This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE INFOCOM 2007 proceedings.

QoS-aware Streaming in Overlay Multicast Considering the Selfishness in Construction Action

Dan Li*, Jianping Wu*, Yong Cui*, and Jiangchuan Liu†

*Department of Computer Science, Tsinghua University, Beijing, China
Email: lidan@csnet1.cs.tsinghua.edu.cn, jianping@cernet.edu.cn, cy@csnet1.cs.tsinghua.edu.cn

†School of Computing Science, Simon Fraser University, British Columbia, Canada
Email:jcliu@cs.sfu.ca

***Abstract*— Most existing overlay multicast proposals have assumed that the nodes are cooperative and thus focus on the global topology optimization. However, a unique and important characteristic of overlay nodes is that, as application-layer agents, they can be selfish with their own interests. To achieve better Quality-of-Service (QoS) or to minimize forwarding overhead, an overlay node can behave selfishly in the information collection or in the overlay construction. While the former has recently been investigated, the impact of selfishness in the construction action remains unclear.**

In this paper, we present the first systematic study on the impact of selfishness in both tree and mesh overlay construction. Our investigation considers multiple QoS measures for streaming applications, including stream latency, resolution, and continuity. Our contribution is twofold: First, we analyze how for selfish overlay nodes to choose a construction-action policy to optimize their individual multi-metric QoS. Second, we demonstrate that the selfishness-aware policy for the construction action is consistent with the QoS optimization for the global multicast session, but not vice versa. The implication is significant: A globally optimal overlay construction itself can be vulnerable to individual selfishness; but, following our directions, we can design an overlay that is both globally optimal and selfish-resistant.

I. INTRODUCTION

Given the multi-receiver nature of video streaming applications, multicast is a natural vehicle to support such applications [1]. It is known that network-layer multicast, or IP multicast [2], is the most efficient, but its reach and scope remain very limited due to many practical and political reasons. Recently, application-layer overlay multicast [3∼13] has emerged as a promising alternative. Overlay multicast realizes routing and data transmission in the application-layer, which is much easier to implement and deploy, though less efficient [14][27]. Overlay multicast is also more flexible, because it is decoupled from the network-layer routing, and end systems in the application layer support much richer semantics.

Optimizing the overlay structure is clearly critical to the performance of these protocols. Existing proposals on overlay structures can be broadly classified into two categories [27], namely, *tree-based* and *mesh-based*. The former follows a well-ordered parent-child relation for data delivery; the latter, however, does not maintain such a fixed relation, but let each node keep a small set of partners to exchange their data availability information, and accordingly fetch expected data.

In both tree and mesh overlays, the structure establishment generally consists of two steps, that is, information collection and construction action. In the first step, overlay nodes learn the information of other nodes and the virtual overlay links, such as the outgoing bandwidth, pair-wise delay, link cost, and etc. In the construction action, based on the available information, each node selects a long-time parent to receive stream data in tree-based overlay multicast, or selects a segment-providing node to fetch a certain segment in mesh-based overlay multicast.

Most existing overlay multicast proposals have assumed that the nodes are cooperative and thus focus on the global topology optimization [15∼16]. However, a unique and important characteristic of overlay nodes is that, as application-layer agents, they can be selfish with their own interests. To achieve better Quality-of-Service (QoS) or to minimize forwarding overhead, an overlay node can behave selfishly in the information collection or in the construction action. The impact of selfishness in the information collection has recently been examined [14][17∼20], but its impact in the construction action remains unclear.

In this paper, we present the first systematic study on the impact of selfishness in both tree and mesh overlay construction. Our investigation considers multiple QoS measures for streaming applications, including latency, streaming rate, and continuity. Our contribution is twofold: First, we analyze how for selfish overlay nodes to choose construction-action policy to optimize their individual multi-metric QoS. Second, we demonstrate that the selfishness-aware construction action will help with optimizing the QoS of the global multicast session, but not vice versa. The implication is significant: A globally optimal overlay construction itself can be vulnerable to individual selfishness; but, following our directions, we can design an overlay that is both globally optimal and selfish-resistant.

The rest of this paper is organized as follows. We introduce the related work in Section II. The model and major notations are presented in Section III. The multiple-metric QoS-aware streaming under selfish policies in construction

This work was supported by the National Natural Science Foundation of China (No. 60403035), the Hi-Tech Research and Development Program of China (No. 2006AA01Z205), and the National Basic Research Program of China (No. 2003CB314801). J. Liu's work was partly supported by a Canadian NSERC Discovery Grant.

0743-166X/07/$25.00 ©2007 IEEE

action is discussed in Section IV and Section V, for tree-based overlay multicast and mesh-based overlay multicast, respectively. Finally, section VI concludes the paper.

II. Related Work

Given the difficulties of deploying IP multicast in the global Internet, overlay multicast has emerged as a promising alternative, particularly for living media streaming. Existing overlay multicast proposals can be broadly classified into two categories according to the overlay structures [27], namely, *tree-based* and *mesh-based*.

In tree-based protocols, each node selects a parent from other participating nodes to receive the streaming data. The parent/children relationships among all nodes compose the overlay structure, i.e., an application-layer multicast tree. Once the tree is established, the data is propagated along the tree branches, and there is no additional control overhead except for occasional branch repair or optimization. Typical protocols belonging to this category include NARADA [3], NICE [4], ZIGZAG [5], Scattercast [6], and Yoid [7], etc.

In mesh-based overlay multicast, there is no explicit parent/children relationship. Each node maintains a number of partners, and the partner relationships among all nodes compose a mesh structure. The streaming data propagated in the mesh is divided into segments. Each node notifies its partners which segments it holds, and requests for segments from partners that hold the segmens. A key advantage of mesh-based overlay multicast is that it can tolerate node dynamics better than tree-based overlay multicast. Mesh-based overlay multicast can be realized by a multi-tree approach, such as SplitStream [8], Bullet [9], CoopNet [10], and etc, or by an unstructured data-driven approach, like PRO [11], CoolStreaming [12], Chainsaw [13], etc.

The optimization of the overall Quality-of-Sevice (QoS) experienced by the users has constantly been the primary design objective for multicast protocols. Many existing protocols have focused on the global optimization and supposed that all the overlay nodes are cooperative. Sripanidkulchai et al. discuss different parent-selecting methods, like randomly policy, minimum-depth-first policy, and longest-first policy [15]. Bishop et al. disclose the advantage of preemption in favor of nodes with higher priorities by experiments [16]. One common characteristic of these researches is that they do not consider the selfishness of individual overlay nodes.

A key difference between overlay multicast and IP multicast, however, is that the overlay nodes are strategic application-layer agents, which can be selfish with their own interests. Naturally, a node would like to receive better QoS as well as to bear less forwarding burden. The impact of the selfish behavior in the information collection has been recently studied. Mathy et al. demonstrate the negative impact of distance cheating among overlay nodes on the stretch and link stress of the multicast tree [14], and Li et al. further study the impact of this kind of cheating on the stability of multicast tree [17]. Habib et al. point out that the QoS of overlay multicast might be negatively influenced if some overlay nodes are not cooperative to contribute resources, which can also be viewed as the information cheating about outgoing bandwidth or available data [18]. Yuen et al. propose a VCG-based strategyproof algorithm to defend cheating about node throughput [19]. Wang et al. study the cheating about link cost in non-cooperative multicast protocols, and also design distributed payment algorithms against this kind of cheating [20]. Later investigations also suggest obtaining the information of other overlay nodes or virtual links by a trustworthy infrastructure, like RandPeer [21].

The selfishness of overlay nodes in the construction action however has yet to be investigated. To the best of our knowledge, the only related work is [22], which proposes an additional payment mechanism. In their work, a node consumes "points" to request for a segment from another node, and earns "points" by sending a segment to another node. The policy of the segment-requesting node to bid for segments and the policy of the segment-providing node to accept bids are also examined, but limited to the QoS considering the network latency and packet loss rate. In this paper, we present a systematic study on the impact of selfishness in construction for both tree and mesh overlay. We consider multiple QoS metrics, including stream latency, resolution, and continuity in our study, and do not impose any additional mechanisms.

III. Models and Definitions

As in existing protocols, we assume that each node maintains only a partial view of other nodes, called *neighbors* [23]. When a node joins a multicast session, it obtains the neighbor list from a central node (e.g., the source node or a node designated by the source), and this list is dynamically updated to suite network changes. The neighboring relationships among all nodes compose the overlay *control structure*. The average number of neighbors each overlay node maintains over the total number of participating nodes is referred to as the *neighbor density*.

There are two steps toward establishing the data delivery structure of an overlay, also called *overlay structure*. In the information collection step, neighboring nodes exchange necessary control information with each other. In the construction step, the nodes send requests to form the overlay from scratch or join an existing overlay. We assume that the nodes are selfish, which strikes to maximize its individual Quality-of-Service (QoS) and minimize the data forwarding overhead. Since tree and mesh represent two distinct approaches for overlay construction, and both have shown their success in theory and practical deployment, we will investigate both of them in this paper.

In live streaming applications, the typical QoS metrics that users care are stream latency, stream resolution, and stream continuity, which are the QoS-measure parameters we use in our model. We also list the major notations used throughout this paper in Tab. I.

IV. Tree-based Overlay Multicast

We begin our study on the tree-based overlay multicast. In this approach, each node selects a parent from the neighbors

TABLE I

MAJOR NOTATIONS IN THIS PAPER

Notations	Definitions
T	The playing duration of a segment of the stream
E	The stream encoding rate on the source node
A	The set of all receiver nodes of the overlay multicast session
n	The total number of receiver nodes in the overlay multicast session
e	The neighbor density of the overlay control structure
Q	The set of all segments of the stream (for mesh-based overlay multicast only)
U	The overall QoS of the multicast session
u_i	The QoS of node i
u_i^q	The segment QoS of node i for segment q (for mesh-based overlay multicast only)
d_i	The source-to-end latency on node i
d_i^q	The source-to-end latency of segment q on node i (for mesh-based overlay multicast only)
f_i^q	The source-to-end latency of segment q on partners of node i that hold the segment when node i needs to request it from a partner (for mesh-based overlay multicast only)
m_{ij}	The distance from node i to node j
r_i	The received stream rate on node i
r_i^q	The received stream rate of segment q on node i (for mesh-based overlay multicast only)
v_i	The total incoming bandwidth of node i
o_i	The total outgoing bandwidth of node i
l_i	The past duration of node i in the multicast session
t_i	The expected future duration of node i in the multicast session
s_i	The average interval between stream pauses on node i in its duration in the multicast session
α	User's weights on the stream latency
β	User's weights on the stream resolution
γ	User's weights on the stream continuity

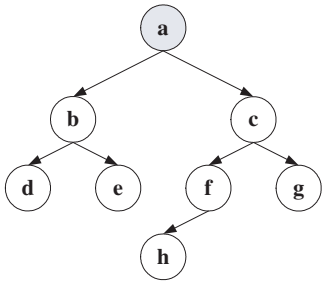

Fig. 1. An example of tree-based overlay structure.

to receive the streaming data, and the parent/children relationships among all nodes compose the overlay structure, i.e., a multicast tree, as illustrated in Fig. 1. Once the multicast tree is established, the data is propagated along the tree and there is no additional control overhead. When a node leaves or fails, all of its descendants will observe a data outage until the multicast tree is repaired.

A. Multi-Metric QoS

In tree-based overlay multicast, the stream latency on a node is evaluated just upon the source-to-end latency of the node, the stream resolution on a node is evaluated upon the received streaming rate on the node, and the stream continuity on a node is evaluated upon the average interval between stream pauses in the duration of the node in the multicast session.

Let u_i denote the QoS of node i, which is to be optimized by this selfish node. The playing duration of a segment of the stream and the stream encoding rate on the source node are both constant, denoted by T and E, respectively. If the source-to-end latency on node i is d_i, the received stream rate on node i is r_i, the duration of node i in the multicast session so far is l_i, and the average interval between stream pauses on node i during its duration in the multicast session is s_i, the QoS of node i in tree-based overlay multicast is expressed as Eq. (1).

$$u_i = \frac{\alpha u_1 + \beta u_2 + \gamma u_3}{\alpha + \beta + \gamma} \tag{1}$$

in which, $u_1 = \log_2(1 + \frac{T}{T+d_i})$, $u_2 = \log_2(1 + \frac{r_i}{E})$, $u_3 = \log_2(1 + \frac{s_i}{l_i})$.

The parameters α, β, and γ represent the user's weights on stream latency, stream resolution, and stream continuity, respectively. The log(.) function is concave, suggesting that the QoS increases more slowly with less source-to-end latency, higher received stream rate, and longer interval between stream pauses. It is easy to prove that the resultant value of u_i is within [0, 1].

We further explain the implication of Eq. (1) as follows:

1) If node i cares stream latency only, i.e., $\beta = 0$ and $\gamma = 0$, the QoS of node i is $u_i = \log_2(1 + \frac{T}{T+d_i})$. In this case, the QoS of node i approaches 1 when the source-to-end latency approaches to 0, and its QoS approaches 0 if the source-to-end latency is excessive.

2) If node i cares stream resolution only, i.e., $\alpha = 0$ and $\gamma = 0$, the QoS of node i is $u_i = \log_2(1 + \frac{r_i}{E})$. Then the QoS of node i approaches 1 when the received stream rate is closer to the stream encoding rate on the source node, and its QoS approaches 0 when the received stream rate is too low.

3) If node i cares stream continuity only, i.e., $\alpha = 0$ and $\beta = 0$, the QoS of node i becomes $\gamma = 0$, the QoS of node i is $u_i = \log_2(1 + \frac{s_i}{l_i})$. Under this situation, the QoS of node i equals 1 when the average interval between stream pauses is just the duration of node i in the multicast session (i.e., there is no stream pause), and its QoS approaches 0 when the average interval between stream pauses is too short (i.e., frequent pauses).

Let A denote the set of all receiver nodes of the overlay multicast session, with a total number of n. The overall QoS of the multicast session, U, is defined as the average QoS of all receiver nodes, i.e.,

$$U = \frac{\sum_{i \in A} u_i}{n} \tag{2}$$

Since each overlay node is selfish and strategic, it will try to optimize its own QoS. We discuss the selfish behavior of overlay nodes during the construction action in tree-based overlay multicast in the following subsections, including both parent selection and children acceptance.

B. Parent Selection

In tree-based overlay multicast, there are two representative parent-selection policies for node i.

Random policy. Randomly select a neighbor as the parent, denoted as x^1.

QoS-aware policy. Select the neighbor that can maximize the QoS of node i as the parent, denoted as x^2.

From the perspective of the selfish node i, the QoS-aware policy is obviously better since this policy optimizes its QoS given the information of neighbors. From the perspective of the overall multicast session, the QoS-aware policy is also better because this policy not only optimizes the QoS of the parent-selecting node i, but also provides higher QoS for other nodes that might select node i as the parent in the future.

The key issue becomes how for the parent-selecting node i to estimate its QoS if selecting some neighbor j as the parent in the multiple-metric environment. The source-to-end latency, the received stream rate and the average interval between stream pauses on node i if selecting neighbor j as the parent can be estimated as follows.

1) Estimation of the source-to-end latency.

In the information collection, neighbor j tells its source-to-end latency, d_j, to its neighbors including node i, and node i measures the distance from neighbor j to itself as m_{ji}. The source-to-end latency on node i if selecting neighbor j as the parent is estimated as $d_i = d_j + m_{ji}$.

2) Estimation of the received stream rate.

Suppose the total incoming bandwidth of node i is v_i. In the information collection, node i learns that the total outgoing bandwidth of neighbor j is o_j, and the received stream rate on neighbor j is r_j. The received stream rate on node i if selecting neighbor j as the parent is estimated as $r_i = min(r_j, o_j, v_i)$.

3) Estimation of the average interval between stream pauses.

In tree-based overlay multicast, the stream pauses experienced by a node are mainly due to the changes of its ancestors. There are two reasons that cause the changes: 1) it is preempted by another node; and 2) an ancestor fails or leaves the session. If a node is frequently preempted by other nodes, it is regarded as having a lower child priority (explained in the next subsection), and will likely be preempted again in the future. Also, according to the measurement study from [16], we assume that the nodes that have already stayed a longer time than others tend to stay longer in the future as well.

Hence, we assume that the node that has experienced longer average interval between stream pauses is likely to have longer average interval between stream pauses in the future. More explicitly, in information collection, node i learns the average interval between stream pauses on neighbor j as s_j. The average interval between stream pauses on node i if selecting neighbor j as the parent is estimated as $s_i = s_j$.

It is worth noting that the duration of a neighbor is also implicitly included in its average interval between stream pauses. For example, if node i selects its parent from two neighbors, j_1 and j_2. Neighbor j_1 stays in the multicast session for 10 seconds and observes 1 stream pause, while j_2 stays in the multicast session for 4 seconds and observes no stream pause. Then neighbor j_1 is assumed to provide node i with longer interval between ancestor changes. This is rational since we suppose the nodes that have stayed longer in the multicast session will stay longer in the future.

Given the parent-requesting node i has the collected information and estimations of the metrics as described above, and it also predicts its future duration in the multicast session as t_i, the parent priority of neighbor node j to node i, P_{ji}, is assigned as Eq. (3).

$$P_{ji} = \frac{\alpha P_1 + \beta P_2 + \gamma P_3}{\alpha + \beta + \gamma} \tag{3}$$

where $P_1 = \log_2(1 + \frac{T}{T+d_j+m_{ji}})$, $P_2 = \log_2(1 + \frac{min(r_j, o_j, v_i)}{E})$, $P_3 = \log_2(1 + \frac{min(s_j, t_i)}{t_i})$.

After assigning the parent priorities to neighbors, the parent-selecting node i will send a parent request with a expected rate (the minimum of its incoming bandwidth and the stream encoding rate) to the neighbor with the highest parent priority. If the parent request is rejected by the best neighbor due to the competition of other nodes, node i will send a parent request to the neighbor with the second highest parent priority, and so on.

We should notice the decision is online. When node i compares the parent priority of the current parent with that of another neighbor, it should take the additional pause to switch to a new parent into consideration.

C. Children Acceptance

An overlay node i can choose to reject all parent requests from other nodes, thus to bear no forwarding burden. However, in an environment where all nodes can choose their own children-acceptance policies, this may not be the best policy to optimize the QoS of node i in the future, because it may re-select parent later due to network dynamics. If node i is willing to accept children within its outgoing bandwidth, and the outgoing bandwidth is enough, all parent requests can be accepted. If the outgoing bandwidth is not enough to accept all parent requests, which is the most common case in overlay multicast, some parent requests have to be rejected or some existing children will be preempted.

We discuss here four representative children-acceptance policies that a selfish node i might adopt.

Negative policy. Not accept any children, denoted as y^1.

Random policy. Randomly accept children within its total outgoing bandwidth, denoted as y^2.

Capacity-aware policy. Prioritize neighbors that might provide higher expected QoS to node i (represented as higher capacity) if becoming node i's parent in the future, denoted as y^3.

Contribution-aware policy. Prioritize neighbors that have ever forwarded more stream data to node i (like tit-for-tat mechanism in BitTorrent [24∼25]), denoted as y^4.

Since a node receives parent requests from neighbors, and will also select its parent from neighbors, its QoS in the future is determined by the children-acceptance policies of its neighbors and its own. The choice of children-acceptance polices of selfish nodes can be modeled as a multi-player game. In this game, the players are the overlay nodes, the strategy space is $\{y^1, y^2, y^3, y^4\}$, and the payoff to each node is its expected QoS in the future. Before solving the

TABLE II

PAYOFF OF NODE i

	y^1_{-i}	y^2_{-i}	y^3_{-i}	y^4_{-i}
y^1_i	0	2	2	1
y^2_i	0	2	2	2
y^3_i	0	2	2	4
y^4_i	0	2	2	3

equilibrium of the multi-player game, we give two definitions in game theory here.

Definition 1: Dominated Strategy

If s_i is the strategy of player i, s_{-i} is the strategy set of all players except player i, and the payoff of player i is $w_i(s_i, s_{-i})$, $s_i^\star$ is called the dominant strategy for player i if it satisfies Eq. (4).

$$w_i(s_i^\star, s_{-i}) \geq w_i(s_i', s_{-i}), \forall s_{-i}, \forall s_i' \neq s_i^\star \tag{4}$$

Definition 2: Dominated Strategy Equilibrium

If $s^\star$ is the strategy set of all players, it is called a dominant strategy equilibrium if $s_i^\star$ is the dominant strategy for each player i.

Theorem 1. The dominated strategy equilibrium in the game of choosing children-acceptance policy in tree-based overlay multicast is that every node adopts policy y^3.

Proof. For each node i, its payoff depends on the children-acceptance policy of its neighbors and its own. For simplicity, node i can assume that the neighbors adopt the same children-acceptance policy. If y_i^k (k=1, 2, 3, 4) denotes that node i adopts policy y^k, and y^k_{-i} (k=1, 2, 3, 4) denotes that the neighbors of node i all adopt the policy y^k, the payoff of node i is shown in Tab. II.

The payoff numbers in Tab. II only represent the *relative* value, not meaning the *absolute* QoS of node i in the future. However, it is enough for us to find the dominant strategy of node i.

Note that among all parent-requesting neighbors, in the future node i will most likely send parent requests to those with higher QoS, since QoS-aware policy is the preferred parent-selection policy. The parent-requesting neighbors that might have higher QoS in the future are called *better potential parents* for node i. Tab. II is explained as follows.

1) If the neighbors all choose policy y^1, the payoff of node i is obviously 0, indicating that node i cannot find any parent from these neighbors in the future. This explains the payoff numbers in the second column of Tab. II.

2) If the neighbors all choose policy y^2, the payoff of node i is 2, no matter which policy node i adopts. Because the child priority of node i to any neighbor node j (including the better potential parents) in the future is random and can be viewed as identical with other neighbors of node j. This explains the payoff numbers in the third column of Tab. II.

3) If the neighbors all choose policy y^3, the payoff of node i is also 2, no matter which policy node i adopts. This is because, compared with other neighbors of node j, the relative capacity of node i to any neighbor node j (including the better potential parents) in the future is also random. This explains the payoff numbers in the fourth column of Tab. II.

4) If the neighbors all choose policy y^4, the payoff of node i is different according to its own policy. We will discuss the four cases (when node i chooses policy y^1, y^2, y^3, y^4) respectively as follows.

i) If node i chooses policy y^2, its payoff is 2. Since node i chooses the random policy, its contribution to any neighbor node j (including the better potential parents) is also random. Thus, the child priority of node i to the contribution-aware neighbor j in the future can be viewed as random. This explains the payoff number in the third line of the fifth column of Tab. II.

ii) If node i chooses policy y^1, its payoff is 1. Node i still has the chance to become the child of any neighbor node j in the future, if node j has available outgoing bandwidth; but the child priority of node i to node j is the lowest, because it provides no data to node j. Thus, the payoff of node i is more than 0 but less than when it adopts policy y^2. This explains the payoff number in the second line of the fifth column of Tab. II.

iii) If node i chooses policy y^4, its payoff is 3. The nodes that have ever contributed more to node i will be better potential parents for node i in the future (recall the discussion about estimating the average interval between stream pauses when selecting parent in the previous section). The QoS of node i in the future will be higher than when it adopts policy y^2, because the better potential parents are likely be served better from a contribution point of view. This explains the payoff number in the fifth line of the fifth column of Tab. II.

iv)If node i chooses policy y^3, its payoff is 4. The payoff of node i is higher than when adopting policy y^4, because the prioritized neighbors are better potential parents if considering the contribution. This explains the payoff number in the fourth line of the fifth column of Tab. II.

Therefore, according to definition 1, y^3 is the dominated strategy of node i. According to definition 2, the dominated strategy equilibrium in the game of choosing children-acceptance policy is that all nodes choose policy y^3.

The next issue is how for node i to assign the child priorities to the parent-requesting nodes under policy y^3. The expected QoS of node i if selecting some parent-requesting node j as the parent in the future is estimated by two factors, that is, the QoS of node i if node j accepts it as a child in the future and the probability of node j accepting node i as a child in the future.

For estimating the QoS of node i if neighbor node j accepts it as a child in the future, node i does not have the information of source-to-end latency of node j, the received stream rate of node j, nor the average interval between stream pauses on node j when node j becomes node i's parent in the future. The information available for node i to estimate are the distance from node j to node i, the total incoming bandwidth and outgoing bandwidth of node j, and the past duration of node j in the multicast session.

On estimating the probability of node j accepting node i as a child in the future, the nodes that have higher outgoing bandwidth and longer duration in the multicast session are prioritized.

Therefore, the child priority of node j to node i, C_{ji}, is

assigned as Eq. (5).

$$C_{ji} = \frac{\alpha C_1 + \beta C_2 + \gamma C_3}{\alpha + \beta + \gamma} * H \quad (5)$$

where $C_1 = \log_2(1+\frac{T}{T+m_{ji}})$, $C_2 = \log_2(1+\frac{min(E,v_j,o_j,v_i)}{E})$, $C_3 = \log_2(1+\frac{min(l_j,t_i)}{t_i})$, and $H = o_j * l_j$.

Eq. (5) indicates that the overlay nodes with higher outgoing bandwidth, higher incoming bandwidth and longer staying time, and those closer to other nodes in the multicast session will be assigned higher child priorities in construction action. We claim that the prioritization of these nodes with higher capacities will also help optimize the overall QoS of the multicast session, which will be demonstrated by the simulations in the following subsection.

D. Simulation

We conduct simulations to study the relationship between the selfish construction-action policy of overlay nodes and the overall QoS of the tree-based overlay multicast session. The simulation is set as follows.

We use the random model to generate the network-layer topology by GT-ITM [26]. There are 2000 routers in the network-layer topology and the link distances between connected routers are within [10ms, 500ms]. The playing duration of a segment of the stream is 5s, and the stream encoding rate on the source node is 1000kbps. To evaluate the overall QoS of the multicast session under different network situations, we test various system configurations. For each configuration, the receiver node with lower sequence number joins earlier in the multicast session and leaves later from the multicast session, which is an important assumption in this paper. The single source node and the receiver nodes are attached to different routers randomly selected among the 2000 routers. The outgoing bandwidth of each node is within [0kbps, 8000kbps], and the incoming bandwidth of each node is within [500kbps, 2000kbps].

To simplify the simulation, all nodes are assumed to adopt the same policy. We compare the overall QoS of the multicast session under different parent-selection policies and children-acceptance policies, that is, x^1+y^2, x^2+y^2, x^1+y^3, x^2+y^3, x^1+y^4, and x^2+y^4. The children-acceptance policy y^1 is not considered since the overlay multicast session cannot be supported if all nodes adopt this policy. We first fix the total number of receiver nodes, n, as 200, and change the neighbor density of the overlay control structure, e, as 10%, 30%, 50%, 70%, and 90%.Then we fix the neighbor density as 20%, and change the total number of receiver nodes as 100, 300, 500, and 700. When evaluating the QoS of the multicast session, user's weights on stream latency, resolution and continuity are all equally set as 1/3.

Figs. 2 and 3 plot the corresponding overall QoS of the multicast session for the two settings above. From both the figures, we can draw the following conclusion. 1) Given the same children-acceptance policy, the overall QoS under QoS-aware parent-selection policy is always better than that under random parent-selection policy. 2) Given the same parent-selection policy, the overall QoS under capacity-aware children-acceptance policy is the best among the three children-acceptance policies. 3) The overall QoS is optimized when all nodes adopt the QoS-aware parent-selection policy and the capacity-aware children-acceptance policy, which is also the preferred choice of selfish individual nodes.

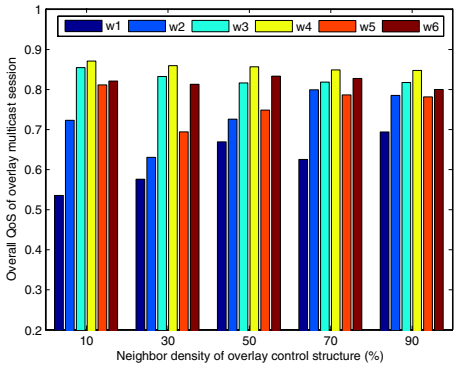

Fig. 2. The overall QoS of the tree-based overlay multicast session (n=200). $w1 = x^1+y^2$, $w2 = x^2+y^2$, $w3 = x^1+y^3$, $w4 = x^2+y^3$, $w5 = x^1+y^4$, and $w6 = x^2+y^4$.

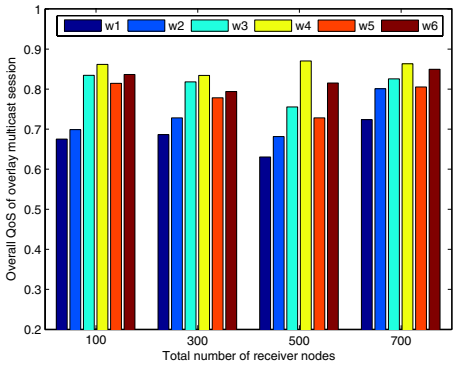

Fig. 3. The overall QoS of the tree-based overlay multicast session (e=20%). $w1 = x^1+y^2$, $w2 = x^2+y^2$, $w3 = x^1+y^3$, $w4 = x^2+y^3$, $w5 = x^1+y^4$, and $w6 = x^2+y^4$.

V. Mesh-based Overlay Multicast

We next discuss the case of mesh-based overlay construction. Unlike the tree-based case, there is no fixed parent/children relationship between overlay nodes in a mesh-based overlay. Instead, each node selects a number of partners from its neighbors in the overlay control structure. The partners are also dynamically adjusted to suite network dynamics. The partner relationships among all nodes compose a mesh structure. In a typical mesh-based overlay, the original stream propagated in the mesh is divided into multiple segments; partners exchange the segment availability information with each other, and each node fetches a certain segment from a partner that holds the segment. Therefore, it is the data availability that drives the propagation of the stream.

Compared with tree-based overlay multicast, the segment notification and segment requests introduce additional control overhead. Yet mesh-based overlay multicast can tolerate

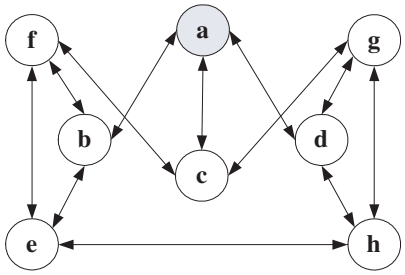

Fig. 4. An example of mesh overlay structure.

node dynamics better. Since a node can receive segments from different partners, the departure or malfunction of an individual node will significantly impact its partners. A node observes stream pause only when a segment is not available on all of its partners before the playback time of the segment. Fig.4 illustrates the overlay structure of mesh-based overlay multicast, with node a being the source.

There are also two steps to establish the overlay structure in mesh-based overlay multicast. In the information collection, each node learns the information of partners, such as the segment availability, the outgoing bandwidth, and the distances from partners to it. In the construction action, each node sends a segment request to the partner selected as the segment-providing node for a certain segment, and responds segment requests within its outgoing bandwidth.

A. Multiple-Metric QoS

In mesh-based overlay multicast, the stream latency on a node is evaluated upon the average source-to-end latency of all segments of the stream, and the stream resolution on a node is evaluated upon the average received streaming rate of all segments. As in tree-based overlay, the stream continuity on a node is evaluated upon the average interval between stream pauses in the duration of the node in the multicast session, but the stream pause in mesh-based overlay multicast is caused by the segment scarcity in partners, not ancestor change, for there are no predefined ancestors.

Let the playing duration of a segment of the stream, the stream encoding rate on the source node, and the segment set of the stream be denoted by T, E, and Q, respectively. The source-to-end latency of segment q on node i is d_i^q, the received stream rate of segment q on node i is r_i^q, the duration of node i in the multicast session so far is l_i, and the average interval between stream pauses on node i during its duration is s_i. Then the segment QoS of node i for segment q, u_i^q, is expressed as Eq. (6).

$$u_i^q = \frac{\alpha u_1^q + \beta u_2^q}{\alpha + \beta} \tag{6}$$

where $u_1^q = \log_2(1 + \frac{T}{T+d_i^q})$, $u_2^q = \log_2(1 + \frac{r_i^q}{E})$.

And the QoS of node i, u_i, is expressed as Eq. (7).

$$u_i = \frac{\alpha u_1 + \beta u_2 + \gamma u_3}{\alpha + \beta + \gamma} \tag{7}$$

where $u_1 = \frac{\sum_{q \in Q} \log_2(1+\frac{T}{d_i^q+T})}{|Q|}$, $u_2 = \frac{\sum_{q \in Q} \log_2(1+\frac{r_i^q}{E})}{|Q|}$, $u_3 = \log_2(1 + \frac{s_i}{l_i})$.

The overall QoS of the multicast session, U, is also defined as the average QoS of all receiver nodes, as in Eq. (2). We discuss the selfishness of overlay nodes in the construction action of mesh-based overlay multicast in the following subsections, which includes the segment request and the segment response policies.

B. Segment Request

Each node exchanges segment availability information with partners, and finds which partners hold certain segments. If a segment is held by multiple partners, a segment-providing node is selected among these partners for the segment. There are also two representative policies for node i to select the segment-providing node for segment q.

Random policy. Randomly select a partner that holds segment q as the segment-providing node, denoted as $x^{1'}$.

QoS-aware policy. Select the partner that can maximize the segment QoS of node i for segment q as the segment-providing node, denoted as $x^{2'}$.

As in tree-based overlay multicast, QoS-aware policy is the optimal choice of selfish overlay nodes and will help optimize the overall QoS of the multicast session. The key issue here is how for the segment-requesting node i to estimate its segment QoS for segment q if selecting some partner j as the segment-providing node for the segment.

In mesh-based overlay multicast, the requesting segment q is stored in the segment-providing node. Therefore, when node i decides to request segment q, the source-to-end latencies of segment q on the partners that hold it are identical, denoted as f_i^q. In the information collection, node i also needs to measure the distance from partner j to it, m_{ij}, and learn the outgoing bandwidths of partner j, o_j. The segment-request priority of node j to node i for segment q, R_{ji}^q, is assigned as Eq. (8).

$$R_{ji}^q = \frac{\alpha R_1^q + \beta R_2}{\alpha + \beta} \tag{8}$$

where $R_1^q = \log_2(1 + \frac{T}{T+m_{ji}+f_i^q})$, $R_2 = \log_2(1 + \frac{min(E,o_j,v_i)}{E})$.

The segment-requesting node i will send a segment request with a requiring rate (the minimum of its incoming bandwidth and the stream encoding rate) to the partner that has the highest segment-request priority for the segment. If the segment request is rejected by the best partner due to competition from other nodes, node i will send a segment request to the partner with the second highest segment-request priority, and so on.

C. Segment Response

When a node receives multiple segment requests from partners, it should have a segment-response policy to accept the segment requests within its total outgoing bandwidth. Similar to those in tree-based overlay multicast, there are four representative segment-response policies for selfish node i.

Negative policy. Not respond any segment request, denoted as $y^{1'}$.

Random policy. Randomly respond segment requests within its total outgoing bandwidth, denoted as $y^{2'}$.

Capacity-aware policy. Prioritize partners that might provide higher expected segment QoS to node i (represented as higher capacity) if becoming node i's segment-providing nodes in the future, denoted as $y^{3'}$.

Contribution-aware policy. Prioritize partners that have ever forwarded more segments to node i, denoted as $y^{4'}$.

We can still use a multi-player game to study the choice of the segment-response policy. In this game, the players are the overlay nodes, the strategy space is $\{y^{1'}, y^{2'}, y^{3'}, y^{4'}\}$, and the payoff of each node is its expected segment QoS in the future.

Theorem 2. The dominated strategy equilibrium in the game of choosing segment-response policy in mesh-based overlay multicast is that each node adopts the policy $y^{3'}$.

The proof is similar to that in theorem 1.

We next discuss how for node i to assign the segment-response priority to a segment-requesting node j when adopting the capacity-aware segment-response policy. To predict the future segment QoS of node i if choosing node j as the segment-providing node, node i needs to learn the distance from node j to it and the total outgoing bandwidth of node j. In addition, node i estimates the probability of its segment request being accepted by node j in the future. The nodes that have higher outgoing bandwidths and stay longer in the multicast session are assumed to accept the segment request of node i with higher probability in the future. Therefore, the segment-response priority of node j to node i, G_{ji}, is assigned as Eq. (9).

$$G_{ji} = \frac{\alpha G_1 + \beta G_2}{\alpha + \beta} * H \tag{9}$$

where $G_1 = \log_2(1 + \frac{T}{T+m_{ji}})$, $G_2 = \log_2(1 + \frac{min(E,o_j,v_i)}{E})$, $H = o_j * l_j$.

Eq. (9) indicates that the overlay nodes with higher outgoing bandwidth and longer staying time, and those closer to other overlay nodes will have higher priorities in the construction action in mesh-based overlay multicast. In fact, the prioritization for these nodes will also help optimize the overall QoS of the multicast session, which will be demonstrated by the simulations in the following subsection.

D. Simulation

We conduct simulations to study the relationship between the selfish construction-action policy of overlay nodes and the overall QoS of the mesh-based overlay multicast session. The simulation set of the overlay control structure is similar to the simulation in tree-based overlay multicast presented in the previous section. The stream on the source node is divided into 600 segments. The number of partners each node maintains is set as 4, which is a recommended value in [12].

All selfish nodes are assumed to adopt the same policy, and we also compare the overall QoS of the multicast session under different segment-request policies and segment- response policies, that is, $x^{1'} + y^{2'}$, $x^{2'} + y^{2'}$, $x^{1'} + y^{3'}$, $x^{2'} + y^{3'}$, $x^{1'} + y^{4'}$, and $x^{2'} + y^{4'}$. The segment-response policy $y^{1'}$ is not considered. To evaluate the overall QoS under different network situations, we first fix the total number of receiver nodes, n, to 200, and change the neighbor density of the overlay control structure, e, as 10%, 30%, 50%, 70%, and 90%; We then fix the neighbor density to 20%, and change the total number of receiver nodes as 100, 300, 500, and 700. When evaluating the QoS of the multicast session, user's weights on stream latency, resolution and continuity are all equally set as 1/3.

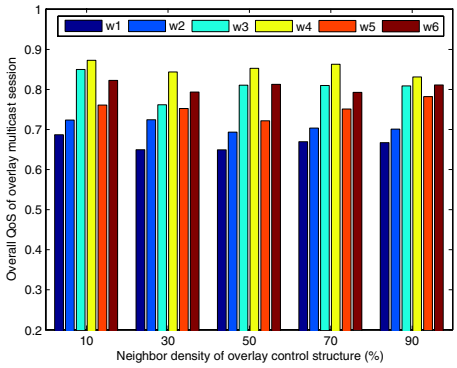

Fig. 5. The overall QoS of the mesh-based overlay multicast session (n=200). $w1 = x^{1'} + y^{2'}$, $w2 = x^{2'} + y^{2'}$, $w3 = x^{1'} + y^{3'}$, $w4 = x^{2'} + y^{3'}$, $w5 = x^{1'} + y^{4'}$, and $w6 = x^{2'} + y^{4'}$.

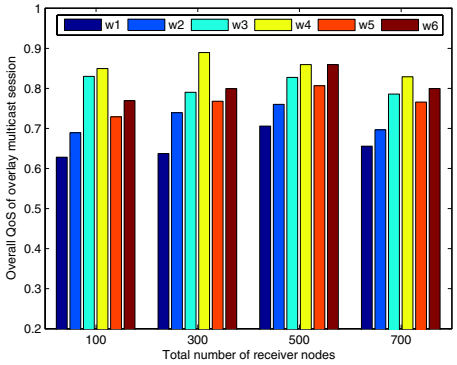

Fig. 6. The overall QoS of the mesh-based overlay multicast session (e=20%). $w1 = x^{1'} + y^{2'}$, $w2 = x^{2'} + y^{2'}$, $w3 = x^{1'} + y^{3'}$, $w4 = x^{2'} + y^{3'}$, $w5 = x^{1'} + y^{4'}$, and $w6 = x^{2'} + y^{4'}$.

Figs. 5 and 6 illustrate the corresponding overall QoS of the multicast session for the settings above. From these two figures, we can get the following results. 1) Given the same segment-response policy, the overall QoS under QoS-aware segment-request policy is always better than under random segment-request policy. 2) Given the same segment-request policy, the overall QoS under capacity-aware segment-response policy is the best among the three segment-response policies. 3) The overall QoS is optimized when all nodes adopt the QoS-aware segment-request policy and the capacity-aware segment-response policy, which is also the preferred choice of selfish individual nodes.

VI. Conclusion

A unique and important characteristic of overlay multicast is that the overlay nodes can be selfish with their own interests. The selfishness can easily defeat the previous efforts on overlay construction that focus on the global optimization with cooperative nodes only. In this paper, we presented a systematical study on the impact of selfishness in both tree and mesh overlay construction. Our investigation considered multiple QoS measures for streaming applications, including stream latency, stream resolution, and stream continuity. We analyzed how for selfish overlay nodes to choose the construction-action policy to optimize their own multiple-metric QoS, and demonstrated by analytical and simulation results that the selfish construction-action policy of overlay nodes will help optimize the overall QoS of the multicast session.

References

[1] J. Liu, B. Li, and Y. Q. Zhang, *Adaptive Video Multicast over the Internet*. *IEEE Multimedia*, 10(1):22-31, 2003

[2] S. E. Deering, *Multicast Routing in Internetworks and Extended LANs*. In Proceedings of *ACM SIGCOMM'88*, Stanford, CA, USA, Aug 1988

[3] Y. H. Chu, S. G. Rao, and H. Zhang, *A Case for End System Multicast*. In Proceedings of *ACM SIGMETRICS'00*, Santa Clara, CA, USA, Jun 2000

[4] S. Banerjee, B. Bhattacharjee, and C. Kommareddy. Scalable Application Layer Multicast. In Proceedings of *ACM SIGCOMM'02*, Pittsburgh, PA, USA, Aug 2002

[5] D. A. Tran, K. A. Hua, and T. Do, *Zigzag: An efficient peer-to-peer scheme for media streaming.* In Proceedings of *IEEE INFOCOM'03*, San Franciso, CA, USA, Mar/Apr 2003

[6] Y. Chawathe, Scattercast: *An Architecture for Internet Broadcast Distribution as an Infrastructure Service. Ph.D. Thesis*, University of California, Berkekey, Dec, 2000

[7] D. Pendarakis, S. Shi, D. Verma, and M. Waldvogel, *ALMI: An Application Level Multicast Infrastructure*. In Proceedings of *USITS'01*, San Francisco, California, USA, Mar 2001

[8] P. Francis, *Yoid: Extending the Internet Multicast Architecture*. *White Paper*, http://www.icir.org/yoid

[9] D. Kostic, A. Rodriguez, J. Albrecht, and A. Vahdat, *Bullet: High Bandwidth Data Dissemination Using an Overlay Mesh*. In Proceedings of *ACM SOSP'03*, Bolton Landing, NY, Oct 2003

[10] V. N. Padmanabhan, H. J. Wang, P. A. Chou, and K. Sripanidkulchai, *Distributing Streaming Media Content Using Cooperative Networking*. In Proceedings of *NOSSDAV'02*, Miami Beach, FL, USA, May 2002

[11] R. Rejaie and S. Stafford, *A framework for architecting peer-to-peer receiver-driven overlays*. In Proceedings of *NOSSDAV'04*, Cork, Ireland, Jun 2004

[12] X. Zhang, J. Liu, B. Li, and T. P. Yum, *CoolStreaming/DONet: A data-driven overlay network for efficient live media streaming*. In Proceedings of *IEEE INFOCOM'05*, Miami, FL, Mar 2005

[13] V. Pai, K. Kumar, K. Tamilmani, V. Sambamurthy, and A. E. Mohr, *Chainsaw: Eliminating Trees from Overlay Multicast*. In Proceedings of *IPTPS'05*, Ithaca, NY, USA, Feb 2005

[14] L. Mathy and N. Blundell, *Impact of Simple Cheating in Application-Level Multicast*. In Proceedings of *IEEE INFOCOM'04*, Hong Kong, China, Mar 2004

[15] K. Sripanidkulchai, A. Ganjam, B. Maggs, and H. Zhang, *The Feasibility of Supporting Large-Scale Live stream Applications with Dynamic Application End-Points*. In Proceedings of *ACM SIGCOMM'04*, Portland, Oregon, USA, Aug/Sep 2004

[16] M. Bishop, S. Rao, and K. Sripanidkulchai, *Considering Priority in Overlay Multicast Protocols under Heterogeneous Environments.* In Proceedings of *IEEE INFOCOM'06*, Barcelona, Spain, Apr 2006

[17] D. Li, Y. Cui, K. Xu, and J. Wu, *Impact of Receiver Cheating on the Stability of ALM Tree*. In Proceedings of *IEEE GLOBECOM'05*, St. Louis, Missouri, USA, Nov/Dec 2005

[18] A. Habib and J. Chuang, *Incentive Mechanism for Peer-to-Peer Media Streaming*. In Proceedings of *IWQOS'04*, Montreal, Canada, Jun 2004

[19] S. Yuen and B. Li, *Strategyproof Mechanisms for Dynamic Multicast Tree Formation in Overlay Networks*. In Proceedings of *IEEE INFOCOM'05*, Miami, Florida, USA, Mar 2005

[20] W. Wang, X. Li, Z. Suny, and Y. Wang, *Design Multicast Protocols for Non-Cooperative Networks*. In Proceedings of *IEEE INFOCOM'05*, Miami, Florida, USA, Mar 2005

[21] J. Liang and K. Nahrstedt, *RandPeer: Membership Management for QoS Sensitive Peer-to-Peer Applications*. In Proceedings of *IEEE INFOCOM'06*, Barcelona, Spain, Apr 2006

[22] G. Tan, and S. A. Jarvis, *A Payment-based Incentive and Service Differentiation Mechanism for Peer-to-Peer Streaming Broadcast*. In Proceedings of *IWQOS'06*, Yale University, New Haven, CT, USA, Jun 2006

[23] T. Moscibroda, S. Schmid, R. Wattenhofer, *On the Topologies Formed by Selfish Peers*. In Proceedings of *IPTPS'06*, Santa Barbara, USA, Feb 2006

[24] D. Qiu and R. Srikant, *Modeling and Performance Analysis of BitTorrent-Like Peer-to-Peer Networks*. In Proceedings of *ACM SIGCOMM'04*, Portland, OR, USA, Aug 2004

[25] A. R. Bharambe and C. Herley, *Analyzing and Improving BitTorrent Performance*. *Microsoft Research Report*, No. MSR-TR-2005-03, Microsoft Research, 2005

[26] E. Zegura, K. Calvert, and S. Bhattacharjee, *How to Model an Internetwork*. In Proceedings of *IEEE INFOCOM'96*, San Francisco, CA, USA, Mar 1996

[27] J. Liu, S. G. Rao, B. Li, and H. Zhang, *Opportunities and Challenges of Peer-to-Peer Internet Video Broadcast*. *Technical Report*, 2006.