Strategyproof Mechanisms for Dynamic Multicast Tree Formation in Overlay Networks

Selwyn Yuen, Baochun Li
Department of Electrical and Computer Engineering
University of Toronto
{*swsyuen,bli*}*@eecg.toronto.edu*

***Abstract*— In overlay multicast, every end host forwards multicast data to other end hosts in order to disseminate data. However, this cooperative behavior cannot be taken for granted, since each overlay node is now a strategic end host. Ideally, a *strategyproof* mechanism should be provided to motivate cooperations among overlay nodes so that a mutually beneficial multicast tree topology results. In this paper, we apply *mechanism design* to the overlay multicast problem. We model the overlay network using the two scenarios of variable and single rate sessions, and further design distributed algorithms that motivate each node towards a better multicast tree. Since network parameters and constraints change dynamically in reality, our protocol dynamically adapts to form a better multicast tree. The correctness and performance of each distributed algorithm are verified by extensive implementation results on PlanetLab.**

***Index Terms*— Economics, Experimentation with real networks/testbeds.**

I. Introduction

Overlay multicast refers to the construction of a multicast tree at the application layer from a data source to multiple receivers, and it enjoys the attractive advantage that the application layer offers unprecedented flexibility and freedom to design new algorithms. Unfortunately, such freedom does not come without challenges: each overlay node is now a *selfish* and *strategic* end host, rather than an obedient router, and cooperative behavior between nodes can no longer be assumed. This means that each node will choose its actions that maximizes its *private utility*, and may be reluctant to replicate and forward messages to downstream children, since forwarding messages to downstream nodes incurs costs.

In this paper, our most important contribution is to apply the theory of *mechanism design* to the overlay multicast problem, and to design a strategyproof mechanism (to be precisely defined in Sec. II) for all multicast *subscribers* to forward messages downstream, while achieving a globally optimal overlay topology in terms of maximum system throughput and minimum forwarding costs. Our solution is based on the celebrated Vickrey-Clarke-Groves (VCG) mechanism [1] from microeconomics. In essence, our solution quantifies the (positive or negative) effects of each node's action *to the rest of the network*. This effect, termed *externality*, must be calculated by the truthful revelation of private information by each node to the public. Once the externality is quantified, each node can take the optimal action and join to the node on the existing multicast tree that will result in the maximum system-wide valuation.

Our objective in this paper is to design *practical* and distributed algorithms based on insights from the theory of mechanism design. In practice, an overlay multicast session may support applications with diverse Quality-of-Service (QoS) requirements. For *delay-sensitive* applications such as multimedia streaming, it is not permitted to buffer data on intermediate relay nodes in the overlay multicast tree, which has been the implicit assumptions of most of the tree construction algorithms. On the other hand, for *delay-insensitive* applications, the intermediate nodes may take advantage of its secondary storage to implement aggressive buffering, even to cache the entire data stream. Such extensive buffering significantly improves throughput at the costs of delay, since peers may exchange missing elements of data at their convenience. This has been the assumption of mesh construction algorithms such as Digital Fountain [2] or Bullet [3]. We seek to design strategyproof algorithms for overlay multicast tree construction in both of these cases, referred to as the scenario of *single rate sessions* and *variable rate sessions*, respectively.

We have successfully applied our theoretical results to the design of a set of distributed algorithms and a working protocol implementation. Through careful and extensive experiments on PlanetLab [4], we have evaluated the correctness, performance, and efficiency of our protocol. Results have shown that our distributed protocol not only converges to the correct solution, but also significantly increases the system throughput in both the variable and single rate scenarios. The message overhead of our protocol is also explored in our experimental studies.

The remainder of this paper is organized as follows. Sec. II first introduces some background information on the VCG mechanism. Sec. III defines our notations and network model, and formalizes our problem statement. In Sec. IV, we present our theoretical solution, distributed algorithms and protocol implementation of the variable rate scenario, followed by a discussion on the experimental results. Sec. V runs in parallel to Sec. IV, except that we shift our attention to the single rate scenario. We discuss related work in Sec. VI, and conclude in Sec. VII.

II. Mechanism Design and the VCG Mechanism

Mechanism design takes the inverse approach when compared to traditional game theory. Instead of finding the equilibria from the model of the game, we have the freedom to choose what outcome we wish the game to stabilize at. Based on the

desired outcome, we *manipulate* the utility function so that the desired outcome is achieved. We give a brief introduction to the concepts and terminologies of mechanism design; for a more in-depth treatment of the subject, the readers are referred to [1], [5].

Consider a game of n nodes. Each node can decide on a strategy $s_i \in S_i$ based on their utility function $u_i(s_i, s_{-i})$. S_i is called the strategy space of i, and s_{-i} simply means the set of strategies chosen by all nodes except i. The following definition formally defines the dominant strategy equilibrium.

Definition 1: A *dominant strategy equilibrium* s^* satisfies the condition $u_i(s_i^*, s_{-i}) \geq u_i(s_i, s_{-i})$ for all i and all strategies (s_i, s_{-i}).

When some private information of a node is announced to the public, it is possible that the node is lying about this information in order to obtain a higher utility. Fortunately, a well-studied class of mechanisms called *strategyproof mechanisms* states that each node can be motivated to become honest, or at least does not have any incentive to lie. In economics, private information is usually called *type*, denoted by $\theta_i \in \Theta_i$ (the type space). We define strategyproofness as follows.

Definition 2: A mechanism is *strategyproof* if for every node i: (1) the strategy space is to declare their types, $S_i = \Theta_i$; and (2) declaring the true type is a dominant strategy, $s_i^* = \theta_i$.

In essence, a solution to the mechanism design problem should: (1) define the desired outcome $o^*(\cdot)$ of the game; and (2) manipulate the utility $u_i(\cdot)$ of each node through payment $p_i(\cdot)$ to achieve the desired outcome. In most mechanism design literature, the utility function is assumed to be *quasi-linear* [1], [5], [6], [7]. This means that the utility function $u_i(\cdot)$ is the sum of the valuation function $v_i(\cdot)$ and the payment function $p_i(\cdot)$, *i.e.*, $u_i = v_i + p_i$. We make the simple, but important, distinction between the utility of i (u_i) and the valuation of i (v_i). The valuation v_i is the intrinsic value of a certain system state to node i. This value cannot be altered externally. However, the utility u_i experienced by node i can be altered by controlling the payment p_i.

The celebrated *VCG mechanism* has been shown to be strategyproof, defined as follows.

Definition 3: A *Vickrey-Clarke-Groves (VCG) mechanism* is the family of mechanisms $M(\theta) = (o(\theta), p(\theta))$ such that:

$$o^*(\theta) \in \arg\max_o \sum_{j=1}^{n} v_j(\theta_j, o(\theta)) \tag{1}$$

$$\begin{aligned} p_i(\theta) &= \sum_{j \neq i} v_j(\theta_j, o^*(\theta)) - \sum_{j \neq i} v_j(\theta_j, o^*_{-i}(\theta_{-i})) && (2) \\ &= \sum_{j \neq i} (v_j - v_j^{-i}) && (3) \end{aligned}$$

Eq. (1) is the VCG outcome function, and it states that the VCG outcome is found by optimizing the total valuation of a network. Eq. (2) is the VCG payment function, which can be interpreted intuitively. The first term in the VCG payment function is the total valuation of the system excluding node i's valuation, given that the optimal outcome $o^*(\theta)$ is achieved. The second term is similar to the first, except we assume that *node i were to withdraw from the game when finding the optimal outcome* $o^*_{-i}(\theta_{-i})$. Eq. (3) combines the two summations into one, and simplifies the notations. v_j^{-i} can be interpreted as the valuation of node j *given that node i is not participating in the game*. Note that the summation is over all node j except node i, to whom we wish to calculate payment. Thus, the VCG payment p_i is independent of v_i. This is one way to see that node i does not have any incentive to lie about its private information, since doing so does not increase its received payment p_i.

If we substitute Eq. (3) to the quasi-linear utility function, we have:

$$\begin{aligned} u_i &= v_i + \left(\sum_{j \neq i} v_j - \sum_{j \neq i} v_j^{-i} \right) && (4) \\ &= \sum_{j} v_j - \sum_{j \neq i} v_j^{-i} && (5) \end{aligned}$$

Eq. (5) simply combines the first two terms in Eq. (4). Intuitively, Eq. (5) implies that the VCG payment function pays a node to the point that its utility u_i is equal to the added value it brought to the system as a whole.

III. Notations and Network Model

Consider an overlay network modeled as a directed graph (N, E), where N represents the set of overlay network nodes and E defines the set of directed virtual links. Let $n = |N|$ be the number of overlay nodes in the network.

Let $b_{ij} \geq 0$ represent the throughput of link ij. Consider the benefit m to each node of receiving one fixed-length multicast data message. It is reasonable to assume that the amount of benefit experienced by node i depends on the data throughput of the incoming connection. We capture this dependency by modeling $m(b_{\mathcal{P}_i i})$ as a function of the incoming throughput $b_{\mathcal{P}_i i} \geq 0$, where $\mathcal{P}_i$ is the parent of i. When there is no risk of ambiguity, we sometimes drop the first subscript of $b_{\mathcal{P}_i i}$ and simply write b_i. So $m(b_{\mathcal{P}_i i})$ is usually written as $m(b_i)$.

We assume that the benefit function is always non-negative, *i.e.*, $m(b_i) \geq 0$ for all $b_i \geq 0$. $m(b_i)$ must also be a non-decreasing function, since the benefit to a node cannot possibly decrease for a higher throughput rate.

Associated with each overlay node i is a forwarding cost c_i, which represents the cost to forward one message to a downstream node. In our overlay multicast problem, $m(b_i)$ and c_i are both private information to individual nodes. Let A_i be the set of neighbor nodes that node i is aware of, *i.e.*, node i has the IP addresses of all nodes in A_i. We define $\mathcal{C}_i$ to be the set of children of node i, and define $\mathcal{P}_i$ to be the parent of node i. Therefore, $\mathcal{C}_i \subseteq A_i$ and $\mathcal{P}_i \in A_i$. We also define $\mathcal{GP}_i$ to be the set of all ancestors (parent, grandparents, *etc.*) of i, and $\mathcal{GC}_i$ to be the set of all descendants (children, grandchildren, *etc.*) of i. In other words, $\mathcal{GC}_i$ includes all nodes in the subtree rooted at i, except i itself. Therefore, $\mathcal{P}_i \in \mathcal{GP}_i$ and $\mathcal{C}_i \subseteq \mathcal{GC}_i$. In addition, we also use the notation T_i to represent the set of all nodes in the subtree rooted at i, *i.e.*, $T_i = \{\mathcal{GC}_i \cup i\}$. If r is the root of the multicast tree, *i.e.*, the source of multicast data, T_r would represent all nodes in a multicast tree.

Finally, we model the finite capacity of the network as per-node throughput limits. This means that the total incoming

throughput must not exceed the limit L_{in}, whereas the total outgoing throughput must not exceed the limit L_{out}. Therefore, for every node i, the *inflow constraint* is $b_{\mathcal{P}_i i} \leq L_{in,i}$, and the *outflow constraint* is $\sum_{j \in \mathcal{C}_i} b_{ij} \leq L_{out,i}$. Since a node cannot forward messages faster than the speed at which it receives them, we have the *data constraint* $b_{\mathcal{P}_i i} \geq b_{ij}$ for each node $i, j \in \mathcal{C}_i$.

Table. I summarizes the general notations introduced so far. Context-specific notations will be introduced as necessary in Sec. IV and Sec. V.

Parameter	Meaning of Parameter
$n = \|N\|, n \geq 0$	Number of nodes
A_i	Set of nodes adjacent to i
$\mathcal{P}_i$	Current parent of i
$\mathcal{C}_i$	Current set of children of i
$\mathcal{GP}_i$	Set of all ancestors of i
$\mathcal{GC}_i$	Set of all descendants of i
r	Root node of a multicast tree
T_i	The set of nodes in the subtree rooted at i
T_r	The set of all nodes in a multicast tree
$b_{ij}, b_{ij} \geq 0$	Throughput of link ij
$m(b_i), m(b_i) \geq 0$	Benefit (per-message) to each node
$c_i, c_i \geq 0$	Relay cost of node i
$L_{in,i}, L_{in,i} \geq 0$	Incoming throughput limit of node i
$L_{out,i}, L_{out,i} \geq 0$	Outgoing throughput limit of node i

TABLE I

SUMMARY OF NOTATIONS IN THE OVERLAY MULTICAST PROBLEM

As mentioned, we consider both delay-sensitive and delay-insensitive applications by modeling the network session as either single or variable rate sessions. This corresponds to network nodes with a *filled* or an *unfilled* buffer, respectively. Specifically, in single rate sessions, a node with a *filled* buffer must receive and send messages at the same throughput, *i.e.*, $b_{\mathcal{P}_i i} = b_{ij}$, thus messages received from upstream will be sent downstream in the most expeditious fashion. On the other hand, in variable rate sessions, a node with its buffer *unfilled* may queue up messages, thereby sending messages to various downstream nodes (potentially) at different throughputs.

We explore the scenarios of variable and single rate sessions in Fig. 1. The white nodes in Fig. 1(a) represent unfilled buffer nodes, whereas the black nodes in Fig. 1(b) represent filled buffer nodes. We will follow this convention in all subsequent figures in this paper. In both figures, node p is receiving data at a rate of 100 KBps from node r through a live data session. p will then make three copies of each message, and send it to the three downstream children a, b, and c respectively. Each node is also labeled with two numbers in square brackets. The number on top represents the incoming throughput limit of i, *i.e.*, $L_{in,i}$, and the bottom number represents the outgoing throughput limit of i, *i.e.*, $L_{out,i}$.

The Scenario of Variable Rate Sessions

In Fig. 1(a), $L_{out,p}$ takes an outgoing throughput limit of 230 KBps. Link pb is constrained by the inflow constraint, and will experience 50 KBps. Then, the remaining 180 KBps should be evenly distributed between pa and pc, *i.e.*, $b_{pa} = 90$ KBps and $b_{pc} = 90$ KBps. The resulting throughputs are $(b_{pa}, b_{pb}, b_{pc}) = (90, 50, 90)$.

In fact, this corresponds to the classic max-min fairness allocation of network resources. *Max-min fairness* requires

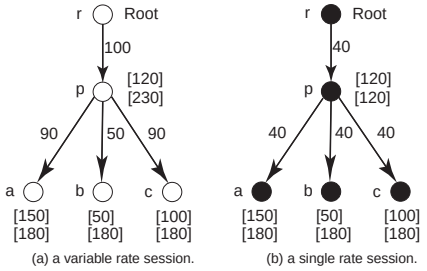


Fig. 1. Illustration of variable and single rate sessions

that: (1) No child j will get a throughput b_{ij} larger than its demand; and (2) All children j with unsatisfied demand will get equal shares of the outgoing throughput at i. We denote the max-min fair allocation of throughput by node i to node j as $L_{fair,ij}$, and use `max-min()` as a function that will allocate the resource $L_{out,i}$ to all demands D_{ij}. In Fig. 1(a), each downstream child of p has a resource demand D_{pj} that equals to $\min(b_{rp}, L_{in,j})$ for $j \in \{a, b, c\}$. In general notations,

$$\begin{aligned} L_{fair,ij} &= \text{max-min}(L_{out,i}, \{D_{i1}, D_{i2}, \ldots, D_{i|\mathcal{C}_i|}\}) \\ \text{where } D_{ij} &= \min(b_{\mathcal{P}_i i}, L_{in,j}) \qquad \forall \quad i, j \in \mathcal{C}_i \end{aligned}$$

The Scenario of Single Rate Sessions

In Fig. 1(b), the outgoing throughput limit $L_{out,p}$ is set to 120 KBps and node p is assumed to have a filled buffer. Suppose all links have throughputs 50 KBps, then the outflow constraint $b_{pa} + b_{pb} + b_{pc} \leq L_{out,p}$ will be violated. In this case, max-min fairness requires that the outgoing throughput of 120 KBps at p to be evenly distributed among a, b, and c. Therefore, $(b_{pa}, b_{pb}, b_{pc}) = (40, 40, 40)$. These slow downstream throughputs will throttle the upstream throughput, resulting in $b_{rp} = 40$. We see that filled buffer nodes tend to decrease the total network throughput.

IV. THE SCENARIO OF VARIABLE RATE SESSIONS

Before we apply the VCG mechanism to our scenario, we need to first quantify the notion of each node's valuation. In this paper, we capture the valuation v_i of each message to each node i intuitively in the form of benefit minus cost. If $|\mathcal{C}_i|$ is the number of children of i, and c_i is the unit forwarding cost, the valuation is simply the benefit $m(b_i)$ minus the total cost $|\mathcal{C}_i|c_i$:

$$v_i = m(b_i) - |\mathcal{C}_i|c_i \tag{6}$$

The values $m(b_i)$ and c_i are both private information to node i, and are not known to any other node in the overlay network. However, strategyproofness of our solution will ensure that each node honestly reports the true value of these information.

A. The VCG Payment

All previous mechanism design literatures that we are aware of have either explicitly or implicitly assumed existence of a trusted third party who is responsible for transferring payments [6], [7], [8], [9]. We have the same assumption here, and adopt the convention that a payment $p_i > 0$ is a payment made from a trusted third party to node i, whereas $p_i < 0$ means that node i must pay the trusted third party. In the context of overlay multicast, the trusted third party is assumed to be the root of

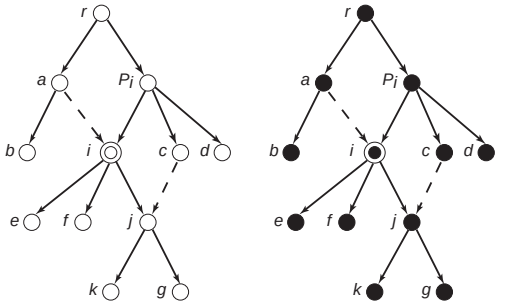


Fig. 2. Derivation of VCG solution at node i

the multicast tree, but can be any third party bank in reality. We will next derive the correct VCG payment according to Eq. (3).

Eq. (3) suggests that in order to calculate p_i correctly, we need to know the difference between v_j and v_j^{-i} for all $j \neq i$. In other words, we need to quantify the net effect of the entrance of i on the rest of the system, *i.e.*, the *externality* of its entrance.

To derive p_i, we use Fig. 2(a) as a visual aid, which depicts the scenario where node i is an internal node with descendants $\mathcal{GC}_i$. Again, the white nodes represent unfilled buffer nodes, and the black nodes represent filled buffer nodes. There are three disjoint sets of nodes to consider, namely the parent $\mathcal{P}_i$, the descendants $\mathcal{GC}_i$, and nodes in other subtrees $\{T_{\mathcal{P}_i} \setminus T_i\}$. Eq. (3) can be expanded as follows:

$$\begin{aligned} p_i &= (v_{\mathcal{P}_i} - v_{\mathcal{P}_i}^{-i}) + \sum_{j \in \{\mathcal{GC}_i \cap \mathcal{P}_j^{-i} \exists\}} (v_j - v_j^{-i}) \\ &\quad + \sum_{j \in \{\mathcal{GC}_i \cap \mathcal{P}_j^{-i} \nexists\}} (v_j - v_j^{-i}) + \sum_{j \in \{T_r \setminus T_i\}} (v_j - v_j^{-i}) \qquad (7) \\ &= -c_{\mathcal{P}_i} + \sum_{j \in \{\mathcal{GC}_i \cap \mathcal{P}_j^{-i} \exists\}} (m(b_j) - m(b_j^{-i})) \\ &\quad + \sum_{j \in \{\mathcal{GC}_i \cap \mathcal{P}_j^{-i} \nexists\}} (m(b_j) - |\mathcal{C}_j| c_j) \\ &\quad + \sum_{j \in \{T_{\mathcal{P}_i} \setminus T_i\}} (m(b_j) - m(b_j^{-i})) \qquad (8) \\ &= P_1 + P_2 + P_3 + P_4 \end{aligned}$$

The first term in Eq. (7) simply indicates that the entrance of i costs its parent $\mathcal{P}_i$ exactly $c_{\mathcal{P}_i}$ to forward each message. The second term in Eq. (7) refers to the descendant of i that has a *second-best parent* $\mathcal{P}_j^{-i}$, *i.e.*, an alternate parent other than the optimal parent. In this case, j still needs to forward to the same set of children $\mathcal{C}_j$ with the same forwarding cost c_j, *i.e.*, $c_j^{-i} = c_j$ and $|\mathcal{C}_j| = |\mathcal{C}_j^{-i}|$. Therefore, the cost component of the valuation function cancels out, and only the difference in benefits $(m(b_j) - m(b_j^{-i}))$ is left. The same argument applies for the fourth term as well. On the other hand, the third term in Eq. (7) refers to the descendant that has no alternate parent. In this case, j has no way of receiving the multicast data except from i, so when i is not participating, the benefit to j becomes zero, and no forwarding can be done, *i.e.*, $m(b_j^{-i}) = 0$, $c_j^{-i} = 0$, and $|\mathcal{C}_j| = 0$. Therefore, we are left with $v_j = m(b_j) - |\mathcal{C}_j| c_j$. Eq. (8) is our final derived VCG payment equation, and we have renamed the terms to P_1, P_2, P_3, and P_4 respectively for convenience.

B. The VCG Outcome

In the context of our overlay multicast problem, an outcome is the set of independent decisions made by each node to join a parent. More formally, an outcome $o = (o_1, \ldots, o_n)$ is a set of choices made by each node i (except the root) on which parent to join such that $o_i \in A_i$. Back to our example in Fig. 1, the outcome vector is $(o_p, o_a, o_b, o_c) = (r, p, p, p)$.

According to Eq. (1), the VCG outcome should maximize the system valuation. This means that any node who is interested in subscribing to a service should join the multicast tree *only when its entrance will increase the overall system valuation*:

$$\sum_j v_j - \sum_{j \neq i} v_j^{-i} > 0 \qquad (9)$$

This is called the *participation constraint*. If there are more than one outcome decision o_i that satisfies the participation constraint, the VCG outcome o_i^* is the one that maximizes the left-hand expression of this constraint. We call the VCG outcome o_i^* the *best parent* of i, and call the best alternate parent the *second-best parent* of i. Compared with Eq. (5), however, we notice that the left-hand expression in Eq. (9) is simply the utility u_i. Therefore, maximizing the left-hand expression in Eq. (9) is the same as maximizing u_i^+, where $u_i^+ = \max(u_i, 0)$.

In summary, if a node i can calculate p_i, it can simply add its valuation and payment to obtain its utility. Under the *strategic* node assumption, a node will join to the parent that maximizes u_i^+. It is possible that none of the potential parents of i satisfies the participation constraint. In this case, node i will choose not to subscribe to the multicast service. This is reasonable because strategic nodes have no incentive to subscribe if the utility is not positive.

C. Distributed Algorithm

Our objective is to design a practical and distributed application-layer algorithm that converges to the VCG outcome and VCG payment previously derived. Information regarding a node at one end of the overlay network takes time to propagate through the network to reach the other end. Therefore, there is an unavoidable, but finite, delay in receiving any information from a node who is at the far end of the network. However, as long as the overlay network parameters are *slowly varying*, *i.e.*, the network dynamics are sufficiently slow, our distributed algorithms should dynamically reconfigure the multicast topology. In other words, our distributed algorithms should ensure that the optimal multicast tree be formed *eventually* in a finite number of steps. In the following, we wish to find a distributed algorithm that will converge to the correct VCG payment in Eq. (8).

Suppose node i wants to calculate the payment p_i that it should receive. To do so, we must find a way to obtain each term in Eq. (8) in a distributed manner. The first term P_1 is the forwarding cost of the parent of i, and can be obtained easily with one simple message exchange with $\mathcal{P}_i$.

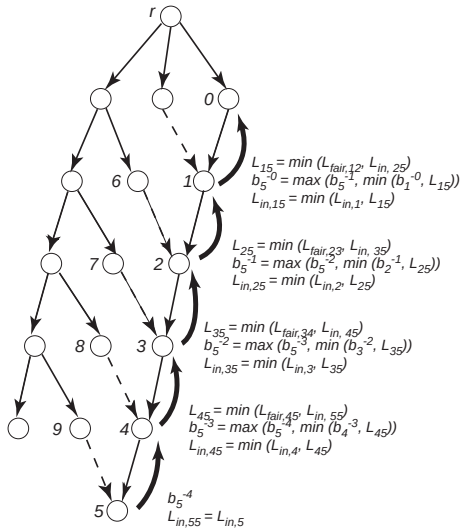


Fig. 3. Variable rate sessions: Distributed computation of updates

Calculating the second term P_2 in Eq. (8) is more challenging. Essentially, we need b_j and b_j^{-i} from every descendant $j \in \mathcal{GC}_i$. This can be achieved if every node periodically sends an update message to its corresponding parent regarding its private information b_j and b_j^{-i}. This implies that the message length is upper-bounded by $O(n)$, where n is the number of nodes in the network.

In the multicast tree shown in Fig. 3, node 5 should send node 4 the message $\{b_5, b_5^{-4}, L_{in,5}\}$, all of which are local information at node 5. Note that b_5 and b_5^{-4} are both used at node 4 to calculate p_4. Node 4 is then responsible for sending $\{b_4, b_4^{-3}, L_{in,4}, b_5, b_5^{-3}, L_{in,45}\}$ to node 3. b_4, b_4^{-3}, and $L_{in,4}$ are readily available at node 4, whereas b_5 is passed on from node 5. $L_{in,45}$ is a new notation used to denote the overall throughput constraint of the path from 4 to 5, and will be discussed in greater detail below. Finally, b_5^{-3} and $L_{in,45}$ needs to be computed, so we now proceed to find a way to derive each of them, purely from the local information at node 4.

How to calculate b_5^{-3} ***at node*** 4***?***

b_5^{-3} refers to the best throughput at which node 5 can receive provided that node 3 does not participate. From Fig. 3, it is clear that there are two alternate paths to consider. The first path goes through node 9 to reach node 5. The second path goes through node 8 and 4 to reach node 5. By inspection, we observe that the former path will deliver data to node 5 at the throughput of exactly b_5^{-4}. The second path will deliver data to node 5 at the throughput of b_4^{-3}, with additional constraints that link 45 may impose, which we call L_{45} for convenience. L_{45} is constrained by two factors: (1) the incoming throughput limit at node 5; and (2) the outgoing throughput limit (according to max-min fair allocation) at node 4, *i.e.*,

$$L_{45} = \min(L_{fair,45}, L_{in,5}) \tag{10}$$

Once L_{45} is found from Eq. (10), we take the minimum of b_4^{-3} and L_{45} to obtain the throughput that node 5 will receive at *if the path through node* 8 *and* 4 *is used to reach* 5. Finally, we find the better throughput out of the two paths that we have considered, as follows:

$$b_5^{-3} = \max(b_5^{-4}, \min(b_4^{-3}, L_{45})) \tag{11}$$

Eq. (11) is essentially a propagation equation that derives b_5^{-3} from b_5^{-4}. At this point, we have successfully calculated b_5^{-3}.

How to calculate $L_{in,45}$ ***at node*** 4***?***

$L_{in,45}$ is the overall throughput constraint of the path from 4 to 5, and includes three constraints: (1) the incoming throughput limit at node 5; (2) the outgoing throughput limit (according to max-min fair allocation) at node 4; and (3) the incoming throughput limit at node 4. Comparing L_{45} with $L_{in,45}$, we note that the only additional constraint of $L_{in,45}$ is the third constraint. We have:

$$\begin{aligned} L_{in,45} &= \min(L_{in,4}, L_{fair,45}, L_{in,5}) \\ &= \min(L_{in,4}, L_{45}) \end{aligned} \tag{12}$$

The second line in Eq. (12) simplifies the definition of $L_{in,45}$ using Eq. (10). We note that all calculations of $L_{in,45}$ can be done locally at node 4. First, $L_{in,4}$ is just local information readily available at node 4. Second, $L_{out,45}$ is found by max-min fair allocation algorithm performed at node 4. Third, $L_{in,5}$ is passed along from node 5. In general, we can define $L_{in,ik}$ to denote the overall throughput constraints of the path from node i to k, where $j \in \mathcal{C}_i$ is the child of i that is along the path from i to k. The general definition of $L_{in,ik}$ is therefore:

$$L_{in,ik} = \begin{cases} L_{in,i} & \text{if } i = k \\ \min(L_{in,i}, L_{fair,ij}, L_{in,jk}) & \text{if } i \neq k \end{cases} \tag{13}$$

Eq. (13) is a recursive definition in the sense that $L_{in,ik}$ is defined in terms of $L_{in,jk}$, where j is a child of i.

We illustrate the general `receive_update()` and `send_update()` algorithms in Table II, where j is a child of i, and k is a descendant in the subtree T_j. Due to space constraints, the interested reader is referred to our technical report [10] for an extensive example of this algorithm. Note that we have incorporated our propagation equations for finding b_5^{-3} and $L_{in,45}$ from Eq. (10), Eq. (11) and Eq. (12) on lines $13-15$. It can be verified that, this algorithm, which is used to calculate the second term of p_i from Eq. (8), is $O(n)$ as expected. Fig. 3 fills in the propagation steps all the way from leaf node 5 to node 1.

Up to this point, our discussion has focused on the second term P_2 in Eq. (8). The calculation of the third term P_3, as we will see shortly, is based on the same algorithms used to find the second term P_2 from Table II. To see how this is true, observe from Eq. (8) that P_2 differs from P_3 in that the former applies to the descendants of i having an alternative path for receiving multicast data, whereas the latter applies to the descendants of i *without* an alternative path for receiving multicast data. In the latter case, the throughput is $b_k^{-i} = 0$. As a result, if a descendant k of node i has $b_k^{-i} = 0$, it belongs to the third term P_3. Otherwise, it belongs to the second term P_2. Notice that in addition to passing up b_k, b_k^{-i},

```
1   void receive_update()
2       msgIn ← receiveMsg()
3       j ← msgIn.source
4       for each k ∈ T_j
5           b_k ← msgIn.b_k
6           b_k^{-i} ← msgIn.b_k^{-i}
7           |C_k| ← msgIn.|C_k|
8           c_k ← msgIn.c_k
9           L_{in,jk} ← msgIn.L_{in,jk}
10      end for
11      L_{fair,ij}^{-l} ← max-min(L_{out,i}, {L_{in,j∈C_i}})
12      for each k ∈ T_j
13          L_ik ← min(L_{fair,ij}, L_{in,jk})
14          b_k^{-P_i} ← max(b_k^{-i}, min(b_i^{-P_i}, L_ik))
15          L_{in,ik} ← min(L_{in,i}, L_ik)
16      end for
17
18  void send_update()
19      msgOut ← {b_i, b_i^{-P_i}, |C_i|, c_i, L_{in,i}}
20      for each k ∈ T_i
21          msgOut ← msgOut ∪ {b_k, b_k^{-P_i}, |C_k|, c_k, L_{in,ik}}
22      end for
23      sendMsg(msgOut,P_i)
```

TABLE II

VARIABLE RATE SCENARIO: DISTRIBUTED ALGORITHM TO RECEIVE AND SEND PERIODIC UPDATES

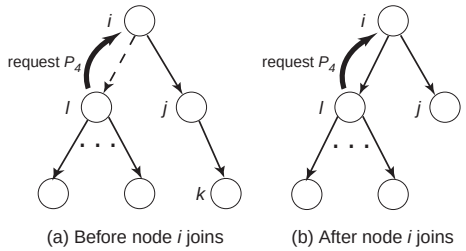


Fig. 4. Variable rate scenario: Node l requests P_4 from node i

and $L_{in,ik}$, we also need to pass $|\mathcal{C}_k|$ and c_k for the calculation of the third term, which are already included on line 21 in `send_update()`.

Finally, we proceed to find a distributed algorithm for the fourth term in Eq. (8), which calculates the change in valuation of nodes from other subtrees $\{T_{\mathcal{P}_i} \setminus T_i\}$. Suppose node i requests the assistance of $\mathcal{P}_i$ to calculate the fourth term P_4. This is presented with an example below.

In Fig. 4(a), consider the scenario where node j is the only child of node i, and node k is the only child of node j. Note that node l is not a child of node i yet, but suppose node l wishes to calculate the fourth term P_4 *if it were to join node* i. From our previous solution for the second and third terms, we should have already obtained the data set $\{b_j, b_j^i, |\mathcal{C}_j|, c_j, L_{in,jj}, b_k, b_k^i, |\mathcal{C}_k|, c_k, L_{in,jk}\}$ from the `receive_update()` method. For the example in Fig. 4(a), we assume these numeric values:

$$b_j = 100 \text{ KBps}, \quad b_j^{-i} = 50 \text{ KBps}$$
$$b_k = 90 \text{ KBps}, \quad b_k^{-i} = 70 \text{ KBps}$$

Note that all four throughputs above assume that l has not yet joined node i. Now suppose the max-min fair allocation algorithm determines that nodes j or k should each get a throughput of 60 KBps after l has joined, then the throughput to node j becomes $b_j^{+l} = 60$ KBps. The superscript $+l$ is used to distinguish between b_j and b_j^{+l}. While the former represents the throughput of j *before* l *joins* i, the latter represents the throughput of j *after* l *joins* i. The second-best throughput $b_j^{-i} = 50$ KBps is not good enough to motivate node j to leave node i. The change in benefit is thus $(m(b_j^{+l}) - (b_j)) = 60 - 100 = -40$. On the other hand, the throughput to node k after l has joined becomes $b_k^{+l} = 70$ KBps. This is because the joining of l to i has decreased the throughput of node k to the point that it decides to leave j. The change in throughput is thus $(m(b_k) - m(b_k^{-l})) = 70 - 90 = -20$.

Consider a very similar scenario in Fig. 4(b), but this time, node l is already a child of node i. For consistency, node k should no longer be the child of node j, since it has moved to the alternate parent, who can provide a throughput of 70 KBps. Again, node l wants to calculate P_4, and requests the assistance of node i. Node j would have the following numeric values:

$$b_j = 60 \text{ KBps}, \quad b_j^{-i} = 50 \text{ KBps}$$

Now, we need to calculate the change in throughput at node j if node l were to leave node i, *i.e.*, b_j^{-l}. From Fig. 4(b), we can see that $b_j^{-l} = 100$ KBps. Therefore, the change in throughput to node j is $(m(b_j) - m(b_j^{-l})) = 60 - 100 = -40$.

We summarize the fourth-term calculations by an algorithm that will be run on node i when a child or potential child l sends a request for P_4. This algorithm, which is also $O(n)$, is presented in Table III. Lines $7-15$ correspond to the scenario in Fig. 4(b), and lines $16-24$ correspond to Fig. 4(a). The interested reader is referred to our technical report [10] for an extensive example explaining this algorithm.

Including all four terms in calculating the payment to node i from Eq. (8), the final distributed VCG algorithm is presented in Table IV.

D. Implementation

To show that our proposed algorithms are practical, we implement the distributed algorithms in *iOverlay* [11], an experimental testbed for implementing and evaluating overlay protocols. The experiments are then deployed on PlanetLab [4].

When a node is first started, each node i is bootstrapped with a random set of neighbors, *i.e.*, A_i. These neighbors then become the set of potential parents that a node can join. Recall that each node i must find the best parent to join. In our protocol implementation, this is achieved by passing messages between neighbors or adjacent nodes. Each message in our protocol is identified by its message type. The mechanics of the tree formation and evolution is based on the join and rejoin process. The join process involves the InfoReq-InfoAck-JoinReq-JoinAck message sequence. Fig. 5(a) illustrates the message passing from a network view. When node i sends out an InfoReq message (labeled iR) to each of its five neighbors, both nodes a and $\mathcal{P}_i$ respond with InfoAck (labeled iA). However, nodes b, e, and j all respond with InfoNak (labeled iN). Node a will then decide whether to stay with the parent

```
1   void request_P4()
2     msgIn ← receiveMsg()
3     l ← msgIn.source
4     P4 ← 0
5     L_fair,ij ← max-min(L_out,i, {L_in,j∈{C_i}})
6     L_ik ← min(L_fair,ij, L_in,jk)
7     if l ∈ C_i        // Existing child
8       for each j ∈ {C_i \ l}
9         L^-l_fair,ij ← max-min(L_out,i, {L_in,j∈{C_i\l}})
10        for each k ∈ T_j
11          L^-l_ik ← min(L^-l_fair,ij, L_in,jk)
12          P4 ← P4 + (m(L_ik) − m(L^-l_ik))
13        end for
14      end for
15      msgOut ← {c_i, P4}
16    else if l ∉ C_i        // Potential new child
17      for each j ∈ C_i
18        L^+l_fair,ij ← max-min(L_out,i, {L_in,j∈{C_i∪l}})
19        for each k ∈ T_j
20          L^+l_ik ← min(L^+l_fair,ij, L_in,jk)
21          P4 ← P4 + (m(L^+l_ik) − m(L_ik))
22        end for
23      end for
24      msgOut ← {c_i, P4}
25    end if
26    sendMsg(msgOut,l)
```

TABLE III

VARIABLE RATE SCENARIO: DISTRIBUTED ALGORITHM TO CALCULATE THE FOURTH PAYMENT TERM

```
1   void VCG_payment()
2     p_i ← P_1 // First term
3     for each k ∈ T_i
4       if b_k^-i ≠ 0 // P_k^-i ∃, Second term
5         p_i ← p_i + (m(b_k) − m(b_k^-i))
6       else if b_k^-i = 0 // P_k^-i ∄, Third term
7         p_i ← p_i + (m(b_k) − |C_k|c_k)
8       end if
9     end for
10    p_i ← p_i + P_4 // Fourth term
```

TABLE IV

VARIABLE RATE SESSIONS: DISTRIBUTED VCG PAYMENT ALGORITHM

$\mathcal{P}_i$ or rejoin to the new parent a, according to a maximization of its private utility u_i^+, as described in Sec. IV-B. Fig. 5(b) illustrates the idea using a timing diagram. Node i first sends an InfoReq to its two neighbors, a and b. Node a, who is a valid potential parent of i, responds with an InfoAck. Node b, who is not a valid potential parent of i, responds with an InfoNak. Since only node a replies with an InfoAck, node i will initiate a JoinReq message to a. A JoinAck message is then replied by node a upon a successful join.

At an early stage of the multicast topology formation process, nodes that are topologically far away may not be able to find any neighbor who is already a subscriber, *i.e.*, all

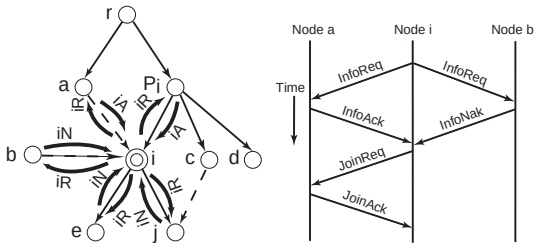


Fig. 5. Distributed Protocol Message Passing

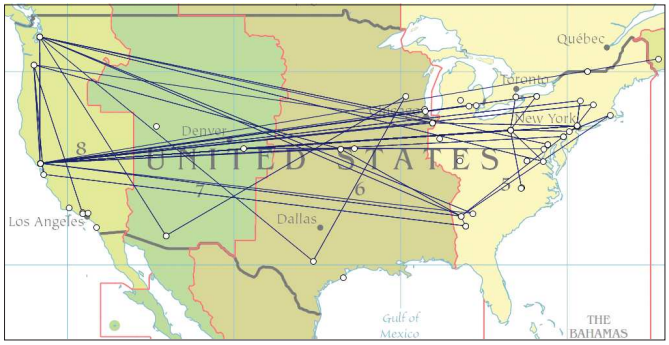


Fig. 6. 80 PlanetLab nodes distributed over North America

neighbors reply with an InfoNak. In contrast, nodes that are adjacent to the *data source*, or the *root*, will get an InfoAck from the root immediately, and the tree formation will begin. Every overlay node on the network who is interested in the multicast service will keep trying to join the multicast group by sending periodic InfoReq. Even after a node successfully joins the multicast group, it still has to periodically probe its neighbors to ensure that it is connecting to the best parent. When a node detects that there is a better parent than its current parent, it will initiate a rejoin process. The rejoin process includes two basic steps: a leave from the existing parent and a join to the new best parent.

We assume that every node is interested in the multicast service. This assumption does not compromise the generality of our experiment because a node who is not interested in a particular multicast service will simply reply with InfoNak whenever it receives an InfoReq, and therefore, will never participate in the formation of the multicast tree anyway.

We define the benefit function and cost to be:

$$m(b_i) = 5 \cdot b_i \tag{14}$$

$$c_i = 10\left(1 - \frac{1}{|\mathcal{C}_i| + 1}\right) \tag{15}$$

b_i in Eq. (14) is the inflow throughput of node i in KBps. Note that Eq. (14) satisfies both the non-negative and non-decreasing criteria of our benefit function. Further, Eq. (15) simulates a higher cost when the number of children increases.

Our experiments involve running 80 PlanetLab nodes distributed over North America in parallel. Fig. 6 presents the geographical distribution of these 80 PlanetLab nodes. The per-node inflow and outflow constraints are emulated by the iOverlay engine [11], and are generated with a power-law distribution over the range specified in Table V, which also summarizes other experimental parameters.

Parameter	Value or Range of Parameter
Number of overlay nodes	80
Inflow throughput limit $L_{in,i}$	$10 - 50$ KBps
Outflow throughput limit $L_{out,i}$	$20 - 100$ KBps
Data message size	50 KB
Buffer size at each node	1000 messages
Number of initial random neighbors	3
Frequency of throughput measurements	8 seconds
Frequency of periodic updates	6 seconds
Frequency of rejoin process	46 seconds

TABLE V

VARIABLE RATE SCENARIO: EXPERIMENTAL PARAMETERS

Fig. 7(a) further plots the number of KB received by each of the 80 nodes. Each line corresponds to the number of KB received by one node over time. We can see that the experienced data throughputs range from about 5 KBps to 35 KBps. Fig. 7(b) quantifies the percentage gain in terms of per-node throughput of our distributed algorithm (labeled *Variable rate VCG*) over a simple benchmark (labeled *Variable rate Random*). In every rejoin process of the *Random* scheme, instead of joining to the best neighbor, each node will choose a random neighbor to join to. We observe that each node generally experiences a higher throughput with our VCG-based algorithm than with the Random scheme. We can sum the throughputs of all nodes to obtain the total throughput in KBps. The total throughputs of the VCG scheme is 1054 KBps, comparing to a total throughput of only 626 KBps — a 68% throughput improvement. This result has confirmed the fact that our distributed protocol is converging to a more optimal multicast tree.

We further evaluate the correctness of our distributed algorithms and protocol implementations in Fig. 7(c), where we track both the *system valuation* and the *system utility* over time. Since the VCG outcome from Eq. (1) maximizes the system valuation $\sum_i v_i$, we expect the total valuation of all nodes to monotonically increase over time. Indeed, the solid line in Fig. 7(c) shows that the system valuation rises from an initial value of 0, and converges at approximately 6500, staying at that level for the rest of the experiment. The minor fluctuations is due to delayed or stale information that is inevitable in any distributed protocol. Overall, this result indicates that each node is indeed maximizing the *system valuation* when making individual decisions based on private utility — a vivid illustration of strategyproofness. In contrast, the system utility demonstrates no observable trend of convergence. This observation clarifies a common misconception: when every node in the network maximizes their private utility, the system utility of a VCG-based mechanism is not necessarily maximized.

In Fig. 7(d), the balance of each node is plotted as a function of time. We have left out the legend of the graph for brevity. From the figure, most subscribers experience a budget surplus (a positive balance) at the end of the experiment. The solid line is drawn from the perspective of the system as a whole, who experiences an overall deficit. This is an empirical illustration of the well-known *budget-balance* problem of VCG, which states that the system will run a budget deficit.

We end with an analysis on the overhead of our distributed protocol. As mentioned in our protocol design, a message can either be a control message or a data message. In our experiments, we measure both the data and control messages in KB, and graph the percentage overhead of each node in Fig. 8. We observe that most nodes experience an overhead of around $0.1 - 1\%$. Furthermore, there are a few nodes who have never joined the multicast tree, since they were unable to find a neighbor who satisfies the participation constraint. Consequently, these nodes have not received any data message, and so they have been left out of Fig. 8. Overall, the overhead of implementing our protocol in the entire overlay network is a modest 0.66%, as seen from the results summarized in Table VI.

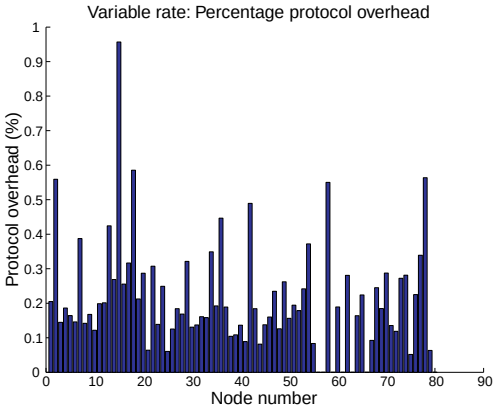


Fig. 8. Variable rate scenario: Percentage overhead of each node

Message Type	Number of KB
Control messages	685 KB
Data messages	103406 KB
Protocol Overhead	0.66%

TABLE VI
VARIABLE RATE SCENARIO: PROTOCOL OVERHEAD

V. THE SCENARIO OF SINGLE RATE SESSIONS

A. The VCG Payment

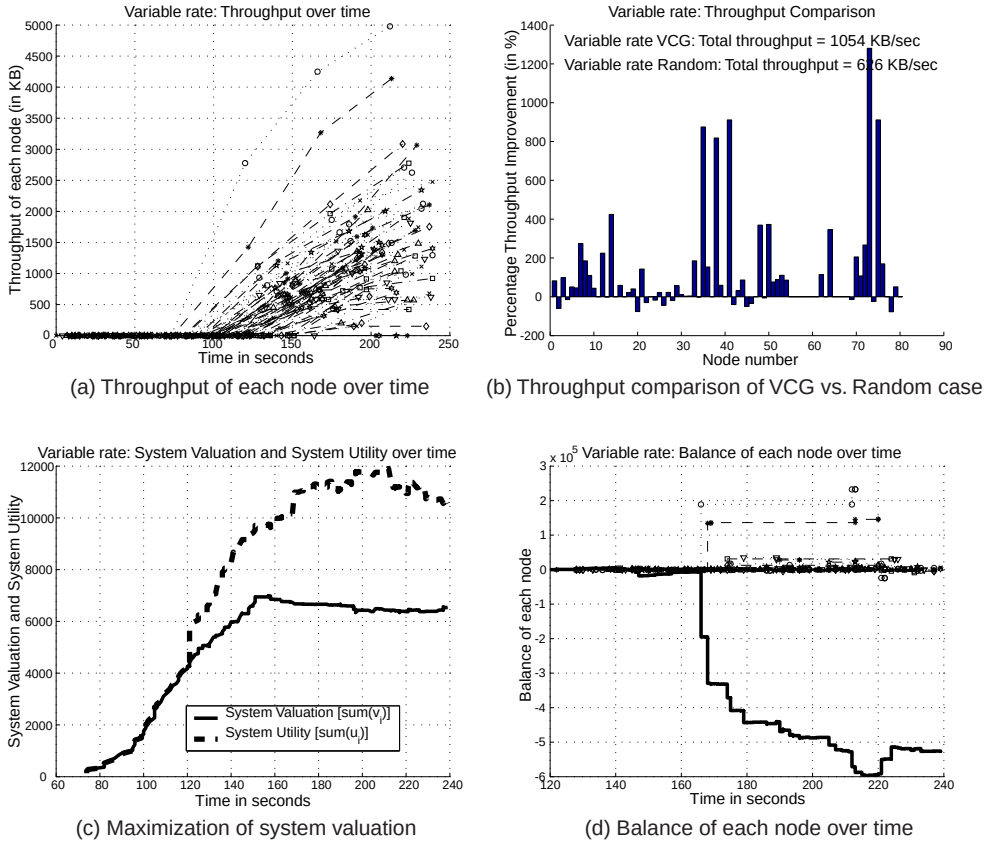
In the single rate scenario (Fig. 2(b)), we assume that every node in the network has no more buffer to queue messages. Messages that are received but not sent out to downstream children will immediately be dropped. Another way to interpret a filled buffer node i is that all network links incident at node i will have the same throughput in operation. If all nodes have a filled buffer, then the entire tree will operate at one uniform throughput in steady-state. We introduce the notation $b(T_i)$ to represent the uniform operating throughput of the entire subtree T_i. To calculate $b(T_i)$, we take the minimum throughput limit of every node in subtree T_i. In other words, L_{in} or L_{out} of one of the nodes in T_i will eventually become the *bottleneck* that determines the value of $b(T_i)$.

$$b(T_i) = \min\left(\min_{j\in T_i}\{L_{in,j}\}, \min_{j\in T_i}\left\{\frac{L_{out,j}}{|\mathcal{C}_j|}\right\}\right) \tag{16}$$

Again, we derive our VCG payment from Eq. (3). Similar to our derivations of payment in the variable rate scenario in Sec. IV-A, we wish to account for the *externality* of the participation of node i. Eq. (17) partitions the set of all nodes (except i) into four disjoint sets, namely the parent of i, the descendants of i who have a second-best parent, the descendants of i who do not have a second-best parent, and the set of all nodes other than those in T_i. Note that each term in Eq. (17) corresponds to an effect of the entrance of i.

$$\begin{aligned} p_i &= (v_{\mathcal{P}_i} - v^{-i}_{\mathcal{P}_i}) + \sum_{j\in\{\mathcal{GC}_i\cap\mathcal{P}_j^{-i}\exists\}} (v_j - v_j^{-i}) \\ &\quad + \sum_{j\in\{\mathcal{GC}_i\cap\mathcal{P}_j^{-i}\nexists\}} (v_j - v_j^{-i}) + \sum_{j\in\{T_r\setminus T_i\}} (v_j - v_j^{-i}) \end{aligned} \tag{17}$$

$$\begin{aligned} &= -c_{\mathcal{P}_i} + \sum_{j\in\{\mathcal{GC}_i\cap\mathcal{P}_j^{-i}\nexists\}} (m(b(T_r)) - |\mathcal{C}_j|c_j) \\ &\quad + \sum_{j\in\{T_r\setminus T_i\}} (m(b(T_r)) - m(b(T_r\setminus T_i))) \end{aligned} \tag{18}$$



(a) Throughput of each node over time

(b) Throughput comparison of VCG vs. Random case

(c) Maximization of system valuation

(d) Balance of each node over time

Fig. 7. Experimental results: the scenario of variable rate sessions

$= P_1 + P_2 + P_3$

For the parent of i, the participation of i will incur an extra forwarding cost $c_{\mathcal{P}_i}$. This explains the first term in Eq. (18). The second term in Eq. (17) is zero, because if a descendant of i can find a second-best parent, then the only difference in valuation will be caused by a difference in throughputs. However, we have already shown that there is only one operating throughput for the entire tree. Thus, the second term in Eq. (17) has been eliminated in Eq. (18). v_j^{-i} in the third term of Eq. (17) is zero because if a descendant of i cannot find a second-best parent, then there is no value. Therefore, we are left with $v_j = m(b(T_r)) - |\mathcal{C}_j| c_j$ in Eq. (18). Finally, for the fourth term in Eq. (17), the cost component of the valuation function will be cancelled out in Eq. (17), leaving only a difference in benefits $(m(b(T_r)) - m(b(T_r \setminus T_i)))$ in Eq. (18). Again, we have renamed the first, second, and third terms of the payment equation to P_1, P_2, and P_3 respectively for convenience.

P_3 in Eq. (18) merits some further discussion. In particular, it suggests that the valuation of nodes in other subtrees will be affected by the entrance of i only when T_i throttles the throughput of the existing tree $\{T_r \setminus T_i\}$. In summary, we have:

$$b(T_r) = \min\left(b(T_r \setminus T_i), b(T_i)\right) \tag{19}$$

B. The VCG Outcome

Similar to the Sec. IV-B, after node i has calculated its payment p_i, it can easily choose the best parent by maximizing its private utility $u_i = v_i + p_i$, subject to the participation constraint of Eq. (9).

C. Distributed Algorithm

We seek to find a distributed algorithm for the single rate scenario. The first term P_1 in Eq. (18) can be found by a simple message exchange between node i and its parent $\mathcal{P}_i$. The second term P_2 in Eq. (18) requires that we know the operating throughput $b(T_r)$, the number of children $|\mathcal{C}_j|$, and the forwarding cost c_j. The latter two pieces of data can simply be passed up from every child to its parent periodically as in our solution in the variable rate scenario, which requires a message length of $O(n)$. We further wish to calculate $b(T_r)$ in a distributed manner. But before we proceed, we first define a new notation L_i that combines the inflow and outflow constraints of node i into one constraint:

$$L_i = \begin{cases} L_{in,i} & \text{if } i \text{ is leaf} \\ \frac{L_{out,i}}{|\mathcal{C}_i|} & \text{if } i \text{ is root} \\ \min\left(L_{in,i}, \frac{L_{out,i}}{|\mathcal{C}_i|}\right) & \text{otherwise} \end{cases} \tag{20}$$

$b(T_r)$ can be calculated in a distributed manner if all nodes pass $b(T_i)$ to their respective parent, and each parent computes the propagation equation as follows:

$$b(T_i) = \min\left(L_i, \min_{j \in \mathcal{C}_i}\{b(T_j)\}\right) \tag{21}$$

The above equation is recursive, and suggests that we should find the minimum of the local L_i as well as the $b(T_j)$ for all children j. Each node stores two throughput numbers, namely $b(T_i)$ and $b(T_r)$. These two numbers are not necessarily the

```
1   void receive_update()
2       msgIn ← receiveMsg()
3       j ← msgIn.source
4       for each k ∈ T_j
5           b(T_k) ← msgIn.b(T_k)
6           |C_k| ← msgIn.|C_k|
7           c_k ← msgIn.c_k
8       end for
9       if P_i = null        // Node i is the root
10          L_i ← ( L_{out,i} / |C_i| )
11      else if |C_i| = 0        // Node i is a leaf
12          L_i ← L_{in,i}
13      else
14          L_i ← min( L_{in,i}, L_{out,i} / |C_i| )
15      end if
16      b(T_i) ← min( L_i, min_{j∈C_i} {b(T_j)} )
17
18  void send_update()
19      msgOut ← {b(T_i), |C_i|, c_i}
20      for each k ∈ T_i
21          msgOut ← msgOut ∪ {b(T_k), |C_k|, c_k}
22      end for
23      sendMsg(msgOut,P_i)
```

TABLE VII

SINGLE RATE SCENARIO: DISTRIBUTED ALGORITHM TO RECEIVE AND SEND PERIODIC UPDATES

```
1   void request_P3()
2    msgIn ← receiveMsg()
3    l ← msgIn.source
4    P_3 ← 0
5    if l ∈ C_i        // Existing child
6      for each j ∈ {C_i \ l}
7        for each k ∈ T_j
8          L_i^{+l} ← min( L_{in,i}, L_{out,i} / (|C_i|−1) )
9          b(T_r \ T_l) ← min(b(T_r \ T_i), L_i^{+l}, min_{j∈{C_i\l}} b(T_j))
10         P_3 ← P_3 + (m(b(T_r)) − m(b(T_r \ T_l)))
11       end for
12     end for
13     msgOut ← {c_i, P_3}
14   else if l ∉ C_i        // Potential new child
15     for each j ∈ C_i
16       for each k ∈ T_j
17         L_i^{+l} ← min( L_{in,i}, L_{out,i} / (|C_i|+1) )
18         b(T_r ∪ T_l) ← min(b(T_r \ T_i), L_i^{+l}, min_{j∈{C_i∪l}} b(T_j))
19         P_3 ← P_3 + (m(b(T_r ∪ T_l)) − m(b(T_r)))
20       end for
21     end for
22     msgOut ← {c_i, P_3}
23   end if
24   sendMsg(msgOut,l)
```

TABLE VIII

SINGLE RATE SCENARIO: DISTRIBUTED ALGORITHM TO CALCULATE THE THIRD PAYMENT TERM

```
1   void VCG_payment()
2       p_i ← P_1 // First term
3       for each k ∈ T_i
4           if b_k^{-i} = 0 // P_k^{-i} ∄, Second term
5               p_i ← p_i + (m(b(T_r)) − |C_k|c_k)
6           end if
7       end for
8       p_i ← p_i + P_3 // Third term
```

TABLE IX

SINGLE RATE SCENARIO: DISTRIBUTED VCG PAYMENT ALGORITHM

same, because $b(T_i)$ considers only the descendants of i, but $b(T_r)$ takes into account of the additional constraints from the rest of the multicast tree. However, these two numbers can be the same if the subtree T_i is the bottleneck of the throughput of the entire tree. Table VII presents the exact algorithm. Note that our propagation equation from Eq. (21) is found on line 16.

We also note that $b(T_i)$ at the root is identical to $b(T_r)$, *i.e.*, $b(T_r) = b(T_i)$. In this way, the root can calculate $b(T_r)$, and can then pass this value down to all descendants, either piggy-backing it on a multicast data message or using a separate control message. Finally, to calculate the third term in Eq. (18), node i can send a request to its parent. The exact algorithm is presented in Table VIII.

The final payment calculation is simply a sum of the three terms in Eq. (18), and is presented in Table IX. Note the similarity between the distributed algorithms of the variable and single rate scenarios. In fact, the structure of these two sets of algorithms only differs in the specific propagation equations used.

D. Implementation

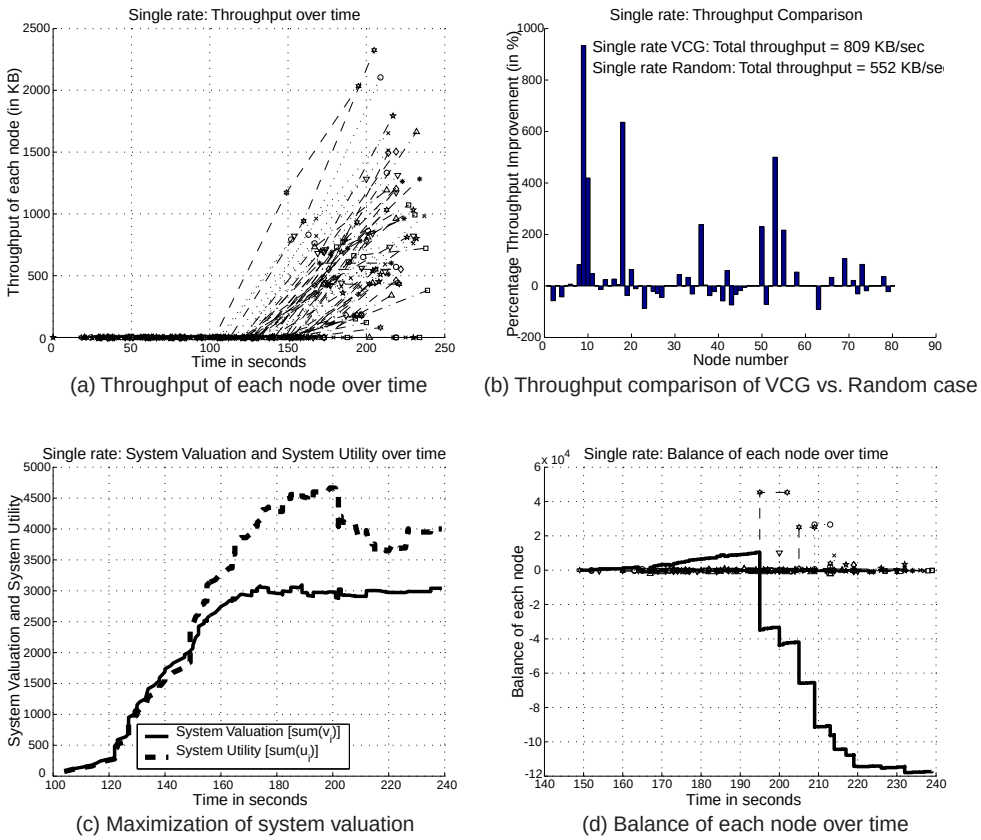
Since the structure of the distributed solutions are similar between the single and variable rate scenarios, we will use the same protocol designed in Sec. IV-D in this subsection. For ease of comparisons, we have kept most experimental parameters the same as before. In the single rate experiments, the buffer size of each node has been set to one message, instead of 1000 messages. Apparently, this is to emulate the single rate behavior. Further, the single rate distributed algorithms require extra control messages to broadcast the value of $b(T_r)$ to all subscribers, and this broadcast message is sent every 2 seconds in the current experiment.

Fig. 9(a) further plots the number of KB received by each of the 80 nodes. We can see that the experienced data throughputs range from about 3 KBps to 25 KBps. Fig. 9(b) compares the throughput performance of our VCG scheme with the Random scheme. The total throughput of all nodes in the VCG scheme is 809 KBps, representing a 47% improvement over the Random case, which has a total throughput of only 552 KBps. As in the variable rate experiments, this result indicates that our protocol is converging to a more optimal multicast tree.

In Fig. 9(c), we plot the system valuation and the system utility as a function of time, where the monotonic increase of system valuation over time can be readily observed, and convergence occurs at around 3000. On the other hand, the system utility displays no observable trend of convergence, as previously explained.

We next turn our attention to the balance of each node over time in Fig. 9(d). Once again, almost all multicast subscribers gain a positive balance over time, and the system experiences a deficit. These results are quite similar to the variable rate experiments, since both are VCG-based algorithms after all.

In a similar style as our results in the variable rate experiments, Fig. 10 graphs the percentage overhead of each PlanetLab node. We observe that most of the nodes experience an overhead percentage of $0.3 - 3\%$. Again, we have left out the nodes who have not found a neighbor satisfying the participation constraint, and thus have not joined to the multicast tree during our experiment. Another observation



(a) Throughput of each node over time

(b) Throughput comparison of VCG vs. Random case

(c) Maximization of system valuation

(d) Balance of each node over time

Fig. 9. Experimental results: the scenario of single rate sessions

is that the message overhead of the single rate scenario is generally larger than that of the variable rate scenario. This is a reasonable result, as explained next.

Table X calculates the overhead of our distributed protocol for the single rate scenario to be 1.44%, which is higher than the corresponding overhead of 0.66% for the variable rate scenario. This increase in overhead is contributed by: (1) an increase in the number of control messages; and (2) a decrease in the number of data messages. In the single rate scenario, the root has to consistently broadcast the multicast tree throughput $b(T_r)$ to every subscriber. These periodic broadcast messages, which are sent every 2 seconds to all subscribers, account for the increase in control messages. In addition, the system as a whole receives only 56635 KB of multicast data in the single rate experiments, in contrast with 103406 KB of multicast data received in the variable rate experiments. Therefore, the whole network is able to receive 83% more multicast data in the variable rate scenario than the single rate scenario — a result that we already argued intuitively at the end of Sec. III. Fig. 11 graphs the percentage throughput improvement of the variable rate scenario over the single rate scenario. From this figure, it is clear that the nodes in the variable rate scenario generally experience a higher throughput than the nodes in the single rate scenario. In summary, although the message overhead is higher than before, the resulting protocol still runs smoothly in our 80-node experiments.

Overall, we believe that the results of our implementations on PlanetLab have verified some of the important properties of our VCG-based distributed protocol. Our protocol has also been evaluated in terms of its convergence to optimality,

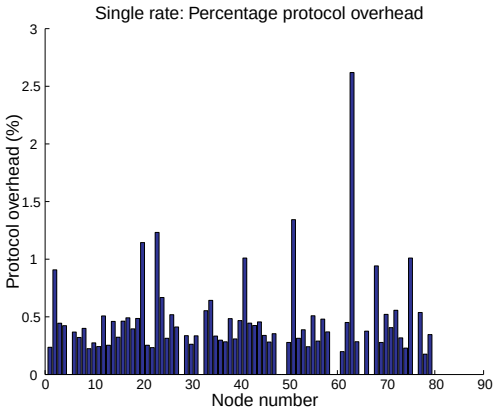


Fig. 10. Single rate scenario: Percentage overhead of each node

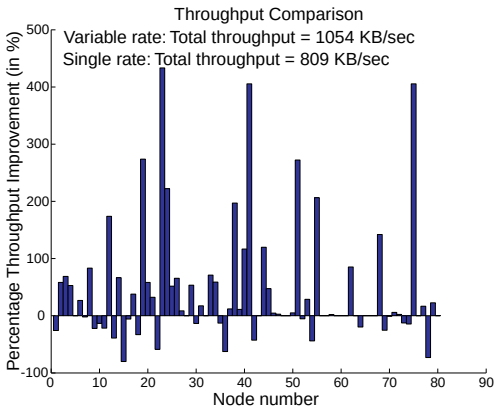


Fig. 11. Throughput comparison of variable vs. single rate scenario

Message Type	Number of KB
Control messages	830 KB
Data messages	56653 KB
Protocol Overhead	1.44%

TABLE X

SINGLE RATE SCENARIO: PROTOCOL OVERHEAD

throughput performance, as well as efficiency. Further, we have not only derived a theoretical VCG-based strategyproof mechanism, but have also extended this theoretical solution to a set of distributed algorithms that will converge to the correct VCG payments in a finite number of steps. Most importantly, the designed protocol that implements these distributed algorithms have also been proven to converge to the overlay multicast tree that maximizes system valuation.

VI. RELATED WORK

In their seminal work, Nisan and Ronen [6] first explored the application of mechanism design to networking problems. They applied the VCG mechanism to the shortest-path routing problem, and subsequently raised a few issues on the computational complexity of the problem. In [7], Feigenbaum *et al.* further reshaped the research agenda and suggested the need of finding a distributed solution to the mechanism design problem, which is called *Distributed Algorithmic Mechanism Design* (DAMD). Parkes *et al.* [12], [13] have also applied the theory of mechanism design to a different problem domain on combinatorial auction. Overall, the theoretical basis of our work is closest in spirit to [9], [14].

Feigenbaum *et al.* [9] applied mechanism design to solve the IP multicast problem in a distributed manner, but with a greater inclination towards theoretical complexity issues. In contrast, we have a more concrete network model and a more realistic problem formulation involving single and variable rate sessions. At the same time, we successfully designed distributed algorithms in each scenario that converge to the VCG solution. Sufficient detail of our distributed algorithms have also been provided that allows a direct implementation on PlanetLab, comparing to the more abstract and theoretical treatment from [9]. Overall, the extensive implementation results presented here have justified the validity of our protocol and mechanism design, which, to our knowledge, has not been accomplished before.

Woodard *et al.* [14] also uses the VCG mechanism to set up the problem of network formation, but with little attention given to formulating a *reasonable* valuation function for a specific scenario. Furthermore, the paper has made no attempt in solving the VCG mechanism in a distributed manner and has not performed any simulations or implementation. Nonetheless, their work provides an interesting perspective of *dynamic* and *sequential* mechanisms, which will serve as a good complement to our current work.

VII. CONCLUDING REMARKS

In this paper, we have solved for a *strategyproof* mechanism for overlay multicast tree formation in both the variable and single rate scenarios. Distributed algorithms have been presented and shown to converge to the global network optimal of maximum system throughput and minimum message forwarding costs. In both solutions, participation of each node is voluntary, so that a node may leave the multicast tree whenever its private utility becomes negative. Thus, every node is entitled to a non-negative utility, and the right amount of incentives will be given to reveal truthful private information. More importantly, not only have we found distributed algorithms to converge towards the VCG solution, but we have also carefully designed a protocol verified by actual implementation in PlanetLab. In summary, we believe that we have narrowed the gap between theory and practice, and have brought the research community one step closer to the actual deployment and realization of these VCG-based strategyproof mechanisms.

REFERENCES

[1] A. Mas-Colell, M. Whinston, and J. Green, *Microeconomic Theory*, chapter 23, Oxford University Press, New York, 1995.

[2] J. Byers and J. Considine, "Informed Content Delivery Across Adaptive Overlay Networks," in *Proc. of ACM SIGCOMM*, August 2002.

[3] D. Kostic, A. Rodriguez, J. Albrecht, and A. Vahdat, "Bullet: High Bandwidth Data Dissemination Using an Overlay Mesh," in *Proc. of ACM SOSP*, 2003.

[4] L. Peterson, D. Culler, T. Anderson, and T. Roscoe, "A Blueprint for Introducing Disruptive Technology into the Internet," in *Proc. of HotNets-I*, Princeton, NJ, October 2002.

[5] M.J. Osborne and A. Rubinstein, *A Course in Game Theory*, chapter 1-3,6,8,10, The MIT Press, Cambridge, Massachusetts, 2002.

[6] N. Nisan and A. Ronen, "Algorithmic Mechanism Design," *Games and Economic Behavior*, vol. 35, pp. 166–196, 2001.

[7] J. Feigenbaum and S. Shenker, "Distributed Algorithmic Mechanism Design: Recent Results and Future Directions," in *Proc. of ACM Dial-M*, Atlanta, Georgia, September 2002.

[8] J. Feigenbaum, C. Papadimitriou, R. Sami, and S. Shenker, "A BGP-based Mechanism for Lowest-Cost Routing," in *Proc. of ACM PODC*, New York, 2002, pp. 173–182, ACM Press.

[9] J. Feigenbaum, C. Papadimitriou, and S. Shenker, "Sharing the Cost of Multicast Transmissions," *Journal of Computer and System Sciences*, vol. 63, no. 1, pp. 21–41, 2001.

[10] Selwyn Yuen and Baochun Li, "Strategyproof Mechanisms for Dynamic Multicast Tree Formation in Overlay Networks," Tech. Rep., University of Toronto, http://iqua.ece.toronto.edu/papers/vcg-overlay.pdf, 2004.

[11] B. Li, J. Guo, and M. Wang, "iOverlay: A Lightweight Middleware Infrastructure for Overlay Application Implementations," in *Proc. of ACM/USENIX Middleware 2004, Toronto, Canada*, 2004.

[12] J. Kalgnanam D.C. Parkes and M. Eso, "Achieving Budget-Balance with VCG-Based Payment Schemes in Combinatorial Exchanges," in *IBM Research Report RC 22218*, March 2002.

[13] C. Ng, D.C. Parkes, and M. Seltzer, "Virtual Worlds: Fast and Strategyproof Auctions for Dynamic Resource Allocation," in *ACM Conference on Electronic Commerce*, 2003.

[14] C. J. Woodard and D. C. Parkes, "Strategyproof Mechanisms for Ad Hoc Network Formation," in *1st Workshop on the Economics of P2P systems*, 2003.