Defending Against Distance Cheating in Link-Weighted Application-Layer Multicast

Dan Li, *Member, IEEE, ACM*, Jianping Wu, *Senior Member, IEEE*, Jiangchuan Liu, *Senior Member, IEEE*, Yong Cui, and Ke Xu, *Senior Member, IEEE, Member, ACM*

***Abstract*—Application-layer multicast (ALM) has recently emerged as a promising solution for diverse group-oriented applications. Unlike dedicated routers in IP multicast, the autonomous end-hosts are generally unreliable and even selfish. A strategic host might cheat about its private information to affect protocol execution and, in turn, to improve its individual benefit. Specifically, in a link-weighted ALM protocol where the hosts measure the distances from their neighbors and accordingly construct the ALM topology, a selfish end-host can easily intercept the measurement message and exaggerate the distances to other nodes, so as to reduce the probability of being a relay. Such distance cheating, rarely happening in IP multicast, can significantly impact the efficiency and stability of the ALM topology. To defend against this kind of cheating, we present a Vickrey–Clarke–Groves (VCG)-based cheat-proof mechanism in this paper. We demonstrate a practical mapping from the utility, payment, and welfare of a VCG mechanism to the link-weighted ALM context. Based on this, we further discuss practical issues for implementing the cheat-proof mechanism—specifically, a trustworthy distributed algorithm for payment computation. Performance analyses show that the overheads of the computation, storage, and communication of our implementation are controlled at low levels, and extensive simulations further testify the implementation's effectiveness. Although there are other similar studies in this area, the contribution of our cheat-proof mechanism and its implementation primarily lies in two aspects. On one hand, we first explicitly solve the distance cheating problem in link-weighted ALM since its proposal by mapping the VCG mechanism to link-weighted ALM context. On the other hand, our distributed implementation can not only effectively defend against distance cheating, but can also avoid the potential cheating behaviors when selfish ALM nodes fulfill the cheat-proof mechanism itself.**

***Index Terms*—Application-layer multicast (ALM), cheat-proof, distance cheating, Vickrey–Clarke–Groves (VCG) mechanism.**

I. Introduction

MULTICAST is an effective vehicle for such group communications as media broadcasting, video conferencing, online gaming, and distance learning. IP multicast, which is implemented in the network layer, is probably the most efficient for the Internet. The deployment of IP multicast, however, remains limited nowadays due to many practical and political concerns, e.g., the lack of incentives to install multicast-capable routers and to carry multicast traffic. Recently, application-layer multicast (ALM) has emerged as a promising alternative [3]–[8]. ALM builds an overlay network out of unicast tunnels across cooperative participating end-hosts; each host is both a receiver and an application-layer router, and multicast is then achieved through data relaying among these nodes.

The application-layer multicast is more readily deployable and flexible than the network-layer solution with dedicated multicast routers. However, the end-hosts are autonomous application-layer nodes, which can join or leave the session at will or easily crash. Hence, efficiency and stability has been critical concerns in ALM topology construction. Even worse, an end-host can be selfish and strategic, which might affect the topology construction to improve its individual benefit. As an example, consider a typical link-weighted ALM protocol, in which each host measures the distances from its neighbors and the ALM topology is accordingly constructed. Given that the operations are in the application-layer, a selfish end-host may easily intercept the measurement message and exaggerate the distance to other nodes, so as to reduce the probability of being a relay. Such distance cheating, rarely happening in IP multicast, can significantly impact the efficiency and stability of an ALM topology [2], [9].

While the negative impact of distance cheating has been carefully investigated in previous studies, there is no explicit solution to defending against it yet. In this paper, we present an effective cheat-proof mechanism for link-weighted ALM, which is motivated by the Vickrey–Clarke–Groves (VCG) mechanism [10]–[12]. We demonstrate a mapping from the utility, payment, and welfare to the link-weighted ALM context. Given this mapping, we discuss a set of practical issues for implementing the mechanism design—specifically, a trustworthy distributed algorithm for payment computation. Our cheat-proof mechanism and its implementation do not depend on specific optimization objectives of the ALM topology construction, and hence can serve a broad range of protocols. Performance analyses show that the overheads of the computation, storage, and communication of our implementation are all controlled at low levels. Extensive simulations are further conducted under diverse network configurations, and the results demonstrate that, in all the configurations, the welfare of an individual receiver is maximized only when it tells the true distance.

Manuscript received July 28, 2009; revised July 23, 2010; accepted February 02, 2011; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor Z. M. Mao. Date of publication March 07, 2011; date of current version October 14, 2011. This paper was supported by the NSFC Project (60873252, 60970104, 60873253) and 973 Project of China (2009CB320501, 2009CB320503, 2011CB302900).

D. Li, J. Wu, Y. Cui, and K. Xu are with the Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China (e-mail: tolidan@gmail.com; jianping@cernet.edu.cn; cy@csnet1.cs.tsinghua.edu.cn; xuke@csnet1.cs.tsinghua.edu.cn).

J. Liu is with the School of Computing Science, Simon Fraser University, Vancouver, BC V6B 5K3, Canada (e-mail: jcliu@cs.sfu.ca).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNET.2011.2118230

We acknowledge that many similar studies are conducted in this area, but the contribution of our cheat-proof mechanism and its implementation primarily lies in two areas. First, we explicitly solve the distance-cheating problem in link-weighted ALM since its proposal by mapping the VCG mechanism to link-weighted ALM context. Second, our distributed implementation can not only effectively defend against distance cheating, but can also avoid cheating behaviors when selfish ALM nodes are fulfilling the cheat-proof mechanism, which is usually not considered by other studies.

II. Related Work

There have been significant studies on IP multicast in the past decade [1]. While it would be the most efficient vehicle, its scope and reach remain confined due to many practical and political issues. ALM thus has emerged as a promising alternative. Typical ALM examples include NARADA [3], NICE [4], and ZIGZAG [5], which, as in IP multicast, generally maintain a tree-structure for data delivering. Enhancements using advanced layered or multiple-description coding or using advanced mesh structures have also been introduced, e.g., SplitStream [6], Bullet [7], DONet [8], etc. , yet the tree structure remains an important building block in these systems.

The participating nodes in ALM are unreliable end-hosts, which are quite different from the dedicated multicast routers. The studies listed previously have addressed many of the efficiency and stability issues with the dynamic end-hosts. In this paper, we consider an orthogonal problem arising with the selfishness of end-hosts. That is, they might cheat about their private information, such as outgoing bandwidth, link cost, and available data, so as to receive multicast data with higher quality or bear less forwarding burden [2], [15].

The community has shown consistent interest in the study on user selfishness in ALM during the past years. Li *et al.* proposed defensive mechanisms against buffer map cheating in mesh-based ALM [30]. They also studied the impact of user selfishness during the construction-action stage of the data topology formation [31]. Xiao investigated cheating and anti-cheating in gossip-based protocols largely used in ALM [32]. Kitayama further studied the ALM host cheating in wireless environment [33].

We focus on distance cheating in link-weighted ALM in particular. Mathy *et al.* [2] showed that such a kind of simple cheating can negatively impact link stress and stretch of ALM trees. Li *et al.* [9] further demonstrate that the tree stability can be noticeably affected as well. Unfortunately, they do not offer practical solutions to prevent the cheating.

In this paper, we present an effective solution to defend against distance cheating in link-weighted ALM, which is motivated by the VCG mechanism. A similar mechanism has also been used by Yuen *et al.* [13] in single-rate and variable-rate ALM. Their focus, however, is on defending against throughput cheating, and the payment to each node is computed by the node itself, which does not address the problem if cheating happens during payment reporting. On the other hand, Wang *et al.* [14] study the link-cost cheating in general noncooperative multicast protocols. The optimal routing problem in the general multicast context is a Steiner tree problem, which is known to be NP-hard [16], [17]. In this context, they show that no efficient mechanisms can enforce selfish agents to correctly implement a payment scheme. In ALM, since all the participating nodes are receivers as well as relays, the Steiner tree problem turns into the much easier Minimum Spanning Tree problem. As a result, the VCG mechanism could be applied. Nevertheless, the mapping and implementation are nontrivial, especially the payment quantification and distributed computation. We will show that in our study.

III. Background and System Model

A. Distance Cheating in Link-Weighted ALM

We consider a general ALM model in which all the nodes are autonomous end-hosts. The total number of nodes, excluding the source, is n. Given the potentially large scale of the multicast session, we assume that each node has only a partial view of the members in the session, referred to as its neighbors. When a node joins the ALM session, it is assigned with neighbors by a central node (e.g., the source node or the node designated by the source node). The neighborships among the nodes form an ALM control topology, in which the average number of neighbors of each node over the total number of other participating nodes is the neighbor density, denoted by e. In link-weighted ALM, each receiver[1] asks its neighbors about their respective distances from the source, referred to as source-to-end distances, and measures its own distances from each neighbor. Given such distance information, the node selects one parent from its neighbors. The resultant topology for data delivering is in general a tree, which is called the ALM tree.

Note that each ALM receiver not only benefits from receiving data, but also suffers from forwarding data to its children. Therefore, a selfish receiver has the motivation to interfere with the distance measurement, so as to contribute less in forwarding but still enjoy good receiving quality. The interfering methods can be various, and we use a simple example to illustrate it. Consider a typical PING-resembled example, in which node j measures its distance from neighbor k by probing k. An honest k may reply the probe immediately, and the round-trip time (RTT) perceived by node j is thus a relatively accurate distance measure for delay-sensitive applications. A strategic k, however, may delay the reply, so as to exaggerate the distance, which in turn reduces the possibility of being selected as a parent. This can be viewed as distance cheating by node k. In our paper, we use a 4-tuple $C(n, e, m, h)$ to model such a kind of distance cheating, where m is the percentage of cheating receivers over all receivers, and each of them increases the actual distance with a cheating degree h. It is worth noting that our model is not confined to this additive operation; it can be easily extended to accommodate multiplicative cheating, i.e., multiply the actual distance by h.

Fig. 1 shows a simple example of the impact of distance cheating with $C(n, e, m, h) = (7, 46.4\%, 42.9\%, 4)$. Fig. 1(a) is the ALM tree constructed based on the actual distances with no cheating. In Fig. 1(b), nodes a, c, and d all exaggerate their distances to other nodes by increasing the actual values by 4. The source-to-end distance of node c then changes from 3 to 6, that of node f changes from 5 to 6, and that of node g changes from 7 to 8. We can see that such cheating substantially changes

[1]In this paper, we use "node" and "receiver" interchangeably, given the context is clear. Note, however, that "nodes" include the source as well.

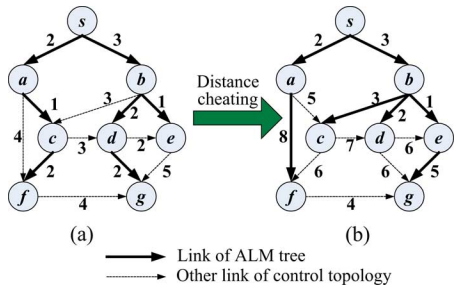

Fig. 1. Illustrative example of distance cheating. (a) ALM tree built without distance cheating. (b) ALM tree built with distance cheating.

the structure of ALM tree, which is clearly suboptimal as compared to that in Fig. 1(a).

Existing studies have also shown that, compared to the optimal ALM tree constructed with honest receivers, the link stress and the stretch will both increase under distance cheating [2]. In addition, since the cheating receivers would benefit from the misbehaviors, they tend to continuously declare wrong information, making the ALM tree unstable [9]. Therefore, distance cheating can significantly degenerate the efficiency and stability of an ALM tree. We address this problem through implementing a well-designed cheat-proof mechanism. Before we proceed on the details of the cheat-proof mechanism and its implementation, we first give an overview of the general algorithmic mechanism design and the VCG mechanism, which has been used in a wide range of computer network problems [19]–[25].

B. Overview of Algorithmic Mechanism Design and VCG Mechanism

Consider a public system consisting of n agents, $\{1,2,3,\ldots,n\}$. Each agent $i(\in \{1,2,3,\ldots,n\})$ has some private information, called its private type t_i, and $t=(t_1,t_2,\ldots,t_n)$ denotes the private types of all the agents. The types are drawn from a publicly known set T, but t_i is known by agent i only. The agents can be selfish and work strategically. In particular, when agent i is required to declare its private type to the public system, it may declare $s_i(\in T)$, referred to as its strategy, which is not necessarily equal to t_i.

The outcome o of the public system is a function of the strategies of all the agents, $o=F(s)$, where $s=(s_1,s_2,\ldots,s_n)$. We also use $s^{-i}=\{s_1,\ldots,s_{i-1},s_{i+1},\ldots,s_n\}$ to represent the strategy set of all the agents but i.

Each agent i wants to maximize its utility u_i, which depends on the outcome o of the public system and its own type t_i. Given that $o=F(s)$, the utility function can be written as $u_i=U_i(F(s),t_i)$. Note that agent i itself could achieve a higher utility by setting s_i differently from t_i, though the outcome of the system would be negatively impacted. The algorithmic mechanism design addresses this cheating by introducing a payment function P. The function defines a payment policy $p_i=P_i(s)$ for each agent i: A positive p_i means agent i should be paid, while a negative p_i means it should be charged. The welfare of agent i thus depends not only on the utility, but also its payment, i.e., welfare $w_i=u_i+p_i$, or $W_i(s)=U_i(F(s),t_i)+P_i(s)$, which now becomes the objective function for the agent to maximize. In a cheat-proof mechanism, each agent will achieve its maximum welfare only if it tells its true type; that is, the following definition applies.

Definition 1: A mechanism $M=(F,P)$ is cheat-proof

$$\forall t\in T^n,\Big[\forall i,\forall s_i\in T,\forall s^{-i}\in T^{n-1},\; W_i(s^{-i},t_i)=U_i(F(s^{-i},t_i),t_i)+P_i(s^{-i},t_i) \geq W_i(s^{-i},s_i)=U_i(F(s^{-i},s_i),t_i)+P_i(s^{-i},s_i)\Big].$$

Meanwhile, we expect to improve the efficiency of the system according to the following definition.

Definition 2: A mechanism $M=(F,P)$ is efficient if and only if it maximizes

$$\sum_{i=1}^{n}U_i(F(s),t_i).$$

The VCG mechanism [10]–[12] is a well-studied efficient cheat-proof mechanism, and our study is motivated by VCG as well.

Definition 3: A mechanism $M=(F,P)$ belongs to the VCG family if it satisfies

$$\forall t\in T^n,\left[\forall s\in T^n,F\in\arg\max_{F}\sum_{i=1}^{n}U_i(F(s),t_i)\right] \quad (1)$$

$$\forall t\in T^n,\Big[\forall s\in T^n,P_i(s)=\sum_{j\neq i}U_j(F(s),t_j)-\sum_{j\neq i}U_j(F(s^{-i}),t_j)\Big]. \quad (2)$$

The above two equations respectively define the VCG outcome and payment functions. Equation (1) suggests that, given the declared type set s, the VCG outcome function $F(.)$ should maximize the total utility of all the agents. Equation (2) suggests that, given the optimal outcome function $F(.)$, the payment to agent i is the total utility of all other agents when agent i participates in the system, minus that when agent i withdraws from the system.

Given the above utility and payment functions, the welfare of agent i in the VCG mechanism is computed as follows:

$$\forall t\in T^n,\Big[\forall s\in T^n,W_i(s)=U_i(F(s),t_i)+P_i(s) =\sum_{j=1}^{n}U_j(F(s),t_j)-\sum_{j\neq i}U_j(F(s^{-i}),t_j) =u-u^{-i}\Big]. \quad (3)$$

We can see from (3) that the welfare of each agent i is the total utility u of all agents when agent i participates in the system, minus the total utility u^{-i} of all other agents when i withdraws from the system. It is obvious that agent i cannot influence the value of u^{-i}. Therefore, in order to maximize its own welfare, it should seek to maximize the total utility u of the public system. Note that u is maximized when all agents tell the truth. Hence, a rational agent i has no motivation to cheat, and the mechanism is thus cheat-proof. More details, as well as formal proofs for the VCG mechanism, can be found in [26].

TABLE I
LIST OF NOTATIONS

n	Total number of nodes in ALM session (excluding source)
e	Neighbor density of control topology
m	Percentage of cheating receivers over all receivers
h	Cheating degree (added value to actual distance)
q_i	Parent of node i
D_i	Set of descendants of node i
T_i	Set of nodes in the sub-tree rooted at node i. $T_i = D_i \cup \{i\}$
X_i	Children set of node i
X_j^{-i}	Children set of node j after node i leaves
A^{-i}	Set of nodes having an alternative path to receive data when node i leaves
$\overline{A^{-i}}$	Set of nodes having no alternative path to receive data when node i leaves
l_{ij}	Distance from node i to node j
d_i	Source-to-end distance of node i
d_j^{-i}	Source-to-end distance of node j when node i leaves
c	Cost of forwarding data to a child
$B(d_j)$	Benefit of node j from receiving multicast data with source-to-end distance d_j

IV. CHEAT-PROOF MECHANISM AGAINST DISTANCE CHEATING

There are, however, some challenges to apply the VCG mechanism in the link-weighted ALM context. We should quantify the utility, the payment, and the welfare of each ALM node in this context, especially defining the computable payment. Also, we need to design a practical algorithm to implement the mapping. The algorithm is preferably distributed with low computation, storage, and communication overheads. In this section, we show such a mapping for the distance-cheating context, and its implementation is presented in Section V. For ease of our exposition, Table I lists the major notations, some of which have already been used in Section III.

A. Utility

In the link-weighted ALM context, the utility of node i can be evaluated as its benefit from receiving the multicast data, minus the cost of forwarding the data to its children. Clearly, its benefit depends on its source-to-end distance, and the cost depends on the number of its children, $|X_i|$. Let $B(d_i)$ represent the benefit of node i from receiving multicast data with a source-to-end distance of d_i, and c be the cost of forwarding data to a child.[2] We have

$$u_i = B(d_i) - |X_i|c \tag{4}$$

[2]In practice, the forwarding costs of different ALM nodes might be different, but in this paper we take them as identical for simplicity.

where benefit $B(\cdot)$ is a nonincreasing function of d_i.

We then consider the following mapping. The private information for each node i in link-weighted ALM is the vector of its actual distances to its neighbors, and its strategy is the measured distance vector by its neighbors. Each receiver selects the parent based on the strategy set s of all the participating nodes. The parent-selection policy is to select the neighbor through which it has the least source-to-end distance, which gives the outcome function $F(.)$, and the resultant ALM tree thus becomes outcome o.

B. Payment Policy

In the VCG payment function (2), the payment of each node i is the difference between the total utility excluding node i's with and without the participation of node i. To evaluate the payment, we classify all the receivers in an ALM tree into three sets and discuss their respective utility changes after node i leaves the multicast session.

Set 1) $D_i \cap A^{-i}$, in which each receiver j satisfies the following two conditions: 1) it is a descendant of node i; and 2) when node i leaves, it still has an alternative parent to receive the multicast data. Both its benefit and cost might change after node i's leave, resulting in a utility change of $B(d_j) - B(d_j^{-i}) - (|X_j| - |X_j^{-i}|)c$.

Set 2) $D_i \cap \overline{A^{-i}}$, in which each receiver j satisfies condition 1) above, but not 2). Since it cannot join the multicast session after node i's leave, its utility after the leave of node i is 0, giving a utility change of $B(d_j) - |X_j|c$.

Set 3) $\overline{T}_i$, which includes the receivers not belonging to the subtree rooted at node i. The benefit of a receiver j in this set will not change after node i leaves, but its forwarding cost may increase if it has to serve as the alternative parent for the nodes in Set 1. It is easy to show that the utility change is $(|X_j^{-i}| - |X_j|)c$.

In summary, the payment of node i can be evaluated as follows:

$$\begin{aligned} p_i &= \sum_{j \in D_i \cap A^{-i}} (u_j - u_j^{-i}) + \sum_{j \in D_i \cap \overline{A^{-i}}} (u_j - u_j^{-i}) \\ &\quad + \sum_{j \in \overline{T_i}} (u_j - u_j^{-i}) \\ &= \sum_{j \in D_i \cap A^{-i}} [B(d_j) - B(d_j^{-i})] + \sum_{j \in D_i \cap \overline{A^{-i}}} B(d_j) \\ &\quad + \left(\sum_{j \in (D_i \cap A^{-i}) \cup \overline{T_i}} |X_j^{-i}| - \sum_{j \neq i} |X_j| \right) c. \end{aligned} \tag{5}$$

Given the total number of nodes (including the source and node i) is $n+1$ in the ALM tree, we have

$$\sum_{j \neq i} |X_j| + |X_i| = n. \tag{6}$$

After the leave of node i, the total number of nodes becomes n, which follows that

$$\sum_{j \in (D_i \cap A^{-i}) \cup \overline{T_i}} |X_j^{-i}| + |D_i \cap \overline{A^{-i}}| = n - 1. \tag{7}$$

Combining (5)–(7), the payment of node i becomes

$$\begin{aligned} p_i &= \sum_{j\in D_i\cap A^{-i}} [B(d_j) - B(d_j^{-i})] + \sum_{j\in D_i\cap\overline{A^{-i}}} B(d_j) \\ &\quad + \left(\sum_{j\in(D_i\cap A^{-i})\cup\overline{T_i}} |X_j^{-i}| - \sum_{j\neq i} |X_j|\right) c \\ &= \sum_{j\in D_i\cap A^{-i}} [B(d_j) - B(d_j^{-i})] + \sum_{j\in D_i\cap\overline{A^{-i}}} B(d_j) \\ &\quad + \left(|X_i| - |D_i\cap\overline{A^{-i}}| - 1\right) c. \end{aligned} \tag{8}$$

Specifically, if node i is a leaf receiver, i.e., $D_i = \emptyset$ and $X_i = \emptyset$, the above payment can be simplified as

$$p_i^{\text{leaf}} = -c. \tag{9}$$

It suggests that a leaf receiver should be charged for the unit forwarding cost of its parent, which follows our intuition.

C. Welfare

Given a receiver's utility (4) and payment (8), its welfare is simply their sum, that is

$$\begin{aligned} w_i &= u_i + p_i \\ &= B(d_i) - |X_i|c + \sum_{j\in D_i\cap A^{-i}} [B(d_j) - B(d_j^{-i})] \\ &\quad + \sum_{j\in D_i\cap\overline{A^{-i}}} B(d_j) + (|X_i| - |D_i\cap\overline{A^{-i}}| - 1)c \\ &= \sum_{j\in T_i} B(d_j) - \sum_{j\in D_i\cap A^{-i}} B(d_j^{-i}) - (|D_i\cap\overline{A^{-i}}| + 1)c \end{aligned} \tag{10}$$

and the VCG mechanism ensures that the welfare is maximized when each receiver behaves honestly. Please note that the welfare is the optimization target for a selfish ALM node under the payment scheme instead of the utility.

V. Distributed Implementation and Practical Issues

We now present a practical distributed algorithm to implement the above cheat-proof mechanism. It is worth noting that the VCG mechanism is efficient, but not necessarily budget-balanced, i.e., the total payments to all agents can be positive or negative [26]. In this case, it is up to the media provider or the source node to bear the surplus or deficit of the payment policy. Note that the monetary micropayment can be only a very small fraction of the total income of the media provider.

A. Design Principles

The first design issue is where to store the payments. For sake of both scalability and trust, a specified third party is desired. Fortunately, even without a dedicated third-party server, the source node can act as the role because its benefit is associated with the overall outcome of the ALM tree while not that of individual nodes. The source node of an ALM session usually has strong computing and storing capacity to play the role as the third party. Unless otherwise specified, in the following we use source node to represent the trustworthy third party.

Then, we focus on the computation of payment for each receiver, which is key to implement the cheat-proof mechanism. Clearly, we cannot rely on a receiver itself to compute its payment because it could cheat to obtain a higher payment when reporting the computed results to the source node. An alternative is to compute the payment to node i by its parent q_i. This is unfortunately not trustworthy either because, according to (8), q_i does the calculation according to the reports from its descendants, which include node i.

To solve this problem, we suggest that the payment to node i be computed by its children, X_i. Intuitively, this choice works because the information demanded for computation will not pass through node i, so it has no chance to cheat to increase its payment. We now formally prove this desired property.

Theorem 1: Any rational node $j \in X_i$ has no motivation to cheat about the payment for node i when j reports this payment to the source node.

Proof: Let $u1$ be the utility of node j when it reports the payment for node i honestly, and $u2$ be the utility of node j when it cheats on node i's payment.

First, assume node j cheats by decreasing the payment for node i. In this case, the cheating may cause node i to leave from the multicast session. Since node j has chosen i as its parent, its welfare cannot be further improved by choosing another parent. Hence, $u2$ cannot be higher than $u1$.

Second, assume node j cheats by increasing the payment for node i. Note that the welfare of node j has already been maximized when it reports the actual payment for node i to the source node. Since the cheating behavior itself brings overhead to node j, $u2$ cannot be higher than $u1$ either.

Hence, node j cannot increase its own utility by cheating on node i's payment, and a rational node j has no motivation to cheat. **Q.E.D**.

We thus extend (8) into (11) to reflect this computation framework

$$\begin{aligned} p_i &= \sum_{j\in D_i\cap A^{-i}} [B(d_j) - B(d_j^{-i})] + \sum_{j\in D_i\cap\overline{A^{-i}}} B(d_j) \\ &\quad + (|X_i| - |D_i\cap\overline{A^{-i}}| - 1)c \\ &= \sum_{k\in X_i} \left\{ \begin{array}{l} \sum_{j\in T_k\cap A^{-i}} [B(d_j) - B(d_j^{-i})] + \sum_{j\in T_k\cap\overline{A^{-i}}} B(d_j) \\ \quad +(1 - |T_k\cap\overline{A^{-i}}|)c \end{array} \right\} \\ &\quad - c. \end{aligned} \tag{11}$$

Let $p_{ik1} = \sum_{j\in T_k\cap A^{-i}} [B(d_j) - B(d_j^{-i})]$, $p_{ik2} = \sum_{j\in T_k\cap\overline{A^{-i}}} B(d_j)$, $p_{ik3} = (1 - |T_k \cap \overline{A^{-i}}|)c$, and $p_{ik} = p_{ik1} + p_{ik2} + p_{ik3}$. Equation (11) can be simplified as

$$\begin{aligned} p_i &= \sum_{k\in X_i} \{p_{ik1} + p_{ik2} + p_{ik3}\} - c \\ &= -c + \sum_{k\in X_i} p_{ik}. \end{aligned} \tag{12}$$

Note that in our model we do not assume either the source node or any receiver maintains the global topology of the ALM

tree. Therefore, the computation for the payment to each receiver can only be conducted in a distributed way. Fortunately, from (12) we can let each child k of node i compute p_{ik} and report it to the source node. The source node then sums all p_{ik}, minus $-c$, to obtain the total payment for node i. A leaf node has no children to report the payment message for it, and thus its payment is simply $-c$, which also follows (9). In this manner, the computation burden put onto the source server is also limited.

Note that the payment computation and report are both periodical. Originally, the payment computation relies on a tree that might be established based on true or cheated distance information. However, given the payment-computation algorithm implementing the cheat-proof mechanism we design, all selfish nodes will be guided to behave honestly during the distance measurement to achieve its optimal welfare. Therefore, the ALM tree will be dynamically adjusted to a final optimal tree. We should also notice that the payment reports from ALM receivers to the source node are routed in the network layer, and other ALM nodes thus have no chance to alter them. The algorithm details are covered in Section V-B.

B. Algorithm Details

There are two types of messages exchanged in our algorithm, namely payment message and record message. The two types of messages are all sent periodically to suite the network dynamics. A payment message is sent from an ALM receiver to the source node, containing the payment about its parent. A record message is sent from an ALM receiver i to its parent q_i in the ALM tree, containing the record set of node i. More explicitly, let l_{ij} denote the distance from node i to node j in the ALM tree. The record set of node i includes the following record information for every node $j \in T_i$: d_j, the source-to-end distance of node j with the participation of parent q_j $-\circ$; $d_j^{-q_j}$, the source-to-end distance of node j when node q_j withdraws; and $l_{q_j j}$, the distance from q_j to j. We use a 3-tuple $(d_j, d_j^{-q_j}, l_{q_j j})$ to represent the record of node j. Note that since we assume the source node will bear the surplus or deficit of the ALM session, there is no need for the children of the source node to compute or report the payment for source node or send the record message to source node. According to Theorem 1, node i has no motivation to alter either the payment message or the record message.

We next give the details of the computation in Algorithms I–VI, among which Algorithms I–IV are executed in each ALM receiver and Algorithms V and VI are executed in the source node.

Algorithm I is for node i to compute the record set, which includes the records for all the nodes in its subtree, $T_i (= D_i + \{i\})$. The record $\{d_i, d_i^{-q_i}, l_{q_i i}\}$ is readily available. For each descendant $j \in D_i$, its record is computed as follows. The value of d_j is previously received from the children of node i. If node i has a second best parent node when node q_i withdraws from the multicast session, j will choose the same parent as before considering the stability of the ALM tree, and $d_j^{-q_i}$ is just the sum of $d_i^{-q_i}$ and l_{ij}. Otherwise, $d_j^{-q_i}$ is the same as d_j^{-i}, which node i has already received from its children. $l_{q_i j}$ is simply $l_{q_i i}$ plus l_{ij}, which is previously received from the children of node i.

Each node i periodically sends its record message to its parent, as shown in Algorithm II. Upon receiving a record message from one of its children, node i locally stores it for future computation of its own record and the payment to its parent, as illustrated in Algorithm III.

Algorithm I: Record-Set Computation on Receiver i

```
01 void recordSetCompute()
02   if d_i^{-q_i} ≠ ∞//i has a second best parent
03     for each j ∈ D_i
04       d_j^{-q_i} ← d_i^{-q_i} + l_ij// still select i as parent
05     end for
06   else //i has no other potential parents
07   for each j ∈ D_i
08   d_j^{-q_i} ← d_j^{-i}// same as when i withdraws
09     end for
10   end if
11   for each j ∈ D_i
12     l_qj ← l_qi + l_ij
13   end for
```

Algorithm II: Sending Record Message on Receiver i

```
01 void recordMsgSend()
02   recordSetCompute()
03   // recompute records before sending to parent
04   msg.type ← record // record message
05   msg.record ← Φ
06   for each j ∈ T_i
07     msg.record ← msg.record ∪ {d_j, d_j^{-q_i}, l_qj}
08   end for
09   msgSendToParent (msg)
```

Algorithm III: Receiving Record Message on Receiver i

```
01 void recordMsgReceive()
02   msg ← msgRcvFromChild()
03   s ← msg.source
04   if s ∈ X_i
05     for each j ∈ T_s
       // record message containing records
       of all nodes in T_s
06     d_j ← msg.d_j
07     d_j^{-i} ← msg.d_j^{-i}
08     l_ij ← msg.l_ij
09     end for
10   end if
```

Algorithm IV shows how node i computes $p_{q_i i}$ and sends the payment message containing $p_{q_i i}$ to the source node. The record set for computing $p_{q_i i}$ is obtained in Algorithm I. Upon receiving each $p_{q_i i}$, the source node stores it and updates the payment array that maintains the content of the payment message from each receiver. We should note here that the network is dynamic and the parent/children relationship among ALM nodes will change from time to time, so each entry in the payment array will timeout after a threshold value. The source node further computes the payment to each receiver in the ALM tree using the information in the payment array, as illustrated in Algorithm VI.

Algorithm IV: Sending Payment Message on Receiver i

```
01 void paymentMsgSend()
02   recordSetCompute()
03   // recompute records before computing the payment
     for parent
04   p_{q_i i1} ← 0  // Eq. (12)
05   p_{q_i i2} ← 0  // Eq. (12)
06   p_{q_i i3} ← 0  // Eq. (12)
07   r ← 0 // to calculate |T_i ∩ complement(A^{-q_i})|, Eq. (11)
08   for each j ∈ T_i
09     if d_j^{-q_i} ≠ ∞ // nodes in T_i ∩ A^{-q_i}
10       p_{q_i i1} ← p_{q_i i1} + B(d_j) − B(d_j^{-q_i})
11     else // nodes in T_i ∩ complement(A^{-q_i})
12       p_{q_i i2} ← p_{q_i i2} + B(d_j)
13       r ← r + 1
14     end if
15   end for
16   p_{q_i i3} ←(1 − r) ∗ c
17   p_{q_i i} ← p_{q_i i1} + p_{q_i i2} + p_{q_i i3}
18   msg.type ← payment // payment message
19   msg.parent ← q_i
20   msg.payment ← p_{q_i i}
21   msgSendtoSource (msg)
```

Algorithm V: Receiving Payment Message on the Source Node

```
01 void paymentMsgReceive()
02   msg ← msgRcvFromReceivers()
03   s ← msg.source
04   payArr[s].parent ← msg.parent
05   payArr[s].value ← msg.payment
06   payArr[s].timeout ← threshold
```

Algorithm VI: Payments Computation on the Source Node

```
01 void paymentsCompute()
     // assume source node maintains all receivers
02   for each receiver j
03     p_j ← −c  // Eq. (11)
04   end for
05   for each receiver j
06     k ← payArr[j].parent
07     p_k ← p_k + payArr[j].value  // Eq. (11)
08   end for
```

C. Algorithm Complexity

We now analyze the algorithm complexity of our implementation, with a focus on the following important measures: computation overhead, storage overhead, and communication overhead.

1) Computation Overhead: The computation overhead of each receiver comes from two parts, namely computing the record set and computing the payment about its parent.

From function recordSetCompute() in Algorithm I, we can see that the grandson of the source node has the highest record-set computation load to compute the record set for the children of the source node, which is $O(n)$. All other receivers usually have only a sublinear overhead. The average overhead of all receivers is $O(\log n)$ in common cases and not worse than $O(n)$, which rarely happens. (The detailed derivation is omitted here for space limitation). From function paymentMsgSend() in Algorithm IV, we can see that the computation overhead of payment computation is almost the same as that of record-set computation.

The computation load of the source node comes only from computing the payment to all receivers in the ALM tree. According to the function paymentsCompute() in Algorithm VI, the computation load on the source node is $O(n)$.

2) Storage Overhead: From function recordMsgReceive() in Algorithm III, each ALM receiver except the children of the source node needs to store its record set. Thus, the maximum storage load $O(n)$ is again at the grandson of the source node. The average storage load of all ALM receivers usually remains $O(\log n)$ and not worse than $O(n)$, which rarely happens.

From function paymentMsgReceive() in Algorithm V, we can see that the source node needs to maintain a payment array that contains the payment message information from all receivers, the size of which is $O(n)$.

3) Communication Overhead: There are two types of messages in our algorithm: the record message and the payment message. Function recordMsgSend() in Algorithm II suggests that the maximum communication overhead is incurred on the overlay paths connecting the source node and its grandson, which is $O(n)$. Meanwhile, the average communication overhead for all paths in the ALM tree is usually $O(\log n)$ and not worse than $O(n)$, which rarely happens.

The payment message is sent from an ALM receiver to the source node, containing the payment it has computed about its parent. According to function paymentMsgSend() in Algorithm IV, the communication overhead on each path connecting an ALM node and the source node is $O(1)$ only. Therefore, the communication overheads on the ALM paths are very low.

VI. Simulations

We conduct extensive simulations to study the performance of our implementation of the cheat-proof mechanism. Unless otherwise specified, the following default settings are adopted in our simulations. We use the GT-ITM toolkit [27] to generate network-layer topologies that consists of links and routers. In each topology, there are 2000 routers, and the link distances between connected routers are uniformly distributed within $[10\text{ ms}, 100\text{ ms}]$. The n receivers in the ALM as well as the source node are attached to $n+1$ randomly drawn routers. We set $B(d_i) = 500/d_i$ and $c = 0.5$, which follows that $u_i = 500/d_i - 0.5*|X_i|$. The quantification of the utility function will not affect the generalization of the results. The ALM tree is built using the shortest-path-tree algorithm. To mitigate the impact of randomness, 50 topologies are generated for each simulation, and the results are averages over them.

We present the overall utility of ALM tree, the welfare of individual receivers, and the payment to individual receivers in Sections VI-A–C.

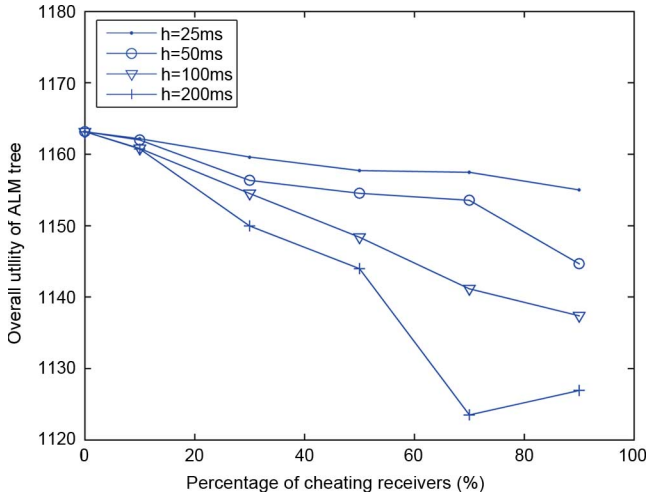

Fig. 2. Overall utility. $n = 500$ and $e = 60\%$.

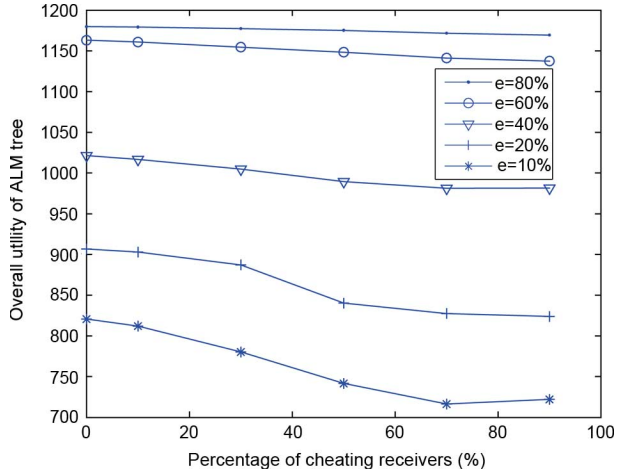

Fig. 3. Overall utility. $n = 500$ and $h = 100$ ms.

A. Overall Utility

In the first set of simulations, we compare the overall utilities of ALM trees to different amounts of cheating receivers. We fix the total number of receivers to 500 and examine the impact of different cheating degrees and the neighbor densities. Specifically, we first set the neighbor density e to 60% and vary the cheating degree h, i.e., the value that a cheating receiver adds to its actual distances to others, from 25 to 200 ms. We then fix the cheating degree h to 100 ms and gradually change the neighbor density e from 10% to 80%.

Figs. 2 and 3 plot the corresponding overall utilities of the ALM trees for the above two settings. In both figures, the overall utility is always maximized when the percentage of cheating receivers is 0% and decreases with higher percentage of cheating receivers. This is not surprising given that the existing studies have shown an ALM tree is optimized with no node cheating. Note that the trends are not necessarily monotonic; see, for example, the line of $h = 200$ ms in Fig. 2. We conjecture that this is because, when there are a lot of cheating receivers, some of them might counteract with each other. Fig. 2 demonstrates that the overall utility is decreasing with higher cheating degree, implying that distance exaggeration will negatively influence the quality of parent selection. On the other hand, Fig. 3 shows that the overall utility is also decreasing with lower neighbor density. This result suggests that, with a limited parent choice, the adverse impact of distance cheating can be more remarkable.

B. Welfare of Individual Receivers

We next investigate the welfare of individual receivers under our cheat-proof mechanism. We define the truth gain of a receiver as the difference of its welfare between telling the truth and cheating, assuming that all other nodes tell the truth. Again, we fix the number of receivers to 500 and then vary the cheating degree and the neighbor density, respectively.

Figs. 4 and 5 give the corresponding truth gains of individual receivers for each aforementioned setting. We can see that the truth gain of an individual receiver is always nonnegative, which suggests that it achieves its maximum welfare by behaving honestly during distance measurement. Apparently, any rational receiver will tell the truth when its truth gain is nonnegative (we assume that a node has no motivation to cheat if the truth gain is 0, for the cheating behavior itself would incur overhead). There are some peaks in these figures, corresponding to the nodes serving a large group of children. Their welfares are significantly decreased if they cheat to avoid accepting children because of the huge payment loss. The figures also suggest that the truth gain increases when the cheating degree is higher or when the neighbor density is lower. In summary, our mechanism effectively defends against distance cheating.

C. Payment to Individual Receivers

Since the payment to a node is the difference between the total utility of all other nodes with and without the participation of this node, the average payment thus reflects the average impact of an individual receiver's dynamics to the whole multicast session. Such payment is closely related to the session size and neighbor density. Hence, we generate multicast sessions with the total number of receiver ranging from 100 to 1000 and the neighbor density from 10% to 80%. Fig. 6 shows the average payment as a function of such session sizes and neighbor densities.

We can see that the average payment decreases when there are more receivers because a single receiver's impact on the total utility becomes relatively smaller. We also see that with the growth of neighbor density, the average payment first increases, but then decreases. This is because the relationship between the neighbor density and the payment lies in two aspects: On the one hand, if a node has more neighbors, it has more chances to affect others; on the other hand, more neighbors implies that a node has more potential parents and thus suffers less when its current parent leaves. Fig. 6 implies that when the neighbor density is relatively low, the former is more noticeable, while the latter becomes dominating when the density is high.

VII. Conclusion

In this paper, we have proposed a cheat-proof mechanism to defend against distance cheating, which presents a practical mapping from the utility, payment, and welfare of the VCG mechanism to the link-weighted ALM context. The mapping ensures that each selfish ALM receiver has to behave honestly during distance measurement to maximize its own welfare. Based on the cheat-proof mechanism, we then demonstrate a trustworthy distributed implementation. Our implementation

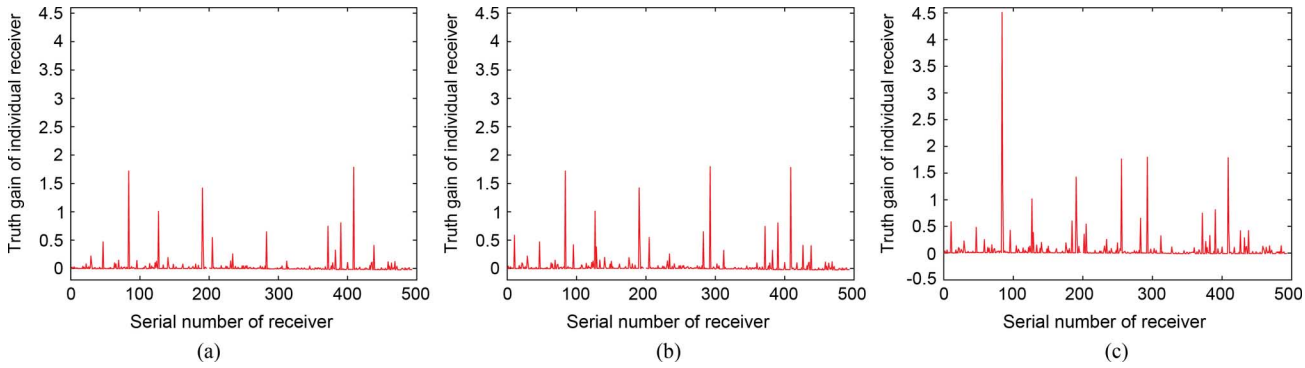

Fig. 4. Truth gains of individual receivers. $n = 500$ and $e = \%$. (a) $h = 50$ ms. (b) $h = 100$ ms. (c) $h = 200$ ms.

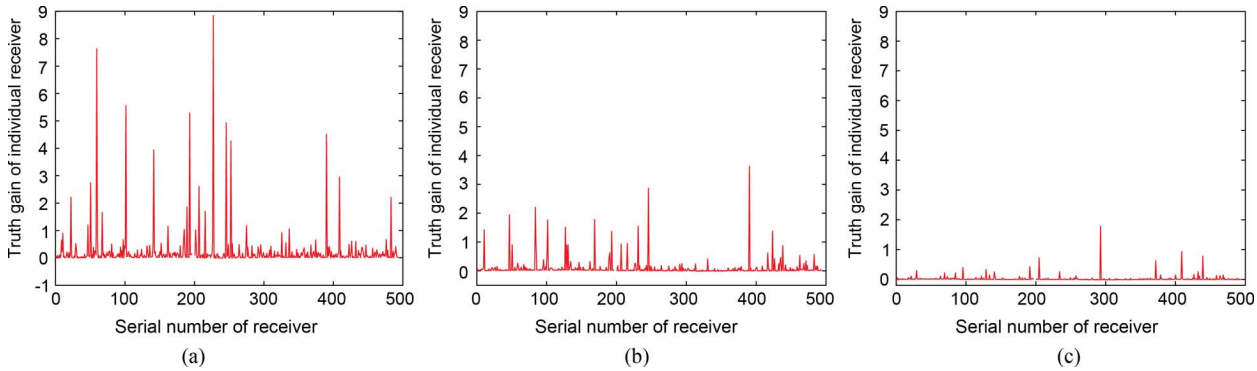

Fig. 5. Truth gains of individual receivers $n = 500$ and $h = 100$ ms. (a) $e = 10\%$. (b) $e = 40\%$. (c) $e = 80\%$.

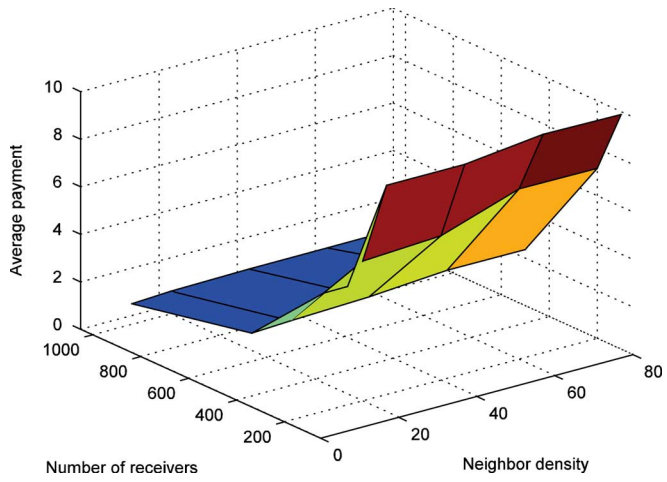

Fig. 6. Average payment as a function of session size and neighbor density.

also effectively prevents a selfish node from cheating the payment-computation process. Meanwhile, its computation, storage, and communication overheads are all controlled at low levels. Such cheat-proof design has also been verified by extensive simulations.

We expect more advanced solutions introduced to migrate our proposal to encompass collusion, Sybil attack, and large-scale ALM session. This is also our future work.

ACKNOWLEDGMENT

Some preliminary results of this work were presented in [28] and [29]. In this paper, we make the following improvements. First, we revisited the original problem and added a section for the problem statement and system model. Second, we improved the algorithm design, which is more light-weighted and more scalable. Third, we conducted more simulations to study the effectiveness of our algorithm.

REFERENCES

[1] S. E. Deering, "Multicast routing in internetworks and extended LANs," in *Proc. ACM SIGCOMM*, Stanford, CA, Aug. 1988, pp. 55–64.
[2] L. Mathy and N. Blundell, "Impact of simple cheating in application-level multicast," in *Proc. IEEE INFOCOM*, Hong Kong, Mar. 2004, vol. 2, pp. 1318–1328.
[3] Y. H. Chu, S. G. Rao, and H. Zhang, "A case for end system multicast," in *Proc. ACM SIGMETRICS*, Santa Clara, CA, Jun. 2000, pp. 1–12.
[4] S. Banerjee, B. Bhattacharjee, and C. Kommareddy, "Scalable application layer multicast," in *Proc. ACM SIGCOMM*, Pittsburgh, PA, Aug. 2002, pp. 205–217.
[5] D. A. Tran, K. A. Hua, and T. Do, "ZIGZAG: An efficient peer-to-peer scheme for media streaming," in *Proc. IEEE INFOCOM*, San Franciso, CA, 2003, vol. 2, pp. 1283–1292.
[6] M. Castro, P. Druschel, A. Kermarrec, A. Nandi, A. Rowstron, and A. Singh, "Splitstream: High-bandwidth content distribution in cooperative environments," in *Proc. ACM SOSP*, Bolton Landing, NY, Oct. 2003, pp. 298–313.
[7] D. Kostic, A. Rodriguez, J. Albrecht, and A. Vahdat, "Bullet: High bandwidth data dissemination using an overlay mesh," in *Proc. ACM SOSP*, Bolton Landing, NY, Oct. 2003, pp. 282–297.
[8] X. Zhang, J. Liu, B. Li, and T. P. Yum, "CoolStreaming/DONet: A data-driven overlay network for peer-to-peer live media streaming," in *Proc. IEEE INFOCOM*, Miami, FL, Mar. 2005, vol. 3, pp. 2102–2111.
[9] D. Li, Y. Cui, K. Xu, and J. Wu, "Impact of receiver cheating on the stability of ALM tree," in *Proc. IEEE GLOBECOM*, St. Louis, MO, 2005, vol. 2, pp. 667–671.
[10] W. Vickrey, "Counterspeculation, auctions and competitive sealed tenders," *J. Finance*, pp. 8–37, 1961.
[11] E. H. Clarke, "Multipart pricing of public goods," *Public Choice*, vol. 11, pp. 17–33, 1971.

[12] T. Groves, "Incentives in teams," *Econometrica*, vol. 41, no. 4, pp. 617–631, 1973.
[13] S. Yuen and B. Li, "Strategyproof mechanisms for dynamic multicast tree formation in overlay networks," in *Proc. IEEE INFOCOM*, Miami, FL, Mar. 2005, vol. 3, pp. 2135–2146.
[14] W. Wang, X. Li, Z. Suny, and Y. Wang, "Design multicast protocols for non-cooperative networks," in *Proc. IEEE INFOCOM*, Miami, FL, Mar. 2005, vol. 3, pp. 1596–1607.
[15] A. Habib and J. Chuang, "Incentive mechanism for peer-to-peer media streaming," in *Proc. IEEE IWQoS*, Montreal, QC, Canada, Jun. 2004, pp. 171–180.
[16] G. Robins and A. Zelikovsky, "Improved Steiner tree approximation in graphs," in *Proc. ACM SODA*, San Francisco, CA, Jan. 2000, pp. 770–779.
[17] H. Takahashi and A. Matsuyama, "An approximate solution for the Steiner problem in graphs," *Math Jpn.*, vol. 24, pp. 573–577, 1980.
[18] T. Moscibroda, S. Schmid, and R. Wattenhofer, "On the topologies formed by selfish peers," in *Proc. PODC*, New York, 2006, pp. 133–142.
[19] N. Nisan and A. Ronen, "Algorithmic mechanism design," *Games Econ. Behav.*, vol. 35, pp. 166–196, 2001.
[20] J. Feigenbaum and S. Shenker, "Distributed algorithmic mechanism design: Recent results and future directions," in *Proc. ACM Dial-M*, Atlanta, GA, Sep. 2002, pp. 1–13.
[21] J. Feigenbaum, C. Papadimitriou, R. Samiy, and S. Shenker, "A BGP-based mechanism for lowest-cost routing," in *Proc. ACM PODC*, Jul. 2002, pp. 173–182.
[22] J. Feigenbaum, C. Papadimitriou, and S. Shenker, "Sharing the cost of multicast transmissions," *J. Comput. Syst. Sci.*, vol. 63, pp. 21–41, 2001.
[23] A. Archer, J. Feigenbaum, A. Krishnamurthy, R. Sami, and S. Shenker, "Approximation and collusion in multicast cost sharing," in *Proc. ACM EC*, San Diego, CA, Jun. 2003, p. 280.
[24] M. Bläser, "Approximate budget balanced mechanisms with low communication costs for the multicast cost-sharing problem," in *Proc. ACM SODA*, New Orleans, LA, Jan. 2004, pp. 625–626.
[25] J. Feigenbaum, A. Krishnamurthy, R. Sami, and S. Shenker, "Hardness results for multicast cost sharing," *Theor. Comput. Sci.*, vol. 304, no. 1–3, pp. 215–236, 2003.
[26] A. M. Colell, M. Whinston, and J. Green, *Microeconomic Theory*. New York: Oxford Univ. Press, 1995, ch. 23.
[27] E. Zegura, K. Calvert, and S. Bhattacharjee, "How to model an internetwork," in *Proc. IEEE INFOCOM*, San Francisco, CA, Mar. 1996, vol. 2, pp. 594–602.
[28] D. Li, W. Jianping, C. Yong, L. Jiangchuan, and X. Ke, "Trustworthy distributed algorithm design to defend distance cheating in link-weighted ALM," in *Proc. ISCIS*, Istanbul, Turkey, Nov. 2006, pp. 844–853.
[29] D. Li, W. Jianping, C. Yong, L. Jiangchuan, and X. Ke, "A VCG motivated cheat-proof mechanism against distance cheating of ALM receivers," in *Proc. ICOIN*, Estoril, Portugal, 2007, pp. 154–163.
[30] D. Li, J. Wu, and Y. Cui, "Defending against buffer map cheating in DONet-likeP P2P streaming," *IEEE Trans. Multimedia*, vol. 11, no. 3, pp. 535–542, Apr. 2009.
[31] D. Li, J. Wu, Y. Cui, and J. Liu, "QoS-aware streaming in overlay multicast considering the selfishness in construction action," in *Proc. IEEE INFOCOM*, Anchorage, AK, 2007, pp. 1154–1162.
[32] X. Xiao, Y. Shi, Y. Tang, and N. Zhang, "Cheating and anti-cheating in gossip-based protocol: An experimental investigation," *IEICE Trans.*, vol. E-91-B, no. 9, pp. 2856–2863, 2008.
[33] K. Kitayama, K. Takahashi, and M. Yamamoto, "Performance evaluation of cheating host in ALM over wireless multi-hop networks," in *Proc. APCC*, Shanghai, China, pp. 812–815.

Dan Li (M'10) received the Ph.D. degree in computer science from Tsinghua University, Beijing, China, in 2007.

He is now an Assistant Professor with the Department of Computer Science, Tsinghua University. Before joining the faculty of Tsinghua University, he spent two years as an Associate Researcher with the Wireless and Networking Group, Microsoft Research Asia, Beijing, China. His research interest spans from Internet architecture and protocols, P2P networks, to cloud computing networks.

Dr. Li is a member of the Association for Computing Machinery (ACM).

Jianping Wu (SM'05) received the Master's and Ph.D. degrees in computer science from Tsinghua University, Beijing, China.

He is now a Full Professor with the Department of Computer Science, Tsinghua University. He has published more than 200 technical papers in academic journals and proceedings of international conferences in the research areas of the network architecture, high-performance routing and switching, protocol testing, and formal methods.

Jiangchuan Liu (S'01–M'03–SM'08) received the B.Eng. degree (*cum laude*) from Tsinghua University, Beijing, China, in 1999, and the Ph.D. degree from the Hong Kong University of Science and Technology, Hong Kong, in 2003, both in computer science.

He is currently an Associate Professor with the School of Computing Science, Simon Fraser University, Vancouver, BC, Canada. His research interests include multimedia systems and networks, wireless ad hoc and sensor networks, and peer-to-peer and overlay networks.

Dr. Liu is a member of Sigma Xi. He is an Associate Editor of the IEEE TRANSACTIONS ON MULTIMEDIA, an Editor of *IEEE Communications Surveys and Tutorials*, and an Area Editor of *Computer Communications*. He is Technical Program Committee Vice Chair for Information Systems of IEEE INFOCOM 2011. He is a recipient of a Microsoft Research Fellowship in 2000, the Hong Kong Young Scientist Award in 2003, and the Canada NSERC DAS Award in 2009. He is a co-recipient of the Best Student Paper Award of IWQoS 2008, the 2009 Best Paper Award of the IEEE ComSoc Multimedia Communications Technical Committee, and the Canada BCNet Broadband Challenge Winner Award in 2009.

Yong Cui received the B.S., M.E., and Ph.D. degrees from Tsinghua University, Beijing, China, in 1999, 2001, and 2004, respectively.

He is an Associate Professor with Tsinghua University, and a Council Member in the China Communication Standards Association. He directed several international and national R&D projects. Having published more than 80 papers, he also holds more than 20 patents. He is one of the authors of RFC 5747 and RFC 5565 on IPv6 transition technologies. His major research interests include computer network architecture and mobile wireless computing.

Dr. Cui won the National Science and Technology Progress Award (Second Class) in 2005 and the Influential Invention Award of China Information Industry in 2004.

Ke Xu (M'02–SM'09) received the B.S., M.S., and Ph.D. degrees in computer science from Tsinghua University, Beijing, China, in 1996, 1998, and 2001, respectively.

Currently, he is a Professor with the Department of Computer Science, Tsinghua University. His research interests include next-generation Internet, switch and router architecture, P2P, and overlay network.

Prof. Xu is a member of the Association for Computing Machinery (ACM).