Adaptive Multicast Tree Construction for Elastic Data Streams

Ying Zhu Ken Q. Pu
University of Ontario Institute of Technology
2000 Simcoe Street North
Oshawa, ON, Canada, L1H 7K4
Telephone: 905-721-8668
Email: ying.zhu@uoit.ca, ken.pu@uoit.ca

***Abstract*— In this paper, we revisit the problem of multicast tree construction in overlay peer-to-peer networks. We present an iterative online multicast tree construction algorithm for multi-session multicasting of elastic content. Our framework allows receivers to request multiple sessions from possibly different servers. Each receiver expresses a quality-of-service requirement using a utility function. We propose an online algorithm which constructs multicast trees to maximize the overall receiver utility functions in an iterative fashion, allowing it to adapt to changes in network topology and cross-traffic.**

I. Introduction

Multicast tree construction is a well-studied problem. Peer-to-peer (P2P) networks heavily rely on efficient construction of multicast trees to disseminate data. In this paper, we focus on the problem of dissemination of *elastic* data. This means that, when necessary, a data stream can be downsampled. Types of elastic data include environmental sensor data, stock quotes, audio and video feeds, etc. The possibility of downsampling creates unique opportunities to better utilize the underlying network. We assume a very general model of multicasting: multiple receivers request different data streams from multiple servers. The multicast problem, therefore, involves multiple sessions between multiple servers and clients. We further provide the flexibility for each client to express their demands on the quality of service (QoS) using a utility function on the received rate.

II. Related Work

Multicast tree has been applied to applications ranging from P2P file sharing [6], [7] to multimedia streaming [10], [5], [4], [3], to distributed conferencing applications [2], [1]. An important feature in multimedia streaming in P2P networks is that the data stream can be downsampled from a higher transmission rate to a lower transmission rate by packet drop policies. It has been shown by Chu et al [2] that the ability to drop packets can greatly improve the overall multicast session. Dán et al [4] studied another issue with multicast tree construction, namely the node dynamics and transmission rate variation. Their solution was to apply coding to overcome node dynamics. Another difficulty in multicasting is the issue of cross-traffic where different multicast sessions interfere with one another due to commonly shared physical network. Tsang et al [9] applies a receiver-coodinated approach to overcome the issue of cross-traffic in multicasting using multicast trees.

In this paper, we adopt a self-adaptive approach to the problem of multi-session multicasting of elastic data streams. In comparison with previous work, we allow receivers to express their QoS specification through utility functions. After an initial multicast tree construction phase, we iteratively make low-cost adjustments to constructed multicast trees in order to maximize the overall utility. The coodinated incremental adjustments allow our system to: (1) adapt to node dynamics, (2) avoid bottlenecks created by cross-traffic, and (3) maintain fairness among multiple multicast sessions.

III. Problem Formulation

A. Multicasting of Elastic Data

We treat the peer-to-peer (P2P) network as a directed graph $G = (V, E)$ where each edge $e \in E$ is labeled by its maximal bandwidth capacity, $\mathrm{capacity}(e) \in \mathbb{R}^+$. We assume that data streams originate from a subset of nodes in the network, and these data streams are to be disseminated to interested receiver nodes. Let $S(v)$ be the set of streams which are generated by the node v, and $S = \bigcup_{v \in V} S(v)$ be the set of all available streams. A multicast session is characterized by a source node $v_s \in V$, a set of receiver nodes $V_R \subseteq V$ and a stream $s \in S(v_s)$. We may write a multicast session as tuple: $M = (v_s, V_R, s)$. Without loss of generality, we assume that the source node and the stream (v_s, s) uniquely identifies a multicast session.

A multicast tree of a multicast session (v_s, V_R, s) is a tree $\tau = (V_\tau, E_\tau)$ where $\textsc{Root}(\tau) = v_s$ and $V_R \subseteq V_\tau$. We denote the children of a node $v \in V_\tau$ as $\downarrow_\tau(v)$, and the parent as $\uparrow_\tau(v)$. Since τ is a tree, each node v (except the root) has a unique in-edge, written $\textsc{In-Edge}(v)$. The set of out-edges of v is written $\textsc{Out-Edges}(v)$. We further define $\uparrow(e) = \textsc{In-Edge}(\textsc{Source}(e))$ and $\downarrow(e) = \textsc{Out-Edges}(\textsc{Target}(e))$.

Each edge $e \in V_\tau$ in the tree is labeled by a flow rate $\mathrm{rate}_\tau(e) > 0$ such that

$$\forall v \in V_\tau, e \in \textsc{Out-Edges}(v),\ \mathrm{rate}_\tau(\textsc{In-Edge}(v)) \geq \mathrm{rate}_\tau(e)$$

We assume that the data to be multicasted are *elastic* in that it can be downsampled to a lower rate if necessary. Elastic data streams are commonly found in many applications of networking such as Voice-over-IP, streaming video in which the quality of multimedia content can be reduced to accomodate lack of bandwidth. In sensor networks, sensor streams

can be downsampled by dropping uninteresting sensor readings. In approximate stream query processing, such as stock quote analysis or network traffic analysis, one may choose to randomly downsample an incoming data stream if the query answer can be estimated from the downsampled stream. The trade-off of downsampling is that only an approximation of the original data stream is received. We assume that each receiver $v_R \in V_R$ will express its requirement of the rate of reception. In Section III-C, we will explain in greater detail how each receiver can express its quality requirements on the received stream.

B. Traffic Control

In general, there are multiple multicast sessions sharing the network resource. A well-known advantage of P2P networks is that peer nodes have more capabilities than simple network routers. In our context, peer nodes also perform the following tasks:

- A node maintains the set of multicast sessions for each outgoing network link in the P2P network.
- A node administrates the bandwidth allocation of all the outgoing network links among their respective multicast sessions.
- A node performs any duplication and downsampling of data when multicasting to its outgoing links.

C. Receiver Utility

Different receiving nodes may participate in the same multicast session with very different purposes for the requested stream. For example, consider a data stream of network traffic. Receiver v_1 may request the network traffic data for the sake of plotting, while receiver v_2 may request the traffic data for computing a daily-moving average, and yet a third receiver v_3 may request the traffic data for anomaly detection. We propose to use *utility functions* $\{U_v : v \in V_R\}$ which map reception rates to measure of preference. Figure 1 shows the utility functions of the three receivers. They reflect the following performance requirements:

- Receiver v_1 *prefers* to receive the data stream up to the rate of 5, and gives no preference to higher rates. Presumably, with rate = 5, the network plot reaches the display resolution.
- Receiver v_2 *requires* the rate to be at least 2, but gives no preference to higher rates.
- Receiver v_3 is *greedy* as it always prefers higher reception rates.

An important property of the receiver utility functions is that they are always monotonic.

Given a multicast tree τ, the flow rate received by a receiver node v is simply

$$\mathrm{rate}_\tau(v) = \min\{\mathrm{rate}(e) : e \in \textsc{Path}_\tau(\textsc{Root}(\tau), v)\}$$

D. The Multicast Tree Construction

Given a collection of multicast sessions, we wish to construct the respective multicast trees in order to maximally satisfy each receiver according to their utility functions. At the same time we also wish to minimize the network load incurred by the multicast trees.

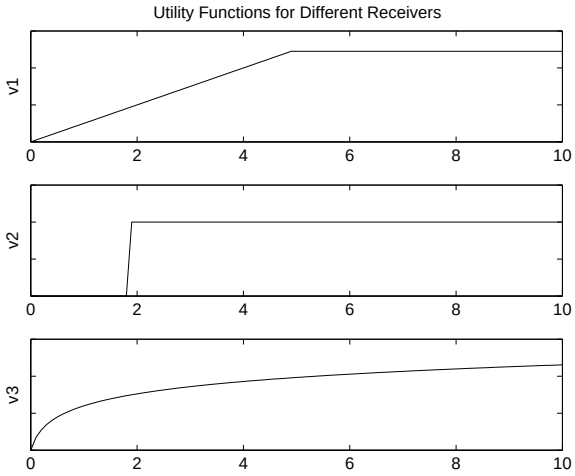

Fig. 1. Different Receiver Utility Functions

Formally, given some multicast sessions $\{M_1, M_2, \ldots, M_k\}$, we wish to construct the multicast trees $\{\tau_1, \tau_2, \ldots, \tau_k\}$ with the following objectives:

(i) The trees respect the total bandwidth capacity:

$$\forall e \in E, \sum_{i \le k} \mathrm{rate}_{\tau_i}(e) \le \mathrm{capacity}(e)$$

(ii) Maximizes the receivers' utilities:
The quality of a multicast tree τ is simply given by the total utility of all of its receiving nodes:

$$U(\tau) = \sum_{v \in V_R} U_v(\mathrm{rate}_\tau(v))$$

(iii) Minimize the network load:
The network load of a multicast tree is given by:

$$L(\tau) = \sum_{e \in E_\tau} \mathrm{rate}_\tau(e)$$

We wish to minimize the total network load $\sum_{i \le k} L(\tau_i)$ without degradation of the receiver utilities significantly.

IV. Algorithms

In this section, we present our iterative multicast tree construction algorithm.

A. Path Construction

Before addressing the problem of multicast tree construction, we first study a relatively simple problem of path construction. Given two nodes: a source v_0 and a target v_1 in the P2P network G, and a utility function U for the target node v_1, we wish to construct a path in G from v_0 to v_1 such that (i) the utility of v_1 is maximized according to U, and (ii) minimal network load is incurred.

Formally, a path $p(v_0, v_1)$ consists a sequence of edges $\langle e_1, e_2, \ldots, e_n \rangle$ and a function rate_p mapping edges e_i to $\mathrm{rate}_p(e_i) > 0$ such that $\textsc{Source}(e_1) = v_0$ and $\textsc{Target}(e_n) = v_1$, $\textsc{Source}(e_i) = \textsc{Target}(e_{i-1})$ for all $i > 1$, and finally $\mathrm{rate}_p(e_i) \ge \mathrm{rate}_p(e_{i-1})$.

The utility of the path p is defined as $U(p) = U(\mathrm{rate}_p(\textsc{In-Edge}_p(v_1)))$, while the network load of the path is given by $L(p) = \sum_{e \in p} \mathrm{rate}_p(e)$.

1) Multi-objective Path Optimization by Lexicographical Ordering: The first way to compare the quality of two paths p_1, p_2 which have the same source and target nodes is by lexicographical ordering of their utility U and L. Let $p_1 > p_2$ be defined as when p_1 is *better* than p_2. The ordering is formally given by,

$$\begin{aligned} U(p_1) > U(p_2) &\implies p_1 > p_2 \\ U(p_1) < U(p_2) &\implies p_1 < p_2 \\ U(p_1) = U(p_2) &\implies p_1 > p_2 \iff L(p_1) < L(p_2) \end{aligned}$$

In order to compute the optimal path using the classical path finding algorithms such as dynamic programming or the Dijkstra's algorithm, we must define utility functions for intermediate nodes in a path, so that one may assign utility values to sub-paths of a path. We simply define the utility function of an intermediate node in a path to be the utility function of its receiver.

Definition 1 (Utility value of sub-paths): Let p be a path from v_0 to v_1, where the receiver node v_1 has the utility function U_1. Then, every intermediate node $v' \in p$ inherits the utility U_1, i.e., $U_{v'} = U_1$.

An immediate consequence is the following.

Proposition 1: Let p' be a sub-path of p. Then $U(p') = U(p)$.

Proposition 2: Suppose that a path $p = \langle p_0, e \rangle$ is optimal, then p_0 is also optimal.

Proposition 3: Dijkstra's algorithm works correctly for computing optimal path according to the multi-objective lexicogrpahical ordering.

Multi-objective lexicographical order has the shortcoming of being overly favorable to the utility of the receiver node. It is possible that the optimal path between two nodes u, v may improve the utility $U(p)$ only slightly compared to the second best path, but is significantly worse in terms of its network load $L(p)$.

EXAMPLE:: Consider a P2P network with only 4 nodes $V = \{\mathsf{a}, \mathsf{b}, \mathsf{c}, \mathsf{d}\}$. It is shown in Figure 2. Consider paths from node a to d. The optimal path by the multi-objective lexicographical ordering is given by $p_1 = \langle \mathsf{a}, \mathsf{b}, \mathsf{c}, \mathsf{d} \rangle$, with $\mathrm{rate}_{p_1}(d) = 10$. However, the network load is high: $L(p_1) = 30$. Yet, a more reasonable path connecting node a to node d is the direct connection $p_2 = (\mathsf{a}, \mathsf{d})$ which has a slight degradation in throughput, $\mathrm{rate}_{p_2}(d) = 9.9$, but a much lower network load $L(p_2) = 9.9$.

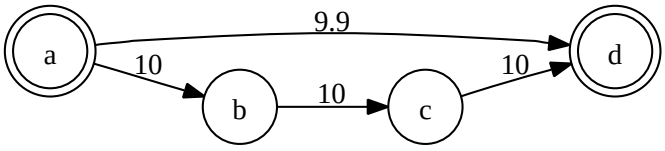

Fig. 2. A simple network

2) Multi-objective Path Optimization by Scoring Function: In order to control the trade-off between the objectives of maximization of receiver utility and minimization of the network load, we introduce a scoring function $f : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ which maps the receiver utility and network load into a single score.

Definition 2 (Scoring Function and Path Score): Given a function $f : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$, it is a valid scoring function if it is monotonically increasing w.r.t. its first argument (receiver utility), and monotically decreasing w.r.t. its second argument (network load). Given a path p and a scoring function f, the path score of p is given by $f(p) = f(U(p), L(p))$. A scoring function $f(x, y)$ is called *separable* if it can be written in the form of $g(x) \cdot h(y)$.

For the rest of this paper, we assume the following simple separable scoring function: $f(x, y) = x \cdot e^{-\alpha y}$, for some constant $\alpha > 0$.

Path optimization by scoring function is superior compared to path optimization by lexicographical ordering. The reason is that, by designing the scoring function, one can control the preference between maximizing receiver utility and minimizing network resource.

The following proposition is an immediate consequence of the monotonicity and separability of the scoring function.

Proposition 4: Let $p = \langle p_0, e \rangle$ be an optimal path according to a separable scoring function f, then the prefix subpath p_0 is also optimal with respect to f.

Proposition 5: Dijkstra's algorithm works correctly for computing optimal path according to any separable scoring function.

The algorithm OPTIMAL-PATH is a slight variation on the standard Dijkstra's algorithm. It extends the standard shortest path problem by

- Evaluating paths using the scoring function on the flows (rather than distance).
- Incorporating a maximum in-flow to the source.

The pseudo-code is given in Algorithm 1 below.

We invoke a function GET-CAPACITY(G, u, v) which returns the available capacity of the edge between u and v in the network G. For a single-session the available capacity is simply the bandwidth offered by the substrate network. However, in later sections, when we deal with multi-session multicast tree constructions, GET-CAPACITY(G, u, v) will be computed dynamically by the P2P network management layer.

B. Single-Session Multicast Tree Construction

We now describe how to construct a single-session multicast-tree. The pseudo-code for the algorithm BUILD-MC-TREE is given below in Algorithm 2.

Definition 3 (Multicast Sessions): A session s in a P2P network G is defined by

- A source SOURCE$(s) \in V_G$.
- A set of receivers RECEIVERS$(s) \subseteq V_G$.
- Receiver utility functions: $U_{t,s} : \mathbf{R} \to \mathbf{R}$ for each receiver node $t \in$ RECEIVERS(s).

Proposition 6: Optimal tree construction is NP-complete.

Proof: It is easy to see that Steiner tree can be reduced to our problem. ■

Proposition 7: For every iteration of the loop (Line 3 of Algorithm BUILD-MC-TREE), the path p^* computed in Line 6 is always such that $\tau \cup p^*$ is cycle-free, thus remains a tree.

Algorithm 1 OPTIMAL-PATH$(G, t_0, t_1, r_{\text{in}}, U)$

Require: G is the network
t_0, t_1 are the source and target resp.
r_{in} is the in-flow rate to the source t_0
U is the utility function of the target
{We assume that $f(x, y)$ is the scoring function}

1: $S \leftarrow \emptyset$
2: flow$[t_0] = r_{\text{in}}$
3: load$[t_0] = 0$
4: $Q \leftarrow V_G$ {add nodes to queue}
5: **while** $Q \neq \emptyset$ **do**
6: $u \leftarrow$ DEQUEUE(Q)
7: $S \leftarrow S \cup \{u\}$
8: **for** $v \in$ ADJACENT-NODES(u) **do**
9: { Relax(u, v) }
10: $s = f(U(\text{flow}(v)), \text{load}(v))$
11: flow$' = \min(\text{flow}(u),$ GET-CAPACITY$(G, u, v))$
12: load$' = \text{load}(u) + \text{flow}'$
13: $s' = f(U(\text{flow}'), \text{load}')$
14: **if** $s' > s$ **then**
15: predecessor$[v] = u$
16: flow$[v] = \text{flow}'$
17: load$[v] = \text{load}'$
18: **end if**
19: **end for**
20: **end while**

Algorithm 2 BUILD-MC-TREE(G, s)

Require: G is the P2P network
s is a session in G

1: $R =$ RECEIVERS(s)
2: $\tau =$ EMPTY-TREE(s)
3: **while** $R \neq \emptyset$ **do**
4: Choose t from R
5: **if** $t \in \tau$ **then**
6: DETACH$(G, t,$ RECEIVERS$(s))$
7: **end if**
8: $v^* = \text{argmax}\{\|$OPTIMAL-PATH$(G, v, \text{rate}_\tau(v), t)\| : v \in \tau\}$
9: $p^* =$ OPTIMAL-PATH$(G, v^*, \text{rate}_\tau(v), t)$
10: $R = R - \{t\}$
11: $\tau = \tau \cup p^*$
12: **end while**
13: **return** τ

Algorithm 3 RE-BUILD-MC-TREE(G, s, τ)

Same as BUILD-MC-TREE*, except:*
delete Line 2 from BUILD-MC-TREE

C. Multi-Session Multicast Tree Construction

In the single-session case, the utility of each receiver can be maximized individually because the receivers do not compete for bandwidth. However in the multi-session case, multicast trees for different sessions compete for the network resource. In order to maintain fairness, we appoint the P2P nodes to track the set of sessions that utilitize its outgoing P2P network links, and maintain the network flow for each session.

In order to manage the bandwidth allocation to multiple sessions, each P2P downstream link maintains the following information:

$b(u, v)$: the total available bandwidth for the link (u, v)
$b(u, v, s)$: bandwidth allocation to session s for the link (u, v)
$r_{\text{available}}(G, u, v)$: available *unused* bandwidth for the link (u, v).

Path Construction for MSMC

We also rely on the modified Dijkstra's algorithm shown in Algorithm 1 to compute the connecting paths between nodes. The only concern is that the available bandwidth reported by GET-CAPACITY is specific to the multicast session, and is calculated by the P2P network management layer. We *modify* Algorithm 1 to use GET-SESSION-CAPACITY instead of GET-CAPACITY. The procedure GET-SESSION-CAPACITY manages bandwidth used by multiple sessions and performs fairness bandwidth allocation if the requested flow rate exceeds the physical capacity.

Algorithm 4 GET-SESSION-CAPACITY(G, u, v, s, r)

Require: The requested rate r is specified.

1: **if** $b(u, v, s) + r_{\text{available}}(u, v) < r$ **then**
2: **for** $s' \in$ SESSIONS(G, u, v) **do**
3: **if** $s' == s$ **then**
4: $c = r / b(u, v)$
5: **else**
6: $c = b(u, v, s') / b(u, v)$
7: **end if**
8: $b(u, v, s') \leftarrow c \cdot b(u, v, s')$
9: **end for**
10: **end if**
11: $\Delta r = \max(0, r - b(u, v, s))$
12: $b(u, v, s) \leftarrow b(u, v, s) + \Delta r$
13: **return** $b(u, v, s)$

Tree Construction for MSMC

The construction and maintenance of multi-session multicast (MSMC) trees rely on the construction of single session multicast trees using the algorithm BUILD-MC-TREE and the incremental tree adjustment algorithm of RE-BUILD-MC-TREE. It is shown in Algorithm 5.

V. EXPERIMENTS

We have evaluated our algorithms with simulated P2P networks. We generated random networks using the network generator Brite [8]. We generated three types of networks of different sizes and numbers of receivers.

Network size	Number of sources	Number of receivers
50	10	10
100	10	20
250	10	50

Algorithm 5 BUILDMSMCTREE(G, S)

Require: G is a P2P network
S is a set of sessions

1: **for** $s \in S$ **do**
2: $\tau_s \leftarrow$ EMPTY-TREE(S)
3: **end for**
4: **while** *true* **do**
5: **for** $s \in S$ **do**
6: $\tau_s =$ RE-BUILD-MC-TREE(G, s)
7: **end for**
8: **end while**

For each network, we started the same number of sessions of 15.

Figure 3 shows the improvements per iteration and the total score of five randomly generated multicast sessions. Observe that in all cases, the iterative algorithm converges within 50 iterations.

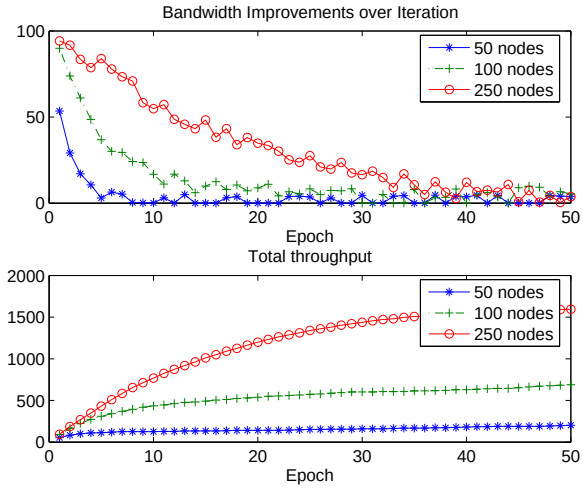

Fig. 3. Iterative Convergence

In order to evaluate the adaptiveness of our algorithm, for the large network of 250 P2P nodes and 50 receivers, we introduced a catastrophic network failure by deleting randomly selected links in the network at time $t = 500$. The total throughput is shown in Figure 4. Observe the iterative multicast tree construction algorithm re-optimizes the multicast paths and was able to recover from the failure. The reason that the recovered throughput is lowered is because the network after the failure is significantly more sparse, thus all the sessions are forced to share more of the physical available bandwidth.

VI. CONCLUSION

We have presented an iterative multicast tree construction algorithm for multi-session multicasting of elastic data streams. Our algorithm allows the receivers to express their QoS requirements as utility functions. Multi-session multicast trees with downsampling are constructed by maximizing the overall receiver utilities while minimizing network usage. In order to adapt to cross-traffic and network dynamics, the multicast trees are iteratively re-optimized. As the experimental data indicates, the algorithm is fast converging and is robust to changes in the P2P network.

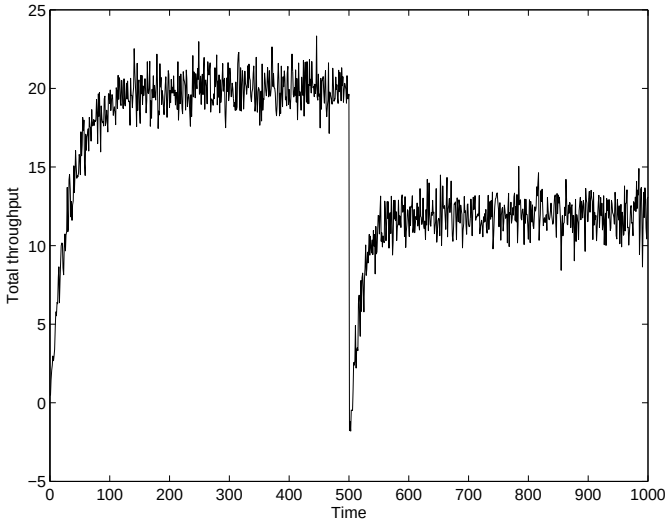

Fig. 4. Recovery from node failures

REFERENCES

[1] M. Castro, P. Druschel, A.-M. Kermarrec, A. R. A. Nandi, and A. Singh. Splitstream: High-bandwidth multicast in cooperative environments. In *Proc. of 18th ACM Symposium on Operating Systems Principles*, pages 298–313, 2003.

[2] Y. H. Chu, S. G. Rao, and H. Zhang. Enabling conferencing applications on the internet using an overlay multicast architecture. In *Proc. of SIGCOMM 01*, pages 55–67, 2001.

[3] Liang Dai, Yi Cui, and Yuan Xue. On scalability of proximity-aware peer-to-peer streaming. In *Proc. of IEEE INFOCOM 2007*, 2007.

[4] Gyorgy Dan, Viktoria Fodor, and Ilias Chatzidrossos. On the performance of multiple-tree-based peer-to-peer live streaming. In *Proc. of IEEE INFOCOM 2007*, pages 2556–2560, 2007.

[5] Xiaohui Gu, Zhen Wen, and Philip S. Y. Bridgenet: An adaptive multi-source stream dissemination overlay network. In *Proc. of IEEE INFOCOM 2007*, pages 2586–2590, 2007.

[6] A. Klemm, C. Lindemann, and O.P. Waldhorst. A special-purpose peer-to-peer file sharing system for mobile ad hoc networks. *Vehicular Technology Conference, 2003. VTC 2003-Fall. 2003 IEEE 58th*, 4:2758–2763 Vol.4, 6-9 Oct. 2003.

[7] D. Kostic, A. Rodriguez, J. R. Albrecht, and A. Vahdat. Bullet: High bandwidth data dissemination using an overlay mesh. In *Proc. of 18th ACM Symposium on Operating Systems Principles*, pages 282–297, 2003.

[8] Alberto Medina, Ibrahim Matta, and John Byers. Brite: A flexible generator of internet topologies. Technical report, Boston University, Boston, MA, USA, 2000.

[9] M. Tsang, C. Wang, K. Tsang, and F. Lau. A receiver-coordinated approach for throughput aggregation in high bandwidth multicast. In *Proc. of IEEE INFOCOM 2007*, pages 2551–2555, 2007.

[10] X. Zhang, J. Liu, B. Li, and P. T. S. Yum. Coolstreaming/donet: A data-driven overlay network for efficient live media streaming booktitle. In *Proc. of IEEE INFOCOM*, 2005.