study how to construct a high-bandwidth overlay tree given the underlay topology.**

We formulate the problem as building a *Maximum Bandwidth Multicast Tree (MBMT)* or a *Minimum Stress Multicast Tree (MSMT)*, depending on whether link bandwidth is available or not. We prove that both problems are NP-hard and are not approximable within a factor of $(2/3 + \epsilon)$, for any $\epsilon > 0$, unless $P = \mathrm{NP}$. We then present approximation algorithms to address them and analyze the algorithm performance. Furthermore, we discuss some practical issues (e.g., group dynamics, resilience and scalability) in system implementation. We evaluate our algorithms on Internet-like topologies. The results show that our algorithms can achieve high tree bandwidth and low link stress with low penalty in end-to-end delay. Measurement study based on PlanetLab further confirms this. Our study shows that the knowledge of underlay is important for constructing efficient overlay trees.

***Index Terms*—Overlay tree, peer-to-peer streaming, topology-aware, tree bandwidth.**

I. Introduction

With the popularity of broadband Internet access, there has been an increasing interest in media streaming services. Recently, P2P streaming has been proposed and developed to overcome limitations in traditional server-based streaming. In a P2P streaming system, cooperative peers self-organize themselves into an overlay network via unicast connections. They cache and relay data for each other, thereby eliminating the need for powerful servers from the system.

In this paper, we consider building a high-bandwidth overlay tree for steaming. A streaming video usually has a certain bitrate. To ensure good streaming quality at a host, the incoming bandwidth of the host should be higher than or equal to the streaming bitrate. We hence need to build a tree with enough tree bandwidth to support the streaming. Here *tree bandwidth* is defined as the minimum path-bandwidth of an overlay tree [1]. Note that average incoming bandwidth of hosts may also be used to evaluate a tree. However, when the incoming bandwidth of a host is lower than the streaming bitrate, the host will encounter packet loss and the received streaming quality will be reduced. Average incoming bandwidth cannot indicate the portion of hosts with good (or bad) streaming qualities. We hence focus on tree bandwidth instead of average incoming bandwidth in this paper.

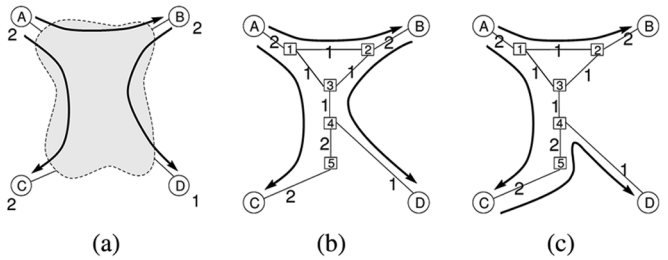

Fig. 1. The impact of underlay information to overlay tree construction. (a) Naive construction of an overlay tree (the label along a host is its degree bound). (b) Tree bandwidth $= 0.5$ (the label along a link is the residual bandwidth of the link). (c) A tree with higher tree bandwidth of 1.0.

Some previous work imposes degree bounds on hosts. That is, the degree of a host in an overlay tree cannot exceed a certain bound [2], [3]. This approach can effectively reduce congestion near hosts. However, it cannot improve tree bandwidth if the bandwidth bottlenecks occur at intermediate links in paths instead of edge links near hosts. Furthermore, two seemingly disjoint overlay paths may share common underlay links; therefore the selection of overlay paths without the knowledge of underlay may lead to serious link congestion. Note that in this paper, a *link* refers to a physical underlay connection between two routers or between a router and a host; while a *path* refers to an end-to-end connection between two hosts, which may contain multiple underlay links. Furthermore, the bandwidth of links or paths refers to the residual bandwidth along links or paths. We show a tree construction example in Fig. 1. A, B, C, and D are four hosts, whose degree bounds are 2,2,2 and 1, respectively. Rectangles with labels 1 to 5 are routers. Without any knowledge of the underlay, a tree may be constructed as Fig. 1(a) shows. Suppose that the link connectivity and link bandwidth on the underlay is as shown in Fig. 1(b). We can see that this tree is of bandwidth 0.5 (the bottleneck link is between routers 3 and 4). Given the underlay information, we can in fact build a tree with higher tree bandwidth of 1 as shown in Fig. 1(c). Therefore, underlay information is important for building a high-bandwidth overlay tree.

Manuscript received October 31, 2006; revised July 29, 2007. This work was supported in part by the Research Grant Council of the Hong Kong Special Administrative Region (HKUST611107) and Hong Kong Innovation and Technology Commission (GHP/045/05). The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Klara Nahrstedt.

The authors are with the Department of Computer Science and Engineering, Hong Kong University of Science and Technology, Kowloon, Hong Kong, China (e-mail: csvenus@cse.ust.hk; kenyiu@cse.ust.hk; gchan@cse.ust.hk; yalding@cse.ust.hk).

Digital Object Identifier 10.1109/TMM.2007.907459

In this paper, we study how to construct a high-bandwidth overlay tree given the underlay information. Given a group of hosts, we assume that the router-level underlay topology among them is available. By topology, we mean the link connectivity and delay information (the inference of a topology will be discussed in Section II-A). Residual bandwidth along links is much more difficult to obtain due to dynamic background flows and high measurement cost. We hence investigate novel algorithms in the following two cases:

- In the absence of link bandwidth: Define *link stress* as the number of copies of a packet transmitted over a certain physical link [2]. We consider minimizing the maximal link stress in a tree. We formulate the problem as building a *Minimum Stress Multicast Tree (MSMT)* on a given topology. Our study shows that it is NP-hard and is not approximable within a factor of $(2/3+\epsilon)$ for any $\epsilon > 0$, unless $P = \mathrm{NP}$. We then propose an approximation algorithm to address it, and analyze the correctness and performance of the algorithm.
- In the presence of link bandwidth: When link bandwidth is available, we build a *Maximum Bandwidth Multicast Tree (MBMT)* on the topology. This problem is also NP-hard since it is equivalent to MSMT if all the links have the same bandwidth. We explore how to extend the approximation algorithm for MSMT to address the MBMT problem.

Besides the algorithms, we further discuss practical issues in constructing the tree, including host joining/leaving, system resilience and scalability. We have conducted simulations on Internet-like topologies as well as Internet measurements on PlanetLab to evaluate our approximation algorithms. The results show that they can achieve higher tree-bandwidth and lower link stress than traditional tree-based protocols such as Narada, Overcast, and TAG. Our results show that underlay information can help build a highly efficient overlay tree.

In the paper, we assume that the network is symmetric in terms of bandwidth. Our algorithms are hence applicable to symmetric networks like Symmetric Digital Subscriber Line (SDSL) or local area networks. We further assume that the router-level path from a host A to another host B is the reverse of the path from B to A.

The rest of the paper is organized as follows. In Section II we briefly review related work. In Section III, we formulate the MSMT problem and prove that it is NP-hard. We then propose an approximation algorithm to address it. In Section IV we formulate the MBMT problem and extend the approximation algorithm for MSMT to address it. In Section V we study the system dynamics, resilience and scalability issues. In Section VI, we present illustrative numerical results based on Internet-like topologies and PlanetLab measurements. We finally conclude in Section VII.

II. Background and Related Work

A. Topology Inference

There are many ways to infer a network topology. Network tomography techniques periodically probe the network and exploit the performance in correlation to infer network topologies [4], [5]. However, because the network properties measured (e.g., loss rate or delay) are often unstable and inaccurate, it is difficult to infer an accurate topology. Border Gateway Protocol (BGP) routing tables can provide AS-level information, but they usually are not available to normal hosts in the Internet [6], [7]. Therefore, traceroute-like tools (e.g., [8], [9]) that can obtain explicit router-level information by end hosts have been widely used in Internet measurements [10]–[13]. For example, in Max-Delta inference, hosts first use some tools to estimate their coordinates [13]. A server then collects host coordinates and chooses the best set of host-pairs to traceroute. It has been shown that Max-Delta can infer a highly accurate topology with a low number of traceroutes. A problem in traceroute measurement is that traceroute results often contain anonymous routers, which significantly distort and inflate the inferred topology. Yao *et al.* study how to infer a topology given the traceroute results, where the inferred topology should be consistent with the traceroute results and contain the minimum number of anonymous routers [14]. They show that producing either an exact or an approximate solution for this problem is NP-hard. Two practical algorithms without the consistency constraint are then proposed to quickly construct an approximate topology [13].

Traceroute can only infer link connectivity and delay but not link bandwidth. We may use Pathchar to measure the bandwidth of each link in an overlay path [9]. But it takes long measurement time and consumes much network bandwidth. In fact, it needs in the order of minutes to estimate the bandwidth of a link. If the cost of measuring link bandwidth cannot be afforded, users can turn to measuring the bandwidth of an overlay path, which can be more easily obtained [15]–[17]. For example, Pathload is a lightweight tool for such purpose which does not incur significant increase in network utilization, delay or loss during measurements and can support concurrent bandwidth measurements toward the same host [15]. We hence propose a simple and fast approach to infer link bandwidth based on path bandwidth measurement in Section VI-B.

B. Constructing Data Delivery Trees

1) IP Multicast Trees: Multimedia streaming needs to send information from one or multiple source hosts to a set of destination hosts. A traditional delivery technique for streaming is IP multicast [18]. In an IP multicast tree, the interior nodes are all routers and the leaves are all hosts. Routers are responsible for replicating and forwarding packets along the tree. There have been many routing algorithms for IP multicast in the literature. Examples include Protocol Independent Multicast (PIM) [19], Distance Vector Multicast Routing Protocol (DVMRP) [20], Multicast Open Shortest Path First (MOSPF) [21], and Core Based Trees (CBT) [22]. Please refer to [23] for a comparison of these technologies.

IP multicast requires significant change to the current Internet infrastructure. It requires multicast-capable routers, which need to maintain per group state for packet replication and are not scalable. Furthermore, IP multicast lacks large-scale commercial management functions such as multicast address allocation and reliable transmission. Therefore, although IP multicast has been proposed for over ten years, it has not been widely deployed yet. As a comparison, in a P2P overlay network, hosts are responsible for packets replication and forwarding. A P2P

network only uses unicast and does not need multicast-capable routers. It is hence more deployable and flexible.

In addition, in IP multicast, there is no need to consider the link sharing issue. Any link in IP multicast has stress 1, regardless of the number of hosts in the multicast session. However, in a P2P network, a link may be shared by multiple paths, and a major issue is how to reduce link stress. Therefore, the construction of an IP multicast tree is significantly different from a P2P overlay tree. The methods and algorithms for IP multicast cannot be straightforwardly applied to a P2P network.

2) Peer-to-Peer Overlay Trees: As discussed, P2P networks do not rely on multicast-capable routers and can be directly deployed on the current Internet. Typical examples include application-layer multicast (ALM) [2], [3], [24]–[26], P2P video-on-demand systems [27], [28], and mesh-based streaming systems [29], [30].

Trees with no Topology Information: Narada is one of the pioneering ALM protocols and aims at building a low-delay overlay tree [2]. In Narada, a joining host first obtains a full list of already joined hosts from a public rendezvous point (RP) and randomly selects a few to set up connections. A host then periodically pings some randomly selected hosts. It adds a new connection into the mesh if the new connection is short enough. It also drops long connections from the mesh. After constructing the mesh, a shortest path tree is built on top of the mesh for data delivery. In Narada, the number of neighbors of a host in the mesh is restricted within a certain range to prevent the host from having too many connections. As discussed above, this approach cannot improve tree bandwidth if the bandwidth bottlenecks occur at intermediate links in the network. Our tree construction method does not depend on the locations of bottleneck links. We infer the bandwidth of links and build a tree circumventing bottleneck links to achieve high tree bandwidth.

Overcast is one of the few ALM protocols that consider tree bandwidth as the primary objective [24]. When a new host arrives, it first estimates its bandwidth to the root and to each of the root's children. If the bandwidth to any of these children is close to the bandwidth to the root, the new host moves one level down to this child and repeats the process. This procedure continues until the current host has no children with satisfactory bandwidth to the new host. Then the current host becomes the new one's parent. Overcast does not perform well for a large group of hosts (as shown in our simulations in Section VI). Later joining hosts often cannot find parents with enough forwarding bandwidth. Furthermore, a new host is inserted into the tree as far from the source as the bandwidth constraint allows. End-to-end delay in Overcast is hence high. As our simulation results show, our algorithms can achieve higher tree bandwidth and lower delay than Overcast.

Topology-Aware Overlay Trees: Topology-Aware Grouping (TAG) is a topology-aware ALM protocol which aims at constructing a low-delay overlay tree [31]. In TAG, a new host needs to measure the router-level path from the source to itself before joining the tree, which is called the *spath* of the host. Each host then selects as its parent the host whose spath has the maximal overlap with its own spath. In this way, TAG can reduce the numbers of underlay hops and hence the delay over the unicast paths. Their study indicates that knowing underlay topology is important to construct efficient overlay trees, and lots of work continues in this area [32]–[34]. However, all these studies focus on reducing end-to-end delay and consider tree bandwidth as secondary or not at all. Different from them, we take tree bandwidth as the primary metric and endeavor to maximize tree bandwidth.

Fast Application-layer Tree (FAT) builds an overlay tree based on topology inference to achieve a certain target tree bandwidth [1]. The target tree bandwidth is a pre-defined system parameter and is used in tree construction. If the resultant tree achieves the target tree bandwidth, FAT will not continue improving the tree. As a comparison, our algorithms in this paper aim to achieve the maximum tree bandwidth. We have analyzed the hardness of the MSMT and MBMT problems and shown that they are NP-hard.

Cui *et al.* study optimal resource allocation in a topology-aware overlay tree [35]. In their settings, hosts form an overlay tree and each host receives data from its parent at a certain rate. Given the underlay topology among hosts (with residual link bandwidth), they study how to maximize the aggregate utility of all hosts under the bandwidth constraint. They allow hosts to have different data receiving rates, which is applicable to file sharing applications. Our study in this paper focuses on streaming applications, where the data receiving rate at a host is a constant (i.e., the video bitrate). To cast our problem into their settings, we need to set $x_{f_1} = x_{f_2}, \forall f_1, f_2 \in F$ and $U_f(x_f) = x_f$ (see (4)–(7) in [35]). That is, each flow in the tree has a constant transmission rate and the utility function is the transmission rate. However, this utility function U_f is not strictly concave and its curvatures are not bounded away from zero. This violates constraints $A1$ and $A2$ in [35]. Hence, the results and algorithms in [35] cannot be applied to our problem.

Cohen *et al.* also study how to maximize tree bandwidth given the underlay topology [36]. Their MPSP(MaxBottleneck) problem is similar to the MBMT problem formulated in our paper. However, our work is different from theirs in several ways. 1) We use a different routing model from theirs. In Cohen's model, the routing path between two hosts can be an arbitrary router-level path between them. But in our model, we enforce the routing path between two hosts to be the shortest path between them on the underlay topology. In the real Internet, the routing path between two hosts is determined by the routing protocols, e.g., OSPF for intra-AS routing and BGP for inter-AS routing. The path cannot be determined by end hosts. Our model is hence more realistic than Cohen's. In other words, our model gives one additional constraint to the problem, and we will show later that this constraint is not trivial. 2) Given the network model, we provide a novel proof of the problem NP-hardness. Following our proof, we can show that the MSMT and MBMT problems are not approximable within a factor of $(2/3 + \epsilon)$, for any $\epsilon > 0$, unless $P = \mathrm{NP}$. 3) We provide new heuristics to address the problems. Our heuristic for MSMT has been proven to achieve a constant approximation ratio.

In detail, Cohen *et al.* have proposed two heuristics to address the MPSP (MaxBottleneck) problem, i.e., Widest Path Heuristic (WPH) that achieves $O(|V_o|)$ approximation ratio (where $|V_o|$ is the number of hosts) and Double Tree Heuristic (DTH) that

achieves constant approximation ratio 2 in undirected graph. We find that there are some limitations in their heuristics. Firstly, the ratio of $O(|V_o|)$ is easy and can be achieved in many other ways. For example, we can start from an empty set and keep adding paths with the maximum bandwidth until all hosts are connected. Denote the final graph as G_S. We can prove that any spanning tree T of G_S can achieve $O(|V_o|)$ approximation ratio[1]. Secondly, DTH uses a similar method as the folklore 2-approximation Steiner tree algorithm. Its approximation ratio of 2 comes from the fact that we can use any arbitrary paths between hosts for routing so that we can schedule the paths in the tree and guarantee that one link is shared by at most two unicast paths. Clearly, this is not true in our model. As a comparison, our heuristics do not assume arbitrary routing between hosts. Our heuristic can achieve a constant approximation ratio for MSMT (see Corollary 1 for more details). Our result has significantly improved previous study on the MSMT problem.

III. Minimum Stress Multicast Tree (MSMT)

In this section, we discuss how to build a Minimum Stress Multicast Tree on an underlay topology. We formulate the problem and show that it is NP-hard. We then propose an approximation algorithm to address it.

A. Problem Formulation and Hardness Analysis

We consider that hosts are in position before tree construction. For example, there are a group of Internet participants waiting for a video conference, or a content distribution network (CDN) needs to distribute data to some pre-deployed proxies.

In our network model, the underlay topology consists of a set of routers, which are inter-connected by physical links. A host is attached to a router, and an overlay path between a pair of hosts is the shortest path between them (in terms of delay) on the underlay. To cast the problem in a more general and realistic setting, we consider that there is a set of overlay paths to choose from, termed the "path set." This path set indicates which host is allowed to connect to which in the overlay tree and the corresponding underlay links. This constraint may be due to some administrative policies, network constraints, costs, firewalls, etc. If there is no constraint on the overlay paths, the set is complete. Note that any two hosts should be able to reach each other through the paths in the set. We further define some notations as follows:

Definition 1:

- An *underlay network* is a connected graph $G = (V, E, C)$, where V is a set of vertices (i.e., routers and hosts), E is a set of undirected edges (i.e., physical links among routers/hosts) and $C : E \to R^+$ is a cost function defined on the edges (e.g., link delay).
- An overlay *path set* is a connected graph $G_o = (V_o, E_o)$, with the set of hosts $V_o \subseteq V$ and each edge $e = (u, v) \in E_o$ being the shortest path between u and v in the underlay network G.
- A *multicast tree* T is a spanning tree of the overlay path set G_o which spans all the hosts in V_o and whose edges belong to E_o.

We take Fig. 1(b) as an example. The underlay network is the whole graph in Fig. 1(b), i.e., $V = \{A, B, C, D, 1, 2, 3, 4, 5\}$ and $E = \{A-1, 1-2, 2-B, 1-3, 2-3, 3-4, 4-5, 5-C, 4-D\}$. If there is no constraint on the overlay paths, the overlay path set contains the all-pairs paths between hosts. That is, the set of hosts $V_o = \{A, B, C, D\}$ and the set of overlay paths $E_o = \{A-B, A-C, A-D, B-C, B-D, C-D\}$. However, if path $A - D$ cannot be used in the tree because, say, they are behind different Network Address Translations (NATs) and cannot directly communicate with each other, this path should be removed from E_o. The paths $\{A - B, A - C, B - D\}$ form a multicast tree T, since all these paths are from E_o and they span all the hosts in V_o.

Note that an edge in G (which is an underlay link) may be shared by multiple edges in T (which are overlay paths). We define tree stress as follows.

Definition 2: Denote $E_s(T) \in E$ the set of underlay links that are used in tree T. The *stress* of a link $e, t(e, T)$, is defined as the number of edges in T (which are overlay paths) that cross e. The *stress* of T is the maximum stress over all the links in $E_s(T)$.

In Fig. 1(b), suppose $T = \{A - B, A - C, B - D\}$. $E_s(T)$ contains all the links in T, i.e., $E_s(T) = \{A - 1, 1 - 2, 2 - B, 1 - 3, 2 - 3, 3 - 4, 4 - 5, 5 - C, 4 - D\}$. Among them, the links $1 - 2, 1 - 3, 2 - 3, 4 - 5, 5 - C$ and $4 - D$ are traversed only once by the paths, thereby of stress 1. The link $A - 1$ is traversed by paths $A - B$ and $A - C$, and hence has stress 2. Similarly, the links $2 - B$ and 3–4 have stress 2. As a result, the stress of T is 2.

We now present the formulation of MSMT and our results on the hardness of the MSMT problem.

Definition 3: Given an underlay network G and an overlay path set G_o, the *Minimum Stress Multicast Tree (MSMT)* is a spanning tree of G_o with the minimum tree stress.

Theorem 1: Given an underlay network G and an overlay path set G_o, it is NP-hard to compute the Minimum Stress Multicast Tree of G_o. Furthermore, MSMT is not approximable within a factor of $(2/3 + \epsilon)$ for any $\epsilon > 0$, unless $P = \text{NP}$.

The theorem can be obtained by a reduction from the known NP-hard problem, *Minimum Degree Spanning Tree (MDST)*. See Appendix I for the proof.

B. An Approximation Algorithm to MSMT

We now present an approximation algorithm to address the MSMT problem. Let $\Delta^*(G, G_o)$ denote the stress of a MSMT on an underlay network G and its corresponding overlay path set G_o. By "removing a link from G," we mean that the link is removed from G and all the overlay paths crossing that link are removed from G_o.

[1]Proof skeleton: Assume that the last overlay path added into G_S is P_L, with residual bandwidth w_L. We can show that the optimal tree bandwidth cannot be higher than w_L. If we can build a tree with tree bandwidth higher than w_L, each path should have residual bandwidth of higher than w_L. According to the mechanism of adding paths to G_S, these paths will be added to G_S before P_L. Clearly, after adding these paths, all the hosts have been connected. We will stop adding other paths and P_L cannot be the last path we add. This contradicts the assumption that P_L is the last path added into G_S.

Given an arbitrary tree T on G_S, there are at most $(|V_o| - 1)$ paths in T sharing the same link. On the other hand, any link in G_S has w_L or higher residual bandwidth. The tree bandwidth of T is hence at least $w_L/(|V_o| - 1)$. As the optimal tree bandwidth cannot be higher than w_L, the tree bandwidth of T is at least $1/O(|V_o|)$ of the optimal tree bandwidth.

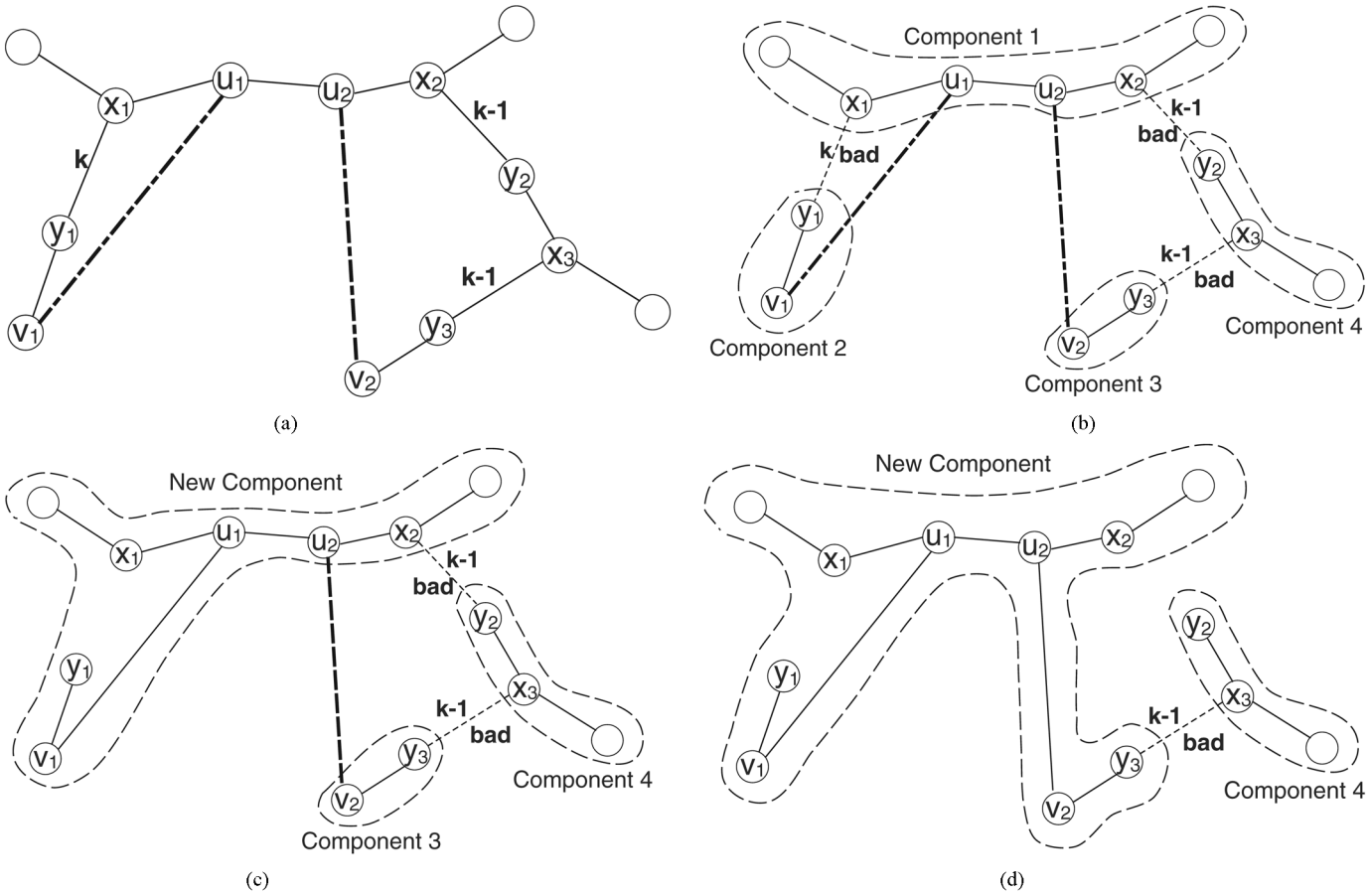

Fig. 2. Example of the MSMT approximation algorithm. (a) An arbitrary spanning tree T with stress k. (b) Removing S_{k-1} from T and generating a forest F. (c) An improvement to T by adding (u_1, v_1) and removing (x_1, y_1). (d) An improvement to T by adding (u_2, v_2) and removing (x_2, y_2).

Theorem 2: Let G be an underlay network and G_o be the corresponding overlay path set. Let T be a tree of G_o with stress k. Let S be the set of links of stress k in $E_s(T)$, and let B be an arbitrary subset of links of stress $k-1$ in $E_s(T)$. Remove $S \cup B$ from G, and hence break the tree T into a forest F. Suppose that the remaining G_o does not contain any overlay paths between different trees in F, and that an overlay path in T contains at most $x (x > 0)$ different links of stress larger than or equal to $k-1$. Then $k \leq \min((|S|+|B|)\times\Delta^*(G, G_o), x\times\Delta^*(G, G_o)+|B|/(|S|+|B|))$.

The proof is shown in Appendix II. Based on this theorem, we provide a polynomial-time algorithm that approximates the MSMT problem. The algorithm starts with an arbitrary spanning tree T of G_o, and seeks to reduce its stress. Let the stress of T be always k. Denote S_i the set of underlay links in $E_s(T)$ whose stresses are at least i. The following operation forms the building block of the algorithm:

Definition 4: Let (u, v) be an edge (overlay path) of G_o which is not in T. Let Y be the cycle generated when (u, v) is added to T. Suppose there exists an overlay path p consisting of links of stress k in Y, and the stresses of links in path (u, v) are at most $k-2$. An *improvement* to T is the modification of T by adding the path (u, v) to T and deleting the path in Y that consists of the most links of stress k.

We illustrate the above definitions in Fig. 2(a). The nodes in the figure are hosts and the solid lines between them are overlay paths. The solid lines form a tree T. Suppose the stress of T is k. Further suppose that the maximum stress k occurs in path (x_1, y_1), and the maximal stresses in paths (x_2, y_2) and (x_3, y_3) are both $k-1$. The maximal stress of links in all the other paths of T is less than $k-1$. If (u_1, v_1) is a path in G_o but not in T, and the stresses of links in path (u_1, v_1) are at most $k-2$, we can add (u_1, v_1) into T and delete (x_1, y_1) from T. This replacement is an improvement to T because 1) adding (u_1, v_1) into the tree only introduces links of stress no larger than $k-1$ and 2) deleting (x_1, y_1) reduces the size of S_k.

We present the approximation algorithm as follows. The algorithm starts with an arbitrary tree T with stress k. At the beginning of each phase of the algorithm, all underlay links in S_{k-1} (i.e., links of stress k or $k-1$ in $E_s(T)$) are removed from T as well as from G and G_o, and marked as *bad*. All other links are marked *good*. We call an overlay path of T consisting of at least one bad link a *bad path*, and an overlay path of T consisting of only good links a *good path*. Suppose T is broken into several components by removing S_{k-1}, each of which is called a *good component*. If there are no paths of G_o between good components, the algorithm stops. In this case, Theorem 2 shows that k is within a certain range of the optimal result $\Delta^*(G, G_o)$. Otherwise, let (u, v) be a path between two good components. Consider the cycle generated in T by adding (u, v). If there is a path consisting of link(s) of stress k in this cycle, a set of improvements which propagate to this path can be identified. Making

these changes reduces the size of S_k by at least one. Otherwise, there is at least one bad path consisting of link(s) of stress $k-1$ in the cycle. Form the union of all components in this cycle along with the bad paths in the cycle and mark all bad paths in this cycle as good. We then go back to look for other paths between good components. In all cases, the algorithm finds either a way to reduce the stress of some link in S_k or a blocking set with which Theorem 2 applies. Algorithm 1 implements the above idea and outputs a spanning tree whose stress is bounded by Theorem 2.

Algorithm 1: Approximation Algorithm for MSMT

INPUT: An underlay network G and an overlay path set G_o

OUTPUT: A spanning tree T of G_o which approximates a MSMT

Step 1: Find a spanning tree T of G_o. Let k be its stress.

Step 2: Mark links of stress k and $k-1$ as bad. Remove these links from T and generate a forest. Also remove these links from G and G_o. Mark all other links as good. Let F be the set of connected components in the forest.

Step 3: While there is a path (u, v) in G_o connecting two different components of F, and all links of stress k are marked as bad, do

1) Find the bad links/paths in the cycle Y generated by T together with (u, v), and mark them as good.
2) Update F by combining the components along the cycle Y and these newly marked paths into a single component. Note that more than two components of F may be combined into one in this step.

Step 4: If there is a link c of stress k marked as good, find a sequence of improvements which propagate to c and update T (and if necessary k) and go back to Step 2.

Step 5: Output the final tree T and its stress k.

We show an example in Fig. 2. As discussed, we suppose that the links of stress k only appear in (x_1, y_1), and that the links of stress $k-1$ only appear in (x_2, y_2) and (x_3, y_3). We mark these links in S_{k-1} as bad and remove them from T as well as from G and G_o. As a result, T is broken into a forest F consisting of four components, as shown in Fig. 2(b). Now we have a path (u_1, v_1) in the current G_o (after removing S_{k-1}) connecting two components of F. Clearly, the maximum stress of links in (u_1, v_1) is at most $k-2$ (if there exists a link in (u_1, v_1) which has appeared in the original tree T and whose stress is higher than $k-2$, this link should be in S_{k-1} and will be removed from G. Then (u_1, v_1) will be removed from G_o.) Let's add (u_1, v_1) into T, which connects component 1 and component 2 to form a new component. We further mark the bad links in (x_1, y_1) as good, and remove (x_1, y_1) from T, as shown in Fig. 2(c). Clearly, the stress of T now becomes $k-1$. We then re-mark all the links and repeat the above procedure. Suppose that we add path (u_2, v_2) into T and remove (x_2, y_2) from T in this phase, as shown in Fig. 2(d). Still, we need to re-mark the links and repeat the above procedure. However, this time after removing the links of stress $k-1$ and $k-2$, we cannot find paths in G_o connecting two components of the forest. Therefore, the algorithm stops and the current tree whose stress is $k-1$ is the approximation result of MSMT.

C. Performance Analysis

Lemma 1: When Algorithm 1 stops, $k \leq \min((|S|+|B|) \times \Delta^*(G, G_o), x \times \Delta^*(G, G_o) + |B|/(|S|+|B|))$, where S is S_k and B is the set of bad links of stress $k-1$ when the algorithm stops.

Proof: Since the algorithm only stops when there are no paths between the good components, the tree T along with these sets S and B satisfies the conditions of Theorem 1, and we get the desired result. ■

We can assume that the maximum number of links in an overlay path is no larger than a constant. This is a reasonable assumption in the Internet. Albert *et al.* have shown that the World-Wide Web forms a small-world network [37]. They estimate the WWW diameter in year 1999 to be around 19 hops, and expect that 1000% increase in the size of the web in the future will change the diameter from 19 to only 21. Based on this assumption, x in Theorem 2 and Lemma 1 is also no larger than a certain constant. We hence have the following result.

Corollary 1: When Algorithm 1 stops, $k \leq c \times \Delta^*(G, G_o)$, where c is a constant.

The corollary can be deduced from Lemma 1. Based on Lemma 1, we have $k \leq x \times \Delta^*(G, G_o) + |B|/(|S|+|B|) \leq x \times \Delta^*(G, G_o) + 1 \leq (x+1) \times \Delta^*(G, G_o) = c \times \Delta^*(G, G_o)$.

It shows that Algorithm 1 can achieve a constant approximation ratio for MSMT. We now analyze the computational complexity of the algorithm.

Lemma 2: Algorithm 1 runs in polynomial time.

Proof: Given a group of N hosts in the network (i.e., $|V_o| = N$), there are exactly $N-1$ overlay paths in a spanning tree. Since the number of underlay links in any path is no larger than a certain constant, the sum of stresses of all the links in the tree is $O(N)$. Hence, the number of links with stress k in the tree is at most $O(N/k)$. Since the size of S_k decreases at least by one in each phase (except the last one), there are at most $O(N/k)$ phases when the maximal stress is k. Summing up the harmonic series corresponding to different values of k, we conclude that there are $O(N \log N)$ phases.

In each phase, we try to find improvements which propagate to links of S_k. Lemma 1 shows that when the algorithm stops, the stress of the resulting tree is within a certain range of the optimal result. Each phase of the algorithm can be implemented in nearly linear time using Tarjan's fast disjoint set union-find algorithm for maintaining connected components [38]. Therefore, the entire algorithm runs in $O(MN\alpha(M, N) \log N)$ time, where M is the number of paths in G_o and α is the inverse of Ackermann's function that is associated with the union-find problem [39]. ■

IV. Maximum Bandwidth Multicast Tree (MBMT)

If link bandwidth information is available, a high-bandwidth overlay tree instead of a minimum-stress tree can be

constructed. In this section, we formulate the Maximum Bandwidth Multicast Tree problem and address it by extending the above approximation for MSMT.

Definition 5:

- An *underlay network* is a connected graph $G = (V, E, C, W)$, with V, E, C as defined in Definition 1, and $W : E \rightarrow R^+$ is the residual bandwidth on the edges.
- An overlay *path set* and a *multicast tree* are defined as in Definition 1.
- The *stress* of a link is defined as in Definition 2. The *tree bandwidth* of a multicast tree T is given by $B(T) = \min_{e \in E_s(T)}(W(e)/t(e,T))$.

Definition 6: Given an underlay network G and an overlay path set G_o, the *Maximum Bandwidth Multicast Tree (MBMT)* is a multicast tree of G_o with the maximum tree bandwidth.

Theorem 3: Given an underlay network G and an overlay path set G_o, it is NP-hard to compute the Maximum Bandwidth Multicast Tree of G_o. Furthermore, MBMT is not approximable within a factor of $(2/3 + \epsilon)$ for any $\epsilon > 0$, unless $P = \mathrm{NP}$.

The proof is trivial given that the MSMT problem is a special case of the MBMT problem.

We now discuss how to extend Algorithm 1 to approximate the MBMT problem. Given a spanning tree T, we repeatedly replace its bottleneck paths (i.e., those with the minimum transmission rate among all paths) with some other paths in G_o to improve the tree bandwidth. We sort all the links in $E_s(T)$ according to their transmission rates in an ascending order as

$$\frac{W(a)}{t(a,T)} \leq \frac{W(b)}{t(b,T)} \leq \cdots$$

where $(W(a)/t(a,T))$ is the tree bandwidth. Suppose that we delete a path p_a crossing link a from T and add another path p_n to form a new tree. To ensure the new tree achieves a higher tree bandwidth, we need to have

$$\frac{W(y)}{t(y,T)+1} > \frac{W(a)}{t(a,T)}, \text{for any link } y \text{ in } p_n.$$

Therefore, we propose Algorithm 2 to approximate the MBMT problem.

Algorithm 2: Approximation Algorithm for MBMT

INPUT: An underlay network G and an overlay path set G_o

OUTPUT: A spanning tree T of G_o which approximates a MBMT

Step 1: Find a spanning tree T of G_o.

Step 2: From the links in $E_s(T)$ with the minimum transmission rate, mark the one(s) with the smallest stress as bad. Remove these links from T and generate a forest. Also remove these links from G and G_o. Mark all other links as good. Let F be the set of connected components in the forest.

Step 3: For any link c in G, if $(W(c) - W(a)/t(a,T) \times (t(c,T)+1)) \leq 0$, with $(W(a)/t(a,T))$ being the tree bandwidth, remove c from G and G_o.

Step 4: If there is a path (u, v) in G_o connecting two different components of F, do

1) Find the bad links/paths in the cycle Y generated by T together with (u, v), and mark them as good.
2) Update F by combining the components along the cycle Y and these newly marked paths into a single component. Note that more than two components of F may be combined into one in this step.
3) Find a sequence of improvements which propagate to the newly marked links and update T.
4) Go back to Step 2.

Step 5: Output the final tree T.

V. Group Management Issues

In our system, there is a server serving the whole group and computing an overlay tree as described above. The server then distributes the tree information to all the hosts. As there may exist new incoming hosts after an overlay tree has been constructed, we discuss in this section how to deal with host joining and leaving. We further study tree resilience and system scalability issues.

A. Host Joining/Leaving

We use an add-on joining mechanism to deal with host joining. First of all, if the new host does not have enough underlay information, it should select some paths to measure (conducting traceroute or bandwidth measurement) using, for example, Max-Delta inference [13]. Based on the inferred topology, the server selects a parent for the new host as follows.

- *When MSMT is used:* The server checks the paths adjacent to the new host in the overlay path set. Each time, it adds one path to the current tree, and evaluates the stress of the new tree. This is easy given the underlay connectivity of the new tree. The server then repeats this procedure for a certain number of times and each time checks one path. The path that leads to the minimum tree stress if being added into the current tree is finally selected to connect the new host to the current tree.
- *When MBMT is used:* Similarly, each time the server randomly adds an adjacent path into the current tree and evaluates the tree bandwidth of the new tree. After repeating this procedure for a certain number of times, the server finally adds the path that leads to the maximum tree bandwidth of the new tree.

Upon failure or leaving of a host, all its children need to re-join the tree. The re-joining process is similar to joining. It is also possible to re-compute the whole tree using Algorithm 1 or 2 if there are a significant number of joining or re-joining hosts.

B. Discussion on Tree Resilience

Using a single tree may not offer satisfactory service, because, firstly, hosts in the system are heterogeneous with different incoming and outgoing bandwidth. A host's incoming path may not be able to provide enough bandwidth for streaming. Secondly, quality degradation at a host affects all its descendants. In a dynamic P2P system, it is difficult for hosts to achieve high streaming quality with a single tree. To address these problems, we can use multiple description coding (MDC) to encode streaming data into multiple descriptions and distribute the descriptions along multiple trees [25], [26].

In MDC, data are encoded into several descriptions. When all the descriptions are received, the original data can be reconstructed. If only a subset of the descriptions are received, the quality of the reconstruction degrades gracefully. The more descriptions a host receives, the lower distortion the reconstructed data would be. Therefore, the source can encode its media content into M descriptions using MDC (where M is a tunable parameter), and transmit the descriptions along M different trees. Note that a host has different descendants in different trees. The trees can be designed so that a descendant of a host in one tree is usually not the host's descendant in other trees. Therefore, packet loss or failure of a host only causes the loss of a single description (out of M descriptions) at each of its descendants. The system resilience is hence improved.

Another important issue in streaming is loss recovery. Although MDC and multiple-tree transmission can improve resilience, packets may still be lost due to background traffic or path/host failure. We can use lateral error recovery (LER) [40] (or many other solutions such as [41], [42]) for efficient loss recovery. LER randomly divides hosts into multiple planes and independently builds an overlay tree in each plane. Each host needs to identify some hosts from other planes as its recovery neighbors. Whenever an error occurs, the host performs retransmission from its recovery neighbors.

Note that mesh-based streaming has been proposed in the recent years [29], [30]. This approach builds a mesh among hosts using gossip algorithms, with hosts exchanging data with their neighbors in the mesh. It can achieve high resilience to network and group dynamics, as each host has multiple incoming paths for data delivery. However, mesh-based streaming has high control overhead due to data scheduling and mesh maintenance. It also has high end-to-end delay because in the gossip mesh a host may not always find close peers as their neighbors. On the contrary, trees introduce lower end-to-end delay and are easier to maintain. Therefore, to achieve both high delivery rate and low end-to-end delay, we can use a high-bandwidth overlay tree as the delivery backbone and enhance it with additional loss recovery mechanisms.

C. Discussion on a Distributed Approach

We have used a central server to compute and maintain the tree. When the number of hosts is large, the server may be overloaded and become the system bottleneck. A possible approach to improve system scalability is to use a cluster-based hierarchical structure for tree construction. An example of cluster-based structure is mOverlay [33]. In the scheme, hosts form a set of location-based clusters. Each cluster has a unique leader for cluster maintenance. Leaders periodically exchange information with their close counterparts to know about their neighbor clusters. Each cluster also selects some supernodes from the cluster members as the leader candidates. If the current leader leaves or fails, one of the candidates will become the new leader.

We can extend our tree construction method based on this hierarchical structure. After forming a set of clusters, each cluster can rely on its leader to construct an overlay tree spanning the cluster members as described above. We can then set up inter-cluster paths to connect these cluster-based trees. There are many ways to set up inter-cluster paths. A simple way is to require all the leaders to form an overlay tree. In another case, each leader can identify some ingress and egress hosts in its own cluster and ask these hosts to set up inter-cluster paths (e.g., [43]).

VI. Illustrative Numerical Results

In this section we present simulation results on both Internet-like topologies and a PlanetLab topology.

A. On Transit-Stub Topologies

We generate a number of (ten) *Transit Stub* topologies with GT-ITM [44]. The topology is a two-layer hierarchy of transit networks and stub networks. Each topology contains 5000 routers and about 30 000 links. Hosts are randomly put into the network. A host is connected to a stub router with 1 ms delay, while the delay of core links is given by the topology generator. Link bandwidth is set as follows: A backbone link (at least one end point is a transit router) can support 8 concurrent media streams, and a non-backbone link can support 2–4 concurrent media streams.

For our protocols, we use Max-Delta to infer the underlay topology and allow each host to traceroute at most $\lceil 2\log_2 N \rceil$ other hosts (N is the group size) [13]. We use the set of the traceroute paths as the overlay path set. We also implement three tree-based ALM protocols for comparison, i.e., Narada, Overcast, and TAG. The settings of these protocols follow those in [2], [24], [31], respectively.

We use *link stress*, *RDP* and *tree bandwidth* to evaluate a tree. Link stress and tree bandwidth are defined as in Section I. RDP is defined as the ratio of the overlay delay from the source to a given host to the delay along the shortest unicast path between them [2].

Fig. 3 compares the performance of the different protocols in terms of maximum and average link stresses. As shown in Fig. 3(a), Narada and TAG do not perform well, because they are optimized for delay and only consider tree bandwidth or link stress as the secondary metric. TAG is better than Narada due to its topology-awareness. Overcast is optimized for bandwidth and hence achieves lower maximum link stress. Our MSMT approximation algorithm performs the best. It reduces the maximum stress averagely by 66.3%, 61.9%, and 53.9% as compared to Narada, TAG and Overcast, respectively. Furthermore, in Fig. 3(b), the MSMT approximation algorithm reduces the average link stress by 35.1%, 26.5% and 21.1% as compared

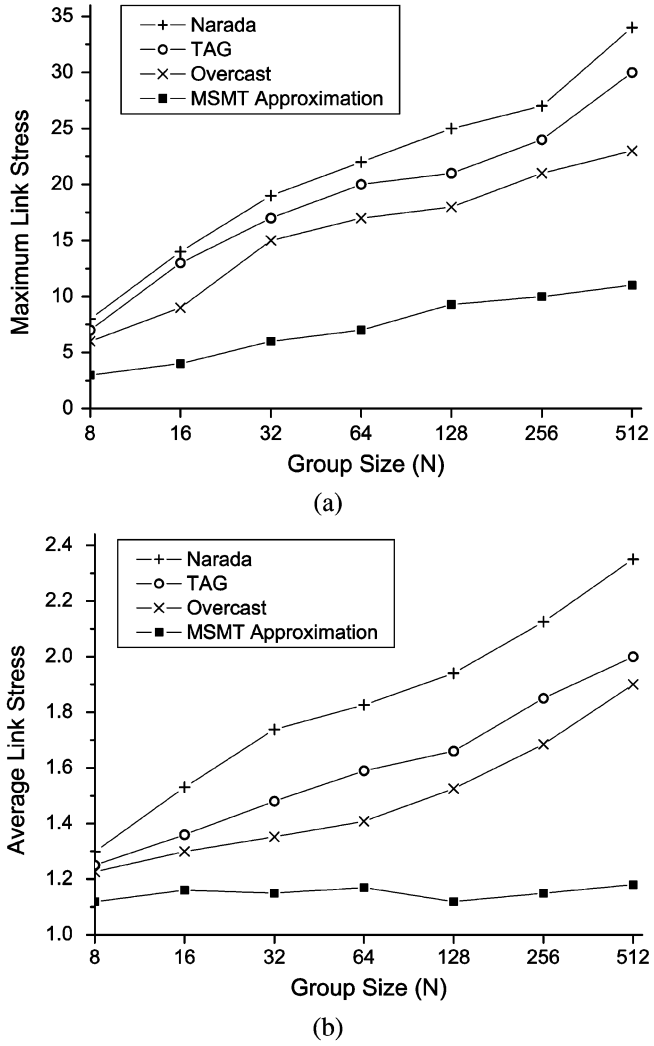

Fig. 3. Performance of the MSMT approximation algorithm (on Transit-Stub topologies). (a) Maximum link stress. (b) Average link stress.

to Narada, TAG and Overcast, respectively. The average link stress achieved by the MSMT approximation algorithm is close to the perfect result of 1 and is stable regardless of the group size. These results show that the MSMT approximation algorithm can efficiently reduce link stress.

As known, link stress is an important metric for streaming applications. For example, if we use Narada for streaming and serve a group of 512 hosts, in our simulation environment each underlay link averagely needs to deliver 2.35 streams and the most heavily loaded link has to deliver 34 streams. As a single stream usually requires several hundred Kbps transmission rate, such delivery load is significantly heavy for the current Internet. This is also the reason why mesh-based streaming has been proposed. Our MSMT approximation algorithm can significantly reduce link stress, both the maximum value and the average value. Hence, it can reduce bandwidth consumption and improve delivery efficiency.

We now evaluate the performance of the MBMT approximation algorithm. In our simulations, we assume that half of the hosts are in position before streaming starts and the other half join during streaming. Fig. 4 compares the performance of the protocols. Fig. 4(a) shows tree bandwidth versus group size for

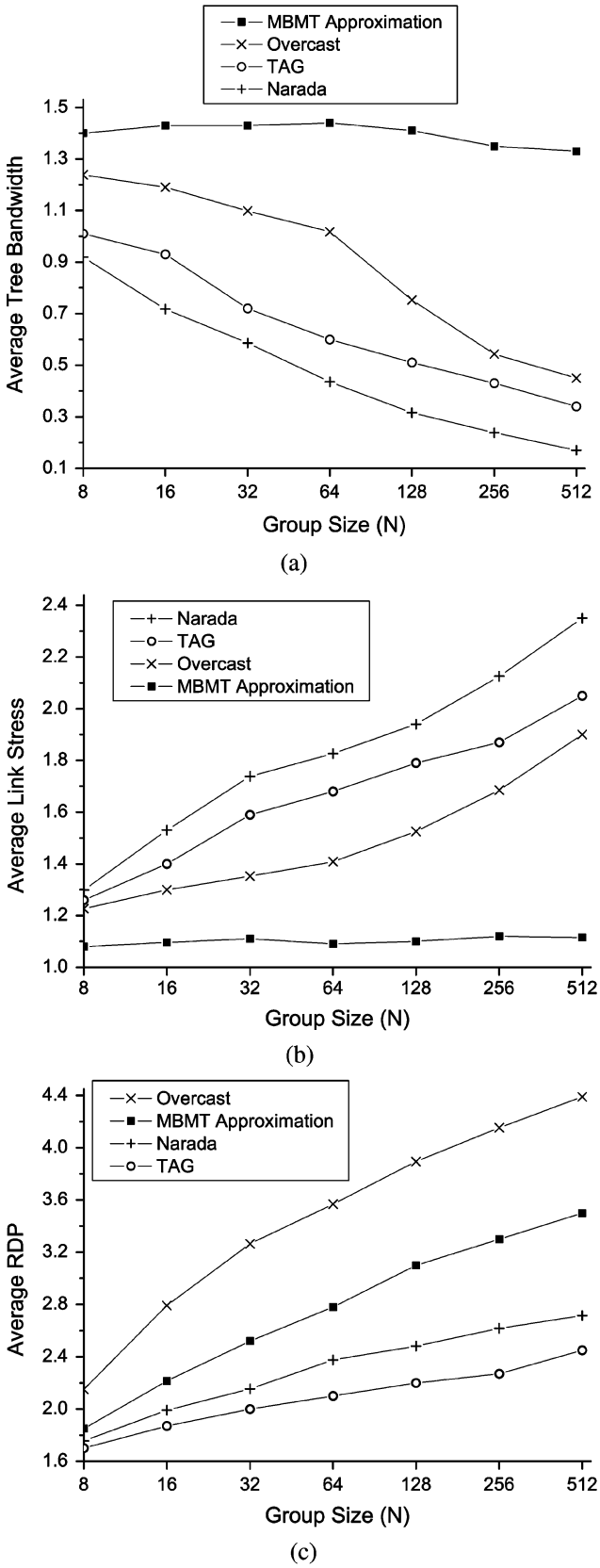

Fig. 4. Performance of the MBMT approximation algorithm (on Transit-Stub topologies). (a) Average tree bandwidth. (b) Average link stress. (c) Average RDP.

different protocols. As the group size increases, tree bandwidth decreases, for there are more contentions on underlay links. The MBMT approximation algorithm achieves much higher tree bandwidth than the other protocols. When the group size is relatively large (> 100), the MBMT approximation algorithm still achieves high tree bandwidth while Overcast does not perform so well. This is because Overcast adopts a greedy tree construction method and later joining hosts often cannot find good parents with enough forwarding bandwidth. TAG and Narada have the lowest tree bandwidth because they do not optimize bandwidth in tree construction. As shown, it is beneficial to infer underlay topologies when constructing high-bandwidth overlay trees.

Fig. 4(b) shows links stress versus group size. Clearly, the MBMT approximation algorithm achieves much lower stress than the other protocols, and its stress does not vary sensitively with the group size. It shows that the MBMT approximation algorithm has distributed delivery loads to different underlay links, thereby leading to high tree bandwidth [as shown in Fig. 4(a)]. Fig. 4(c) shows RDP of the protocols. Overcast has the highest RDP because it is not optimized for delay. Furthermore, it always inserts a new host as far from the source as the bandwidth constraint allows. The RDP of other protocols is lower. As compared to Narada and TAG, the MBMT approximation algorithm achieves high tree bandwidth while incurring small penalty in delay.

B. On Planetlab Topology

We have conducted traceroutes on PlanetLab to obtain a PlanetLab topology [45]. We randomly select a number of hosts from PlanetLab and conduct all-pairs traceroutes between them. Due to network and host dynamics (some hosts may unexpectedly fail during our measurements), a small portion of the traceroutes cannot be completed. From the traceroutes obtained, we construct a topology consisting of 72 hosts and 2540 overlay paths (we assume that paths are symmetric and only measure one path between a pair of hosts).

We use an alternative approach to estimate path bandwidth instead of Pathload. In the measurement, a sender sends a data clip to a receiver which measures the downloading time. The path bandwidth is computed as the data size divided by the downloading time. The packet size is set to 1 000 bytes and a data clip contains 1 000 packets. Each host measures its bandwidth to all the other hosts. Among the 72 hosts, we obtain the bandwidth results on 2189 paths (some transfer cannot be completed due to network and host dynamics). The maximum and minimum bandwidth is about 14.4 Mbps and 35 Kbps, respectively, and over 94% paths have bandwidth higher than 1 Mbps. We combine the traceroute results and bandwidth measurement results to build a testbed topology, with 72 hosts, 2176 overlay paths (the path set to be used in our protocol), 1145 underlay links, and 627 routers.

Based on the measured path bandwidth, we infer link bandwidth as follows. We first assign the bandwidth of all the links in a topology as 0. In the subsequent adjustment process, a link bandwidth can only be increased but not reduced. After conducting path bandwidth measurement between two hosts, say b Kbps, we check the bandwidth of all the underlay links in the path. If it is less than b, we increase it to b. Otherwise, its bandwidth remains unchanged. We show an example in Fig. 5. Fig. 5(a) is the actual underlay topology with link bandwidth as indicated. A, B, C are hosts, and 1, 2, 3, 4 are routers. We first measure the bandwidth of path $A - B$ as 3 (suppose that the path bandwidth measurement is accurate). We hence assign bandwidth 3 to links $A-1$, 1–2, 2–4, and $4-B$. After path bandwidth measurement from B to C, we update the bandwidth of links $B - 4$ and 4–2 to 5, and assign 5 to links 2–3 and $3 - C$. Finally, after measuring the bandwidth of path $C - A$, we assign bandwidth 2 to link 3–1, and leave the bandwidth of links $C - 3$ and $1 - A$ unchanged. Therefore, the inferred link bandwidth is as Fig. 5(b) shows. Clearly, this approach provides a reasonably accurate estimation on link bandwidth.

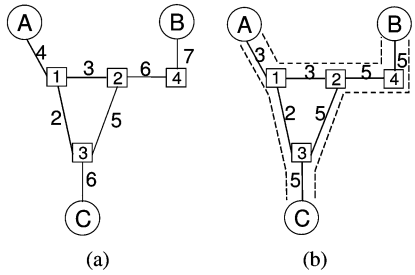

Fig. 5. Example of link bandwidth inference based on end-to-end path bandwidth measurement. (a) Actual underlay topology and link bandwidth. (b) Inferred link bandwidth.

We simulate the MBMT approximation algorithm, Overcast, Narada and TAG on this topology. In each simulation, we randomly select some hosts as group members (ranging from 5 to 65). Fig. 6 shows the results on this topology. In Fig. 6(a), the MBMT approximation algorithm always achieves higher tree bandwidth than the other protocols. This confirms the results on the Transit-Stub topologies. On average, the MBMT approximation algorithm can achieve 40% higher tree bandwidth than Overcast, 131% higher tree bandwidth than TAG and 267% higher tree bandwidth than Narada. It shows that Narada performs poorly in terms of tree bandwidth and is not applicable for streaming applications. On the contrary, Overcast performs much better and the MBMT approximation algorithm can further improve tree bandwidth. In fact, we expect larger performance gap between the MBMT approximation algorithm and Overcast when the group size increases, because the simulation results on the Transit-Stub topologies have shown that the tree bandwidth achieved by Overcast declines quickly with the increase of the group size.

Fig. 6(b) shows the stresses of the three protocols. As expected, the MBMT approximation algorithm achieves the lowest stress, while Narada achieves the highest. On average, the stress of MBMT is only 92% of that of Overcast, 86% of that of TAG and 81% of that of Narada. Fig. 6(c) shows the RDP values. Overcast has the highest RDP. The reason has been explained above. The other protocols have much lower RDP. The RDP of MBMT is slightly higher than that of Narada and TAG. On average, the RDP of MBMT is 73% of that of Overcast, 108% of that of Narada and 110% of that of TAG.

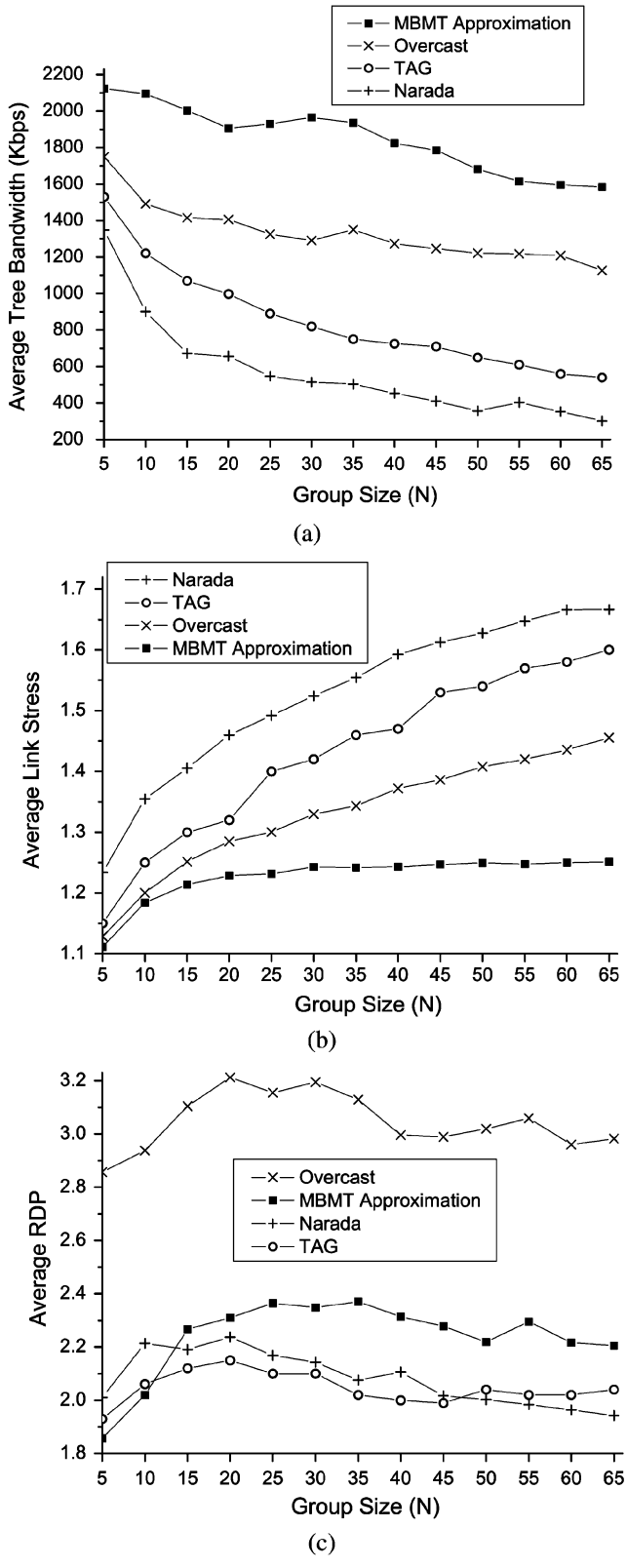

Fig. 6. Performance of the MBMT approximation algorithm (on the PlanetLab topology). (a) Average tree bandwidth. (b) Average link stress. (c) Average RDP.

In summary, the MBMT approximation algorithm achieves high tree bandwidth and low link stress, by introducing small penalty in RDP.

VII. Concluding Remarks and Future Work

In this paper, we study how to build a high-bandwidth overlay tree based on underlay information. We formulate two types of tree construction problems on a router-level topology, namely, Minimum Stress Multicast Tree if link bandwidth is not known and Maximum Bandwidth Multicast Tree if link bandwidth is known. We prove that both of them are NP-hard and are not approximable within a certain factor, unless $P = \mathrm{NP}$. We hence study approximation algorithms to address them and analyze the algorithm performance. We further investigate practical issues in system implementation, including group dynamics, tree resilience and system scalability.

We have conducted simulations on Internet-like topologies and a PlanetLab topology. The results on Internet-like topologies show that the average link stress achieved by our MSMT approximation algorithm is close to the perfect result of 1 and is stable regardless of the group size. The algorithm has reduced the average link stress by 35.1%, 26.5% and 21.1% as compared to Narada, TAG and Overcast, respectively. Furthermore, on the PlanetLab topology, the MBMT approximation algorithm can achieve 40% higher tree bandwidth than Overcast, 131% higher tree bandwidth than TAG and 267% higher tree bandwidth than Narada. These results show that our approximation algorithms can achieve high tree bandwidth, low link stress with small penalty in RDP. Our study shows that it is beneficial to infer the underlay topology before an overlay tree is constructed.

In this paper, we have assumed that link bandwidth is symmetric. Our algorithms are hence applicable to networks with symmetric bandwidth, e.g., Symmetric Digital Subscriber Line (SDSL), local area network, high-speed Ethernet or campus network. However, there are also a lot of asymmetric networks in the Internet, e.g., ADSL and cable networks. It will be our future work to extend our algorithms to asymmetric networks. Note that in the current Internet, the core networks are symmetric. Hence, we only need to consider asymmetry on the last-hop networks. A possible approach is to put some fanout constraints on hosts for the last-hop network. This problem is more practical and easier to address than considering a fully asymmetric network.

Appendix I
Proof of Theorem 1

Definition 7: Given a graph $G_x = (V_x, E_x)$, the *Minimum Degree Spanning Tree* is a spanning tree of G_x with the minimum maximal degree of the vertices.

Lemma 3: The Minimum Degree Spanning Tree problem is NP-hard and is not approximable within a factor of $(3/2 - \epsilon)$ for any $\epsilon > 0$, unless $P = \mathrm{NP}$ [46].

Lemma 4: The MSMT problem is polynomially transformable from the MDST problem.

Proof: Given a MDST instance $G = (V, E)$, where $V = \{v_1, v_2, \ldots, v_n\}$, we introduce a new vertex set $U = \{u_1, u_2, \ldots, u_n\}$ and a new edge set $F = \{(u_1, v_1), (u_2, v_2), \ldots, (u_n, v_n)\}$. The MSMT instance can be constructed as follows. Construct an *underlay network* $G_u = (V_u, E_u, C)$, where $V_u = V \cup U, E_u = E \cup F$ and for each edge $e \in E_u, C(e) = 1$. Construct an overlay *path*

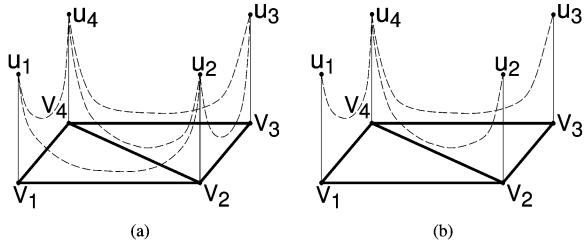

Fig. 7. Transformation from a MDST instance to a MSMT instance. (a) The original MDST instance G is indicated by the bold edges and their incident vertices. Underlay network G contains all the solid edges and the vertices. Overlay path set G_o contains the dashed edges with their incident vertices. Each dashed edge corresponds to a shortest path in G_u. For example, edge (u_4, u_2) corresponds the path $u_4 - v_4 - v_2 - u_2$. And G_o is isomorphic to G. (b) A spanning tree T of G_o is represented by the dashed edges with their incident vertices. $E_s(T)$ contains all the bold edges. The degree of u_4 in T is 3. Thus, edge (u_4, v_4) in G_u is shared by the 3 edges adjacent to u_4. The maximal stress of T is hence 3.

set $G_o = (U, E_o)$, where $E_o = \{(u_i, u_j) | (v_i, v_j) \in E\}$. From Definition 1, each edge $(u_i, u_j) \in E_o$ represents the path $u_i - v_i - v_j - u_j$ in G_u, since this path is the shortest path between u_i and u_j in G_u. Fig. 7 gives an example of this transformation. It is easy to see that G_o is isomorphic to the original graph $G = (V, E)$. Thus, we can consider both MSMT and MDST in graph G_o.

Given a spanning tree T of G_o, we transfer T into the underlay network G_u by replacing each edge $e \in T$ with the corresponding shortest path. Note that only edges in F can be shared by edges from T. For a vertex $u_i \in T$ with degree d, the edge (u_i, v_i) is shared by d edges adjacent to u_i in T and hence has a stress of d. Therefore, if the maximal degree of T is $d_{\max}$, the stress of T is $d_{\max}$ and vice versa. Thus, T is the Minimum Stress Multicast Tree iff T is the Minimum Degree Spanning Tree. Furthermore, this transformation can be done in polynomial time. ■

We now present the Proof of Theorem 1.

Proof of Theorem 1: Since the MDST problem is NP-hard and MSMT can be transformed from MDST in polynomial time, the MSMT problem is also NP-hard.

For the approximation part, suppose there is a polynomial algorithm which can always return a multicast tree with stress $s_{\text{appr}} \leq (2/3 + \epsilon) s_{\text{opt}}$ for any $\epsilon > 0$, where s_{opt} is the optimal stress. We apply it to the instance we constructed from MDST. Let $s_{\text{appr}} = d$ and $s_{\text{opt}} = d_{\min}$, where d is the maximal degree of the multicast tree, and $d_{\min}$ is the maximal degree of the optimal MDST. We can deduce that $d \leq (3/2 - 9\epsilon/(4 + 6\epsilon)) d_{\min}$. This contradicts Lemma 3, unless $P = \text{NP}$. ■

APPENDIX II
PROOF OF THEOREM II

Lemma 5: Suppose Q is a proper subset of the underlay edge set E of G. Let $|Q| = q$, and suppose the removal of Q from G disconnects G_o into t components. Then $\Delta^*(G, G_o) \geq \lceil (t-1)/q \rceil$.

Proof: Consider t components of the overlay network G_o. Any overlay path connecting these components needs to cross at least one underlay link in Q. To build a spanning tree over G_o, there are at least $t-1$ overlay paths between these components. Therefore, the average stress of links in Q in any spanning tree is at least $d = \lceil (t-1)/q \rceil$, and there is at least one link in Q whose stress is at least d. ■

We now present the Proof of Theorem 2.

Proof of Theorem 2: There are no overlay paths in G_o between components in $T - (S \cup B)$, and hence, the connected components of $T - (S \cup B)$ and $G_o - (S \cup B)$ are the same. If we can count the number of components in $T - (S \cup B)$, we can use Lemma 5 to obtain a lower bound on $\Delta^*(G, G_o)$.

The sum of stresses of links in $S \cup B$ is $|S| \times k + |B| \times (k-1)$. For each link e, the $t(e, T)$ overlay paths crossing it are distinct. Since an overlay path contains at most x different links of stress $k-1$ or k, the number of distinct overlay paths in T that cross at least one link in $S \cup B$ is at least $\max(k, \lceil (|S| \times k + |B| \times (k-1))/x \rceil)$. The forest F obtained from T by removing these paths has at least $\max(k, \lceil (|S| \times k + |B| \times (k-1))/x \rceil) + 1$ components. Therefore, an application of Lemma 5 yields

$$\begin{aligned}
\Delta^*(G, G_o) &\geq \left\lceil \frac{\max\left(k, \left\lceil \frac{|S| \times k + |B| \times (k-1)}{x} \right\rceil\right)}{|S| + |B|} \right\rceil \\
&= \max\left(\left\lceil \frac{k}{|S| + |B|} \right\rceil, \left\lceil \frac{\left\lceil \frac{|S| \times k + |B| \times (k-1)}{x} \right\rceil}{|S| + |B|} \right\rceil \right) \\
&\geq \max\left(\frac{k}{|S| + |B|}, \frac{k}{x} - \frac{|B|}{x(|S| + |B|)} \right).
\end{aligned}$$

As a result, we obtain that $k \leq \min((|S| + |B|) \times \Delta^*(G, G_o), x \times \Delta^*(G, G_o) + |B|/(|S| + |B|))$. ■

REFERENCES

[1] X. Jin, Y. Wang, and S.-H. G. Chan, "Fast overlay tree based on efficient end-to-end measurements," in *Proc. IEEE ICC'05*, May 2005, pp. 1319–1323.

[2] Y. H. Chu, S. Rao, S. Seshan, and H. Zhang, "A case for end system multicast," *IEEE J. Sel. Areas Commun.*, vol. 20, no. 8, pp. 1456–1471, Oct. 2002.

[3] S. Banerjee, B. Bhattacharjee, and C. Kommareddy, "Scalable application layer multicast," in *Proc. ACM SIGCOMM'02*, Aug. 2002, pp. 205–217.

[4] M. Coates, R. Castro, R. Nowak, M. Gadhiok, R. King, and Y. Tsang, "Maximum likelihood network topology identification from edge-based unicast measurements," in *Proc. ACM SIGMETRICS'02*, 2002, pp. 11–20.

[5] M. Coates, A. Hero, R. Nowak, and B. Yu, "Internet tomography," *IEEE Signal. Process. Mag.*, vol. 19, no. 3, pp. 47–65, May. 2002.

[6] D. G. Andersen, N. Feamster, S. Bauer, and H. Balakrishnan, "Topology inference from BGP routing dynamics," in *Proc. ACM SIGCOMM IMW'02*, Nov. 2002, pp. 243–248.

[7] F. Wang and L. Gao, "On inferring and characterizing Internet routing policies," in *Proc. ACM SIGCOMM IMC'03*, Oct. 2003, pp. 15–26.

[8] Traceroute [Online]. Available: http://www.traceroute.org/

[9] V. Jacobson, Pathchar [Online]. Available: http://www.caida.org/tools/utilities/others/pathchar/

[10] R. Govindan and H. Tangmunarunkit, "Heuristics for Internet map discovery," in *Proc. IEEE INFOCOM'00*, Mar. 2000, pp. 1371–1380.

[11] N. Spring, R. Mahajan, and D. Wetherall, "Measuring ISP topologies with Rocketfuel," in *Proc. ACM SIGCOMM'02*, Aug. 2002, pp. 133–145.

[12] B. Donnet, P. Raoult, T. Friedman, and M. Crovella, "Efficient algorithms for large-scale topology discovery," in *Proc. ACM SIGMETRICS'05*, Jun. 2005, pp. 327–338.

[13] X. Jin, W.-P. K. Yiu, S.-H. G. Chan, and Y. Wang, "Network topology inference based on end-to-end measurements," *IEEE J. Sel. Areas Commun.*, vol. 24, no. 12, pp. 2182–2195, Dec. 2006.

[14] B. Yao, R. Viswanathan, F. Chang, and D. G. Waddington, "Topology inference in the presence of anonymous routers," in *Proc. IEEE INFOCOM'03*, Apr. 2003, pp. 353–363.
[15] M. Jain and C. Dovrolis, "End-to-end available bandwidth: Measurement methodology, dynamics, and relation with TCP throughput," in *Proc. ACM SIGCOMM'02*, Aug. 2002, pp. 295–308.
[16] N. Hu and P. Steenkiste, "Evaluation and characterization of available bandwidth probing techniques," *IEEE J. Sel. Areas Commun.*, vol. 21, no. 6, pp. 879–894, Aug. 2003.
[17] V. Ribeiro, R. Riedi, R. Baraniuk, J. Navratil, and L. Cottrell, "pathChirp: Efficient available bandwidth estimation for network paths," in *Proc. PAM'03*, Apr. 2003.
[18] S. E. Deering, "Multicast routing in internetworks and extended LANs," in *Proc. ACM SIGCOMM CCR*, Aug. 1988, vol. 18, no. 4, pp. 55–64.
[19] S. Deering, D. L. Estrin, D. Farinacci, V. Jacobson, C.-G. Liu, and L. Wei, "The PIM architecture for wide-area multicast routing," *IEEE/ACM Trans. Netw.*, vol. 4, no. 2, pp. 153–162, Apr. 1996.
[20] S. E. Deering and D. R. Cheriton, "Multicast routing in datagram internetworks and extended LANs," *ACM Trans. Comput. Syst.*, vol. 8, no. 2, pp. 85–110, May. 1, 1990.
[21] J. Moy, Multicast Extensions to OSPF, Mar. 1994, RFC 1584.
[22] T. Ballardie, P. Francis, and J. Crowcroft, "Core based trees (CBT)—An architecture for scalable inter-domain multicast routing," in *Proc. ACM SIGCOMM'93*, Sep. 1993, pp. 85–95.
[23] C. A. S. Oliveira and P. M. Pardalos, "A survey of combinatorial optimization problems in multicast routing," *Comput. Oper. Res.*, vol. 32, no. 8, pp. 1953–1981, Aug. 2005.
[24] J. Jannotti, D. K. Gifford, K. L. Johnson, M. F. Kaashoek, and J. W. O'Toole, "Overcast: Reliable multicasting with an overlay network," in *Proc. OSDI'00*, Oct. 2000, pp. 197–212.
[25] M. Castro, P. Druschel, A.-M. Kermarrec, A. Nandi, A. Rowstron, and A. Singh, "SplitStream: High-Bandwidth multicast in cooperative environments," in *Proc. ACM SOSP'03*, Oct. 2003, pp. 298–313.
[26] D. Kostic, A. Rodriguez, J. Albrecht, and A. Vahdat, "Bullet: High bandwidth data dissemination using an overlay mesh," in *Proc. ACM SOSP'03*, Oct. 2003, pp. 282–297.
[27] Y. Guo, K. Suh, J. Kurose, and D. Towsley, "P2Cast: Peer-To-peer patching scheme for VoD service," in *Proc. WWW'03*, May 2003, pp. 301–309.
[28] T. Do, K. A. Hua, and M. Tantaoui, "P2VoD: Providing fault tolerant video-on-demand streaming in peer-to-peer environment," in *Proc. IEEE ICC'04*, Jun. 2004, pp. 1467–1472.
[29] X. Zhang, J. Liu, B. Li, and T.-S. P. Yum, "CoolStreaming/DONet: A data-driven overlay network for peer-to-peer live media streaming," in *Proc. IEEE INFOCOM'05*, Mar. 2005, pp. 2102–2111.
[30] Y. Tang, J.-G. Luo, Q. Zhang, M. Zhang, and S.-Q. Yang, "Deploying P2P networks for large-scale live video-streaming service," *IEEE Commun. Mag.*, vol. 45, no. 6, pp. 100–106, Jun. 2007.
[31] M. Kwon and S. Fahmy, "Topology-aware overlay networks for group communication," in *Proc. ACM NOSSDAV'02*, May 2002, pp. 127–136.
[32] M. Waldvogel and R. Rinaldi, "Efficient topology-aware overlay network," *ACM SIGCOMM CCR*, vol. 33, no. 1, pp. 101–106, Jan. 2003.
[33] X. Y. Zhang, Q. Zhang, Z. Zhang, G. Song, and W. Zhu, "A construction of locality-aware overlay network: mOverlay and its performance," *IEEE J. Select. Areas Commun.*, vol. 22, no. 1, pp. 18–28, Jan. 2004.
[34] R. Winter, T. Zahn, and J. Schiller, "Topology-aware overlay construction in dynamic networks," in *Proc. IEEE ICN'04*, Mar. 2004.
[35] Y. Cui, Y. Xue, and K. Nahrstedt, "Optimal resource allocation in overlay multicast," in *Proc. IEEE ICNP'03*, Nov. 2003, pp. 71–81.
[36] R. Cohen and G. Kaempfer, "A unicast-based approach for streaming multicast," in *Proc. IEEE INFOCOM'01*, Apr. 2001, pp. 440–448.
[37] R. Albert, H. Jeong, and A.-L. Barabasi, "Diameter of the world-wide web," *Nature*, vol. 401, pp. 130–131, Sep. 1999.
[38] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*. Cambridge, MA: MIT Press, 2001.
[39] D. S. Hochbaum, Ed., *Approximation Algorithms For NP-Hard Problems*. New York: PWS, 1997.
[40] W.-P. Yiu, K.-F. Wong, S.-H. Chan, W.-C. Wong, Q. Zhang, W.-W. Zhu, and Y.-Q. Zhang, "Lateral error recovery for media streaming in application-level multicast," *IEEE Trans. Multimedia*, vol. 8, no. 2, pp. 219–232, Apr. 2006.
[41] S. Banerjee, S. Lee, B. Bhattacharjee, and A. Srinivasan, "Resilient multicast using overlays," *IEEE/ACM Trans. Netw.*, vol. 14, no. 2, pp. 237–248, Apr. 2006.
[42] Z. Fei and M. Yang, "A proactive tree recovery mechanism for resilient overlay multicast," *IEEE/ACM Trans. Netw.*, vol. 15, no. 1, pp. 173–186, Feb. 2007.
[43] K.-W. Cheuk, S.-H. Chan, and J. Lee, "Island multicast: The combination of IP multicast with application-level multicast," in *Proc. IEEE ICC'04*, Jun. 2004, pp. 1441–1445.
[44] E. Zegura, K. Calvert, and S. Bhattacharjee, "How to model an internetwork," in *Proc. IEEE INFOCOM'96*, Mar. 1996, pp. 594–602.
[45] PlanetLab [Online]. Available: http://www.planet-lab.org
[46] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. New York: Freeman, 1979.

Xing Jin (S'04) received the B.Eng. degree in computer science and technology from Tsinghua University, Beijing, China, in 2002. He is currently pursuing the Ph.D. degree in the Department of Computer Science and Engineering, Hong Kong University of Science and Technology, Kowloon.

His research interests include overlay multicast with applications and QoS issues, Internet topology inference, end-to-end measurements, and peer-to-peer streaming. He has been a junior editor of the *Journal of Multimedia* since 2006.

Mr. Jin was awarded the Microsoft Research Fellowship in 2005.

W.-P. Ken Yiu (S'03) received the B.Eng. and M.Phil. degrees in computer science from the Hong Kong University of Science and Technology (HKUST), Kowloon, in 2002 and 2004, respectively. He is currently pursuing the Ph.D. degree in the Department of Computer Science and Engineering, HKUST.

His research interests include computer networks, peer-to-peer systems, multimedia networking, and network security.

Mr. Yiu was awarded the Academic Achievement Medal from HKUST in 2002, and the Sir Edward Youde Memorial Fellowship from Sir Edward Youde Memorial Fund in 2005 and 2006.

S.-H. Gary Chan (S'89–M'98–SM'03) received the B.S.E. degree (Highest Honor) in electrical engineering from Princeton University, Princeton, NJ, in 1993, with certificates in applied and computational mathematics, engineering physics, and engineering and management systems, and the M.S.E. and Ph.D. degrees in electrical engineering from Stanford University, Stanford, CA, in 1994 and 1999, respectively, with a minor in business administration.

He is currently an Associate Professor with the Department of Computer Science and Engineering, Hong Kong University of Science and Technology, Kowloon, and an Adjunct Researcher with Microsoft Research Asia, Beijing, China. He was a Visiting Assistant Professor in Networking with the Department of Computer Science, University of California, Davis, from 1998 to 1999. During 1992-1993, he was a Research Intern at the NEC Research Institute, Princeton, NJ. His research interests include multimedia networking, peer-to-peer technologies, and streaming and wireless communication networks.

Dr. Chan is a member of Tau Beta Pi, Sigma Xi, and Phi Beta Kappa. He was a William and Leila Fellow at Stanford University during 1993-1994. At Princeton University, he was the recipient of the Charles Ira Young Memorial Tablet and Medal, and the POEM Newport Award of Excellence in 1993. He served as a Vice-Chair of IEEE COMSOC Multimedia Communications Technical Committee (MMTC) from 2003 to 2006. He is a Guest Editor for the *IEEE Communication Magazine* (Special Issues on Peer-to-Peer Multimedia Streaming), 2007, and *Multimedia Tools and Applications* (Special Issue on Advances in Consumer Communications and Networking), 2006. He is Co-Chair of the Multimedia Symposium for IEEE ICC (2007). He was the Co-Chair for the workshop on Advances in Peer-to-Peer Multimedia Streaming for the ACM Multimedia Conference (2005), and the Multimedia Symposia for IEEE GLOBECOM (2006) and IEEE ICC (2005).

Yajun Wang received the B.Eng. degree in computer science from the University of Science and Technology of China, Hefei, in 2002. He is currently pursuing the Ph.D. degree in the Department of Computer Science and Engineering, Hong Kong University of Science and Technology, Kowloon.

His research interests include computational geometry, combinatorics, algorithms, and data structures.